Xu, Guoning; Lin, Yupeng; Wu, Zhiying; Chen, Qingxin; Mao, Ning

Article

Research on the scheduling method of ground resource under uncertain arrival time

Operations Research Perspectives

Provided in Cooperation with:
Elsevier

Suggested Citation: Xu, Guoning; Lin, Yupeng; Wu, Zhiying; Chen, Qingxin; Mao, Ning (2023) : Research on the scheduling method of ground resource under uncertain arrival time, Operations Research Perspectives, ISSN 2214-7160, Elsevier, Amsterdam, Vol. 11, pp. 1-11, https://doi.org/10.1016/j.orp.2023.100291

This Version is available at:
https://hdl.handle.net/10419/325776

Operations Research Perspectives 11 (2023) 100291

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

Research on the scheduling method of ground resource under uncertain arrival time

Guoning Xu, Yupeng Lin *, Zhiying Wu, Qingxin Chen, Ning Mao

Key Laboratory of Computer Integrated Manufacturing System, Guangdong University of Technology, Guangdong Province, PR China

ARTICLE INFO

Keywords:
Air transport
Airport ground resource
Proactive scheduling
Improved adaptive large neighborhood search
Algorithm
Arrival time uncertain

ABSTRACT

We present a two-stage scheduling approach including proactive and reactive scheduling to solve the ground resource scheduling problem with uncertain arrival time. In the first stage, an integer programming model is constructed to minimize the delay and transfer costs. After solving this model, we obtain a baseline scheduling plan that considers the service arrival time uncertainty. In the second stage, the feasibility of the subsequent benchmark plan is evaluated based on the current state of the services and resources. The reactive scheduling model is enabled when trigger conditions are met. Moreover, an improved adaptive large neighborhood search is designed to solve the proactive scheduling model effectively. Real data from an international airport in South China is used as a test case to compare different scheduling strategies. The results show that it is difficult to handle the uncertainty of the problem with the benchmark plan that simply considered buffer time. Compared with rolling time-domain scheduling, the average transfer cost of the scheduling strategy proposed in this paper increased slightly, but the average service delay cost can be reduced significantly. Algorithm-wise, instances of different scales are designed to verify the effectiveness of the improved adaptive large neighborhood search algorithm. The efficiency of the algorithm scheme is better than that of the Gurobi solver scheme in medium to large-scale problems. Therefore, the forward and reactive strategies can better handle the uncertainty of airport ground protection services as they can simultaneously guide the allocation and utilization of airport ground protection resources.

1. Introduction

With the development of the civil aviation industry, the number of flights has increased, thereby increasing pressure on airport ground resource scheduling. Many important ground services are received during flight transit, and these services are executed by different ground resources. Most domestic civil airports employ resource division and manual scheduling, and different types of resources are managed and scheduled by their business units. Before decision-making, the dispatchers can determine the duration of each ground service and the number of resources required based on the international/domestic attributes including the flight, aircraft size, and airline service manual. The flight schedule is a weekly cycle. There is some deviation between the flight plan arrival and departure times and the actual times. The weekly baseline scheduling plan based on the flight plan is difficult to guide the field work. Frequent adjustments bring troubles to the work of dispatcher. Therefore, dispatcher is more willing to prefer online scheduling. When decisions have to be made, scheduling is done in conjunction with flights and resources in real time. The dispatcher predicts the start time of ground services based on the scheduled arrival and departure times of the flight and the current status of the flight. The core idea of this scheduling method is online scheduling, which constantly identifies upcoming services and dispatches available resources to them individually.

The research results of the airport ground resource scheduling problem mainly revolve around ferry trucks, fuel trucks, baggage trucks, and deicing equipment. They can be divided into classic assignment models, vehicle routing problems with time windows (VRPTW) models, and network structure models in model construction. Zhao et al. [1] constructed a classical assignment model for the cooperative scheduling problem of the ferry and tractor. This type of model is preferred as the basis when individual resource variability is considered. Otherwise, the VRPTW model is mainly used as the core. Ip et al. [2] and Chen et al. [3] studied collaborative resource scheduling and centralized scheduling, respectively, based on the VRPTW model. However, the VRPTW model often leads to inefficient decision-making when there are many types of

* Corresponding author.
E-mail address: 506230668@qq.com (Y. Lin).

https://doi.org/10.1016/j.orp.2023.100291
Received 23 May 2023; Received in revised form 9 November 2023; Accepted 9 November 2023
Available online 15 November 2023

ground resources and concentrated flight arrivals. To solve this problem, researchers [4] proposed the temporal constraints level procedure (TCLP) method, which first coordinates the situation of each type of service time, then decomposes the problem into several subproblems, and finally solves it by the variable neighborhood search method. In the static scheduling problem of the ferry, researchers [5] used the maximum flow network model to transform the original problem, greatly improving the efficiency of problem-solving. In addition to the above, given the special characteristics of deicing equipment, queueing theory methods are often used to solve the problem. For the problem of flight delays caused by deicing services, researchers [6] proposed a decision algorithm and collaborative algorithm for the distribution of multiple deicing areas to reduce the delay duration of flights to a certain extent. The research results mentioned above are all from studies of deterministic problems.

As large and complex service systems, airports have the characteristic of airports. The research for ground service uncertainty is an important challenge [7]. A recent article reviewing the airport ground scheduling problem states that only a small number of studies consider problem uncertainty in airside scheduling problems [8]. When Schmidt reviewed the current research status of aircraft turnaround modeling, it was found that only a few scholars have conducted research on uncertainty issues [9]. Moreover, several different ideas have been used to consider problem uncertainty. The first idea is to predict the uncertainty of the problem. Alexander et al. [10] designed a fuzzy-rule-based adaptive algorithm to estimate the uncertainty of the aircraft taxiing time and path. The second idea is robust scheduling. Robust scheduling is a scheduling strategy that aims to maintain the stability and performance of task scheduling in the face of uncertainties and perturbations. The strategy focuses on reducing the impact of uncertainties on the scheduling scheme by taking preventive measures. Among them, robust scheduling can include methods such as robust optimization and stochastic planning. Robust optimization, as an optimization method of robust scheduling, aims to find the optimal solution with strong robustness to uncertainty under consideration of uncertainty. Liang et al. [11] pointed out that robust optimization describes the fluctuation parameters through uncertain sets and search a solution feasible for all possible implementation cases while minimizing the worst-case objective functions. Diepen et al. [12] maximized the idle time between services to provide a more robust scheme for the ferry. Stochastic planning, as another method of robust scheduling, mainly consists of expected value modeling and chance constrained planning. Uncertainty in ground service duration and resource transfer time seriously affected the enforceability of scheduling schemes. Padrón et al. [13] built a target probability model to describe this type of stochastic problem. They generated many scenarios randomly by the Monte Carlo simulation method, and maximized the number of scenarios completed on time. The random arrival of flights is a very important influence on scheduling. Zhu et al. [14] described the uncertainty of the problem with chance constraints. The stochastic model can describe the uncertainty of the problem well. However, it is easily limited by the sample size. The uncertainty distribution pattern can be analyzed by historical data, but it is difficult to generate scenario tree probabilities. The description of the scenario with equal probability is affected by factors such as the sample size and algorithm convergence. The efficiency of the solution limits the practical application of stochastic models. The third idea is proactive and reactive scheduling. In the development of the flight transit scheme, owing to the variability of the flight plan, researchers [15] divided the problem into two phases. In a long planning phase, which determines the daily resource allocation quantity based on the flight plan. In a short planning phase, which revises the resource allocation quantity based on the weekly updated flight information. This scheduling method is proactive and reactive scheduling (PRS). PRS is commonly found in the field of project scheduling research, and it mainly includes two stages: proactive scheduling and reactive scheduling. In the proactive scheduling stage, it is necessary to consider the comprehensiveness of the problem as much as possible to provide a global benchmark plan. The reactive scheduling phase is triggered when the benchmark plan is not applicable, and the plan is quickly adjusted. This method has high applicability in airport ground support resource scheduling problems.

In the airport ground scheduling problem, methods such as proactive scheduling and reactive scheduling are used relatively rarely. However, the two methods are widely used in ship transportation and manufacturing industries. Reactive scheduling is an online/real-time scheduling method. The advantages and disadvantages of the scheduling scheme are influenced by the field situation. Xiang et al. [16] proposed a reaction strategy for the problem of the simultaneous assignment of berths and quay cranes with uncertainty. The main reason for adopting this strategy is the deviation of the ship arrival time, which has a disruptive impact on the scheduling plan. This is highly similar to the uncertainty of the arrival and departure times of flights in this article. Owing to the consideration that the resource transfer time cannot be ignored in this article, the scheduling lead time affects the applicability of reactive scheduling. Proactive scheduling is a commonly used offline scheduling method. By considering the uncertainty of the problem as much as possible, scheduling schemes with robust effects are generated. Cui et al. [17] proposed a proactive scheduling model in optimizing a flow shop with fault uncertainty; operations on the flight deck of an aircraft carrier were performed in an uncertain environment. Further, Su et al. [18] proposed a proactive robust optimization method for aircraft deck scheduling with a stochastic running time. These references all adopt proactive scheduling methods to generate more robust scheduling schemes. Owing to the inevitable deviation between the planned arrival and departure time of the flight and the actual time, proactive scheduling alone makes it difficult to address the problem studied in this article. Kang et al. [19] used the PRS strategy framework to deal with the uncertainty of container ship arrival and towing process time. In the proactive scheduling stage, the expected variability and uncertainty factors in the execution of tugboat formation scheduling are considered; in the reactive scheduling stage, the baseline scheduling plan is appropriately adjusted to handle unexpected situations with minimum recovery cost. Tan et al. [20] studied the optimization of berth allocation and quay crane allocation under the uncertainty of the vessel arrival time and the fluctuation of loading and unloading volume. The proactive scheduling stage determines the baseline plan, and the reaction scheduling strategy responds to the disturbances during execution. In the airport ground resource scheduling problem, a certain lead time is required for scheduling decisions because the resource transfer time is not negligible. Considering the lack of foresight and global of reactive scheduling, and the difficulty of proactive scheduling to handle field disturbances, neither of these two methods can effectively solve the problem of this paper. The PRS decision framework can handle the arrival time uncertainty, satisfy the decision lead time, and also adjust the scheduling scheme according to the field situation. Algorithm-wise, most existing solution methods use approximate algorithms to solve the problem. Among the deterministic problems, common meta-heuristic algorithms are the genetic algorithm [2,3] and the particle swarm algorithm [1]. In stochastic problems, Monte Carlo simulation methods [17] are used, or uncertainty constraints are transformed into linear constraints before solving. Additionally, researchers [4,10]used a variable neighborhood search method.

This paper studies the scheduling problem of a single type of ground resources during flight transit. The airport operation and control (AOC) center decouples the service structure based on the service standard, the flight planned arrival and departure times. After completing the decomposition, the planned arrival time and service duration of each service are determined. The service duration is relatively certain, in this paper does not consider the variation of service duration. In the actual scheduling, the deviation between the flight plan time and the actual time cannot be negligible. According to the historical times of flights, it can be known that the deviation time obeys a certain distribution. Correspondingly, it can be assumed that the service plan arrival time

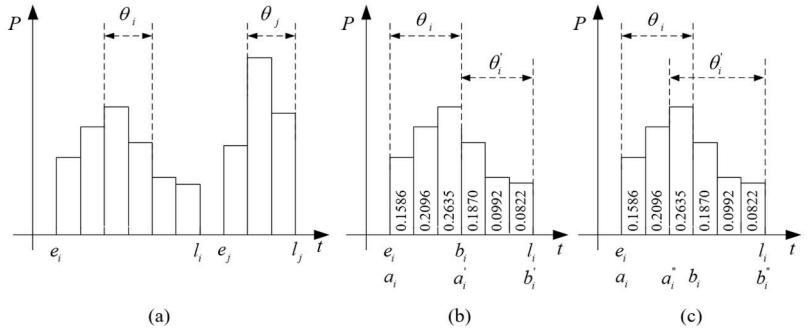

Fig. 1. Relationship between **θ** and **η**.

also obeys the same distribution. The lower limit of the distribution is defined as the earliest arrival time of the service, and the upper limit is defined as the latest arrival time. Since the airport ground is extensive, and resource scheduling needs to consider the necessary transfer time. With the requirement of decision lead time, it results in not being able to make decisions after the service arrival time is fully determined. For this reason, the PRS decision framework is introduced. In the proactive scheduling phase, a model based on the concept of uncertainty coverage is constructed. An improved ALNS algorithm is designed to improve the solution efficiency. In the reactive scheduling phase, design a scheduling model based on conditional triggering. Time advances to monitor the baseline schedule execution and determine whether the service will incur unplanned delay costs. Reactive scheduling is initiated to adjust the baseline program if necessary.

The subsequent sections of this paper are roughly arranged as follows. In Section 2, the model of each stage in the PRS decision framework is established, while the model of the pre-registration scheduling stage is adjusted. In Section 3, an improved ALNS algorithm is designed. In Section 4, a case study is conducted to compare the scheduling effects of different scheduling methods using the data of an international airport as an analysis case. Finally, conclusions are drawn in Section 5.

2. Problem description and model construction

2.1. Problem description and traditional proactive scheduling model

The following problem assumptions were made:

- The resource transfer time is only related to the distance between the aircraft positions;
- A constant number of resources in the decision cycle;
- The service arrival time uncertainty distribution is known;

The airport ground resource scheduling problem can be described as follows: In the decision cycle, several flights need to execute transit services, where the set of a type of services is $N-\{0\}$; the index numbers of subscripts are denoted by i and j; $i.j \in N, i \neq j$; and the total number of services is n. The set of resources corresponding to this type of service is G, the total number of resources is g, and the index number is denoted by k. If resource k executes service i followed by service j, then the decision variable $(x_{i,j,k})$ takes the value 1; otherwise, it takes the value 0. $t_{i,j}$ is used to denote the transfer time between service i and service j. The scheduling scheme requires that the number of resources and the number of services demanded (r_i) coincide. Note that when $i = 0$, it specifically refers to the virtual service with a duration of 0. At the initial decision time, all of the resources are sourced from the virtual service. Use w_i to denote the service duration of service i with $w_0 = 0$ and $r_0 = g$. Before constructing the proactive scheduling model (PSM), we add the following parameters and decision variables:

Parameters

α: unit time delay cost;
β: unit time transfer cost;
e_i: the earliest time of arrival of service i;
l_i: the latest arrival time of service i;
θ_i: the buffer time for service i;
M: a very large number;

Decision variables

s_i: the scheduled start time of service i;
d_i: the length of delay for service i, taking a nonnegative value;

PSM:

$$\min\left(\sum_{i\in N}\alpha d_i + \sum_{i\in N}\sum_{j\in N}\sum_{k\in G}\beta t_{i,j}x_{i,j,k}\right) \tag{1}$$

s.t.

$$\sum_{i\in N}\sum_{k\in G}x_{i,j,k} = r_j, \forall j \in N \tag{2}$$

$$s_i + w_i + \theta_i + t_{i,j} + M\left(x_{i,j,k} - 1\right) \leq s_j, \ \forall i,j \in N, k \in G \tag{3}$$

$$\sum_{j\in N}x_{0,j,k} = 1, \forall k \in G \tag{4}$$

$$\sum_{i\in N}x_{i,0,k} = 1, \forall k \in G \tag{5}$$

$$\sum_{i\in N}x_{i,j,k} - \sum_{i\in N}x_{j,i,k} = 0, \ \forall j \in N-\{0\}, k \in G \tag{6}$$

$$s_i \leq l_i + d_i, \forall i \in N \tag{7}$$

$$e_i \leq s_i, \forall i \in N \tag{8}$$

where Eq. (1) is the optimization objective to minimize the service delay time and resource transfer time cost; Eq. (2) indicates that all of the services meet the resource quantity demand; The number of resources required by service j is r_j. When any resource k is transferred from any service i to j, the decision variable $x_{i,j,k} = 1$, where $i \neq j$. i can take the value of the source service when j is not the source service, so as to ensure that sufficient resources are available. Eq. (3) indicates that the same resource can only execute one service at the same time; Eqs. (4) and (5) indicate that all of the resources start from the virtual service and finally return to the virtual service; Eq. (6) indicates that after a resource finishes a service, it must go to the next service from the current service; Eqs. (7) and (8) indicate the service plan start time, which needs to start within a limited time window; otherwise, the exceeding part is understood as service delay.

2.2. Buffer time and uncertainty coverage

In practical problems, the type of flight and the size of the aircraft type may lead to differences in the distribution of ground service arrival

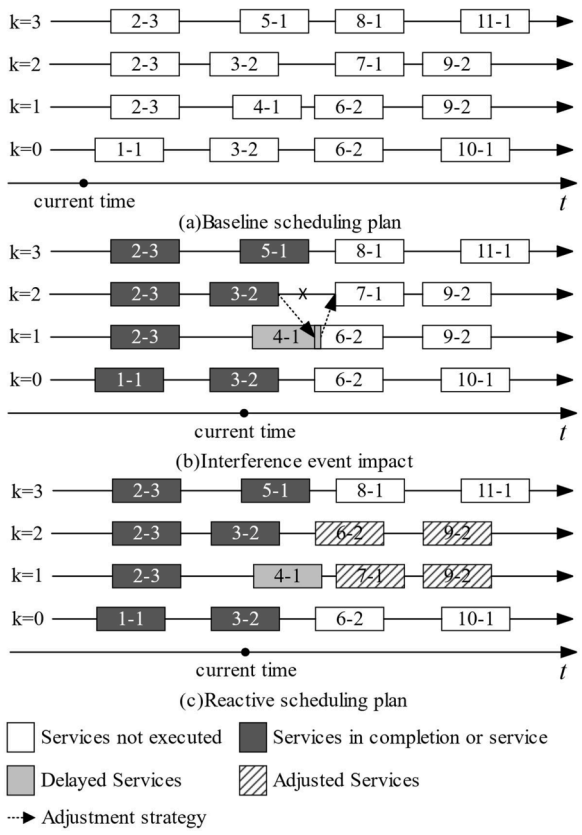

Fig. 2. Reactive scheduling diagram.

times. To simplify the problem, researchers often use minutes as the minimum time scale [8]. Time discretization allows the description of different service arrival times without being restricted to a continuous distribution function. Common approaches to finding robustness schemes use the maximizing buffer time as the optimization objective [14,15]. This approach may lead to excessive resource demand;

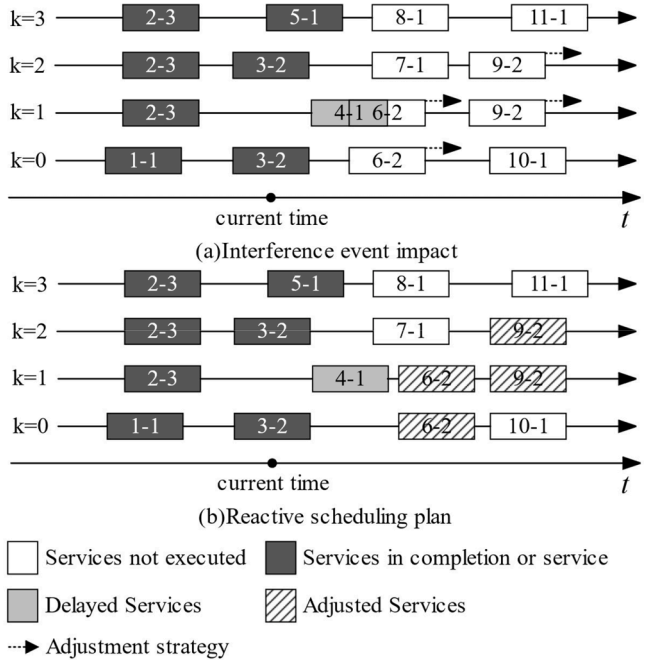

Fig. 3. Service time change in reactive scheduling.

additionally, the demand for buffer time for different services is affected by the service uncertainty pattern. In Fig. 1, the horizontal coordinates represent the time axis, which cannot be further subdivided per scale time, and the vertical coordinates represent the probability of service arrival at different moments. The same buffer time, that is, $\theta_i = \theta_j$, can be seen in Fig. 1(a), which may handle the service uncertainty to different degrees. In Fig. 1(a), θ_j handles the uncertainty of service j to a greater extent.

In this study, the continuous time is discretized with a minimum time granularity of minutes. θ_i denotes the buffer time of service i in Eq. (3) of the PSM. For service i, the earliest arrival and latest arrival time intervals are $[e_i, l_i]$, and $P(t_i)$ denotes the service i arrival time probability distribution function, with

$$\sum_{t_i \in [e_i, l_i]} P(t_i) = 1, \forall i \in N \tag{9}$$

where t_i is the service i start time random variable.

For the convenience of illustrating the subsequent content, the new notation is defined as follows: Note that η is the uncertainty coverage, which specifically refers to the service arrival time uncertainty probability in this paper. Moreover, η_i refers to the coverage of service i. a_i and b_i denote the start and end time under the coverage requirement, respectively. Different superscripts for θ_i, η_i, a_i, and b_i are used to distinguish different variables. In the discrete distribution function of the same service, the values of a_i and b_i have the following effect: the coverage rate may be different under the same buffer time. As shown in Fig. 1(b), $\theta_i = \theta_i^{'} = 3$, but $\eta_i = 0.6317 > \eta_i^{'} = 0.3683$; the buffering time may be different under the same coverage. As shown in Fig. 1(c), $\eta_i = \eta_i^{''} = 0.6317$, but $\theta_i = 3 < \theta_i^{'} = 4$. To address the above two points, we construct the buffering time and coverage relationship by requiring the minimization of the buffering time under the same coverage, as in Eq. (10). The formulas of the two relations are as follows:

$$\theta_i = \min(b_i - a_i) \tag{10}$$

$$a_i \le b_i, \forall a_i, b_i \in [e_i, l_i] \tag{11}$$

$$\sum_{a_i}^{b_i} P(t_i) \ge \eta_i, \forall a_i, b_i \in [e_i, l_i]. \tag{12}$$

Since the service arrival time probability distribution has been made discrete, the left and right parts of Eq. (12) are not often equal, so the inequality sign is used.

2.3. Reactive scheduling model

After proactive scheduling, a baseline scheduling plan that considers the uncertainty of the service arrival time is obtained. In the baseline scheduling plan, the order in which the services are executed by the resource is specified. If the services to be executed by a resource are concatenated, the chain of services scheduled to be executed by that resource is obtained. In Fig. 2(a), the straight line below represents the time axis; the black dot on the axis indicates the current time; the rectangle indicates the service block; and the values inside the rectangle indicate the service number and the number of resources required for that service. There are four service chains, corresponding to different resources. Among them, the service chains of resource $k = 1$ are 2→4→6→9. In the forward scheduling phase, if $\eta < 1$, the baseline plan cannot fully handle the impact of the service arrival time uncertainty.

As the baseline scheduling plan advances over time, the plan execution is continuously monitored to determine whether the service will incur unplanned delay costs. In Fig. 2(b), the resource $k = 1$ execution service chain has a slight delay in service 4. If the execution of service 4 is continued by $k = 1$, it will lead to disruption of the operation time of the subsequent service, 6. This disruptive impact may be offset

by subsequent buffer times, but the disruption should be eliminated as soon as possible to avoid incurring additional costs. Therefore, reactive scheduling is initiated to try to adjust the existing baseline schedule. In Fig. 2(c), the resource $k = 0$ successor service is the same as the successor service with resource $k = 1$. Finally, resources $k = 1$ and $k = 2$ are exchanged for subsequent service chains, with $k = 1$ adjusted from 6→9 to 7→9.

The reaction scheduling problem can be described as follows: the set of services (N) is divided into two parts, the set of services that have been executed or are being executed (N^-) and the set of services that have not yet been executed (N^+), using the current moment as the benchmark. In the baseline scheduling plan, the set of service chains consisting of N^+ is denoted as H, $\forall h \in H$. When service i requires multiple units of resources, service i exists in multiple service chains, and the set of service chains containing service i is defined as H_i and has $\forall i \in N^+$. At the current moment, if resource k is in the idle or transfer process state, the current moment is taken as the release time (f_k) of resource k. If resource k is in the job state, the current service schedule completion time is taken as the f_k of resource k. The necessary transfer time from the current position of resource k to the service chain (h) is denoted by $t_{h,k}$. The unexecuted part of the baseline plan is adjusted under constraints so that time constraints and state constraints are satisfied. Define 0–1 decision variables, where $y_{h,k}$ takes the value of 1 if resource k is assigned to service chain h, and 0 otherwise. The start time of service i may change during the adjustment process. In Fig. 3(a), service 4 shows a serious delay, resulting in a direct disruption of the subsequent plan for resource $k = 1$ and an indirect disruption of the subsequent plan for resource $k = 0$. Since resources $k = 2$ and $k = 3$ are in occupied status, the start of execution of service 6 can only be delayed without additional resource support, as in Fig. 3(b). Whether the subsequent service of service 6 is affected depends on whether the interval between service 6 and the subsequent service is sufficient and whether the scheme can be adjusted when the reaction scheduling is started again as time advances. Therefore, the minimization of the trailing cost remains part of the optimization objective in this phase of scheduling.

Reactive scheduling model (RSM) parameters are added as follows:

Parameters

γ: service start time coefficient;

e_i^*: the actual arrival time of service i;

RSM:

$$min \sum_{i \in N} (\alpha d_i + \gamma s_i) \tag{13}$$

$$\sum_{k \in G} y_{h,k} = 1, \forall h \in H \tag{14}$$

$$\sum_{h \in H} y_{h,k} \leq 1, \forall k \in G \tag{15}$$

$$\sum_{k \in G} (f_k + t_{h,k}) y_{h,k} \leq s_i, \forall i \in N^+, \ h \in H_i \tag{16}$$

$$e_i^* \leq s_i, \forall i \in N^+ \tag{17}$$

Eqs. (7) and (8)

Eq. (13) is the optimization objective and has two components: the first part is to minimize the service drag time, and the second part aims to start the service as early as possible after its arrival. Eq. (14) ensures that each service chain is assigned at least once in the decision cycle. Eq. (15) indicates that no more than one service chain can be assigned per unit of resource. Moreover, Eq. (16) indicates that the service start time cannot be earlier than the sum of the time of resource release and the necessary transfer time. Eq. (17) indicates that there may be some services in the reaction scheduling order that have already arrived but have not yet started, requiring that the service time cannot be smaller than the actual arrival time of the service. Meanwhile, the supplementary Eqs. (7) and (8) are calculated as the service start time limit and the delay time, requiring $\forall i \in N^+$.

Table 1
Main steps to improve the algorithm.

STEP 1: Initialize α, β, and algorithm parameters;
STE P2: Generate the initial solution (s^0) using the greedy algorithm, and record it as the current solution (s^*), and $f(s^*)$;
STEP 3: Number of iterations $I \leq IMax$; otherwise, skip to STEP 9;
STEP 4: The number of iterations in the cycle $m \leq mMax$; otherwise, update the sub-algorithm weights at each stage, $m = 0$;
STEP 5: Randomly enable a removal sub-algorithm to remove q services;
STEP 6: Randomly enable a repair sub-algorithm to repair the q services;
STEP 7: Adjudicate the probability that the new solution s will be accepted, and update s^*;
STEP 8: Update I and m, and return to STEP 2;
STEP 9: Output s^*.

Table 2
Greedy algorithm to obtain the initial solution.

STEP 1: Initialize $D = N - \{0\}, crad(D) = n - 1, crad(H) =$.
STEP 2: When $crad(D) > 0$:
2.1: Randomly select a task $i \in D$ and randomly determine s_i;
2.2: Calculate the insertion cost ($C_{i,h}$) for service i inserted into each service chain;
2.3: Select the service chain with the smallest cost (r_i) and insert the service i into the corresponding location;
STEP 3: Update $D = D - \{i\}$ and return to STEP 2;
STEP 4: Output the initial solution s^0 and its cost $f(s^0)$;

3. Algorithm design

In the ground scheduling problem, the flight arrival density and the decision cycle affect the size of the decision problem. To solve the PSM accurately and efficiently, we designed an improved ALNS algorithm. ALNS is an extended algorithm proposed by Ropke et al. [21] based on the large-neighborhood search algorithm of Shaw et al. [22] for the fetch-delivery path planning problem with time windows. The core of the algorithm is similar to the evolutionary method proposed by Li et al. [23]. This method obtains a solution that meets the output conditions through a cycle of decomposition, evolutionary destruction, random recombination, and score acceptance. The heuristic algorithm used in this article can design the destruction-repair operator according to the problem characteristics and has the characteristics of a fast solution speed and a large solution size.

The notation is defined as follows:

D: the set of services for which resources have not been scheduled;

L: the set of services for which resources have been scheduled;

$C_{i,h}$: the cost of insertion of service i into service chain h;

$f(s)$: the target fetch corresponding to the feasible solution s;

$crad(\cdot)$ the number of elements in the set $\cdot$;

The main steps of the algorithm are shown in Table 1.

3.1. Initial solution and time update mechanism

(1) Initial solution

The service start time (s_i) is constrained by the time window $[e_i, l_i]$, where the left end of the time window is a hard constraint and the right end is a soft constraint, and the part beyond the right end is defined as the delay time (d_i). When obtaining the initial feasible solution of PSM using the greedy algorithm, we mainly consider the delay cost arising from the insertion position. The main construction steps of the initial solution are shown in Table 2.

(2) Time update mechanism

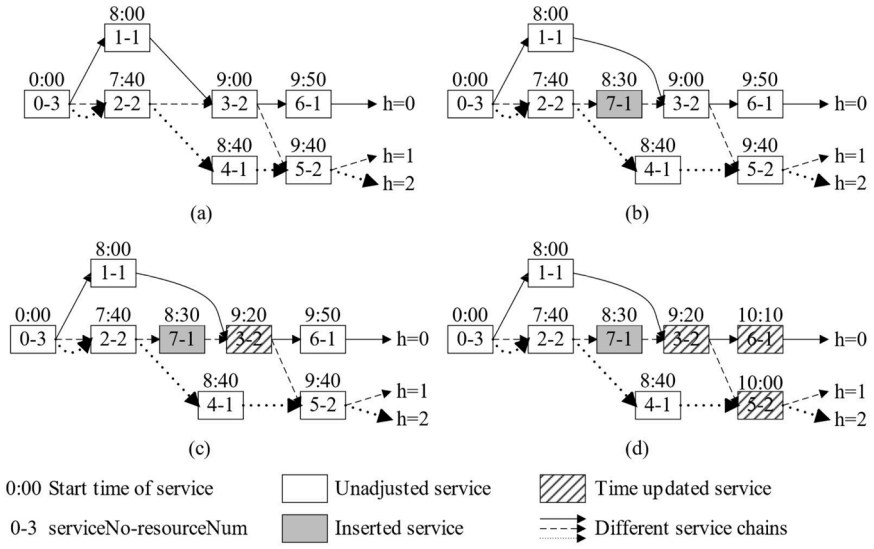

Fig. 4. Update process of the time update mechanism.

Eq. (2) in PSM can be understood as a synchronization constraint [24]. A service requires multiple units of resources to be executed simultaneously, which makes the service chains of resources interrelated. When the start time of such a service is adjusted, it will affect the other service chains involved in that service. The time of the subsequent services of these service chains may need to be adjusted, so there is some transferability of this effect. The ALNS algorithm needs to update the service time of each service node in s^* during the insertion or removal of s^*, and it needs to update the cost (C_h) of the current service chain.

To quickly adjust the change of each time node after the change of service, we introduce the "time update mechanism." When service i is inserted into the existing solution, first find the service immediately after the service, which is assumed to be the set N_i^+. Then, the service start time is updated in the middle order of the set N_i^+. As an example, in Fig. 4, the rectangle indicates the service, the data inside the graph indicate the service number and the number of resources needed, and the top of the graph indicates the start time of that service. The lines and services form a service chain, and the line direction indicates the order of service execution. In a single type of ground support resource scheduling problem, both services and resources are homogeneous. After the first stage of allocation, any service chain can be executed by any resource, but the resource can only execute at most one of the service chains. To better distinguish the relationships between services in different service chains, different line descriptions are used. Three service chains are shown in Fig. 4(a) as $h = 0$: 0→1→3→6→, $h = 1$: 0→2→3→5→, and $h = 2$: 0→2→4→5→. In Fig. 4(b), the service 7 is inserted into $h = 1$. First, the set $N_7^+ = \{3\}$ of the direct immediate after service of service 7 is obtained; then, the set N_7^+ is arranged in ascending order, and the service time is updated one by one in order, as in Fig. 4(c); then, the set $N_3^+ = \{5, 6\}$ of the direct immediate after service of service 3 is obtained, and the service time is updated in order, as in Fig. 4(d); and proceed in this manner until all of the subsequent services are updated.

3.2. Removal algorithm

The removal and repair algorithm weights of the ALNS algorithm are updated mainly using a roulette wheel approach to the selection. The repair algorithm and the removal algorithm weights are updated in the same way, but both are selected independently. Let δ_u be the weight of the u algorithm and δ' be the initial weight of all of the algorithms. During the iteration, the update of the repair and removal algorithm weights is related to the performance of the new solution (s'). There are three types of new solution performance: new solution, better than the current solution, and worse than the current solution but accepted, with scores of σ_0, σ_1, and σ_2, respectively. The weight update is performed in one cycle of *mMax* iterations, and the weights are adjusted for various removal and repair algorithms at the end of one cycle. Denote $\delta_{u,v}$ as the weight of algorithm u in cycle v; $z_{u,v}$ denotes the total score of algorithm u in cycle v; and $\tau_{u,v}$ denotes the number of times algorithm u is selected in cycle v. The weight update of algorithm u in cycle $(v+1)$ is given by

$$\delta_{u,v+1} = (1-\lambda)\delta_{u,v} + \lambda\left(\frac{z_{u,v}}{\tau_{u,v}}\right), \tag{18}$$

where λ is the impact factor indicating the extent to which the weights are affected by the new scores.

This section describes four removal algorithms: random removal, worst-path removal, Shaw removal, and worst-value removal. They share the common purpose of removing q services from the current solution and updating the time of each service in the service chain through a time update mechanism after the removal. Among them, the parameter p is introduced to increase the randomness of the Shaw removal and the worst-value removal algorithms to expand the search range and avoid falling into the local optimum.

(1) Random removal

The random removal method randomly selects q services from the current solution and then removes these services from the solution. The quality of the fix is also unstable owing to the high randomness, but the operation is fast.

(2) Worst-path removal

The worst-path removal method selects the service chain with the highest cost from existing service chains and removes all of the services from that chain. If the number of services removed is less than q, then the service chain with the highest cost is selected again for removal. This process is repeated until the total number of services removed is not less than q.

(3) Shaw removal

The Shaw removal method was proposed by Shaw in 1998 [22]. The similarity of different services is calculated by the common properties

Table 3
Shaw removal algorithm.

STEP 1: Randomly select service $i^{'}$ from the set L and move the service to the set D; that is, $D = \{i^{'}\}$;
STEP 2: Randomly select service j from the set D and compute all $S(i^{'}, j)$ with $\forall j \in L$ according to Eq. (19);
STEP 3: Sort all $S(i^{'}, j)$ in descending order and determine the number of removed services using Eq. (20);
STEP 4: Skip to STEP 2 when $crad(D) < q$;
STEP 5: Output the set D and the temporary service chain;

Table 4
Greedy repair algorithm.

STEP 1: Calculate the cost ($C_{i,h}$) incurred by inserting each service i in the set D into the temporary service chain h;
STEP 2: Arrange the costs in ascending order and obtain the minimum cost ($C^0_{i,h}$) and the corresponding insertion position (h_i);
STEP 3: Insert service i into position h_i and update set D.
STEP 4: Repeat the above steps until $D = \varnothing$;

between services, and the services with higher similarity are removed from the current service chain. Experiments have shown that removing similar services can help the subsequent service chain repair obtain a better solution, while removing less similar services yields a worse or original result.

In this paper, the similarity of services i and j, denoted as $S_{i,j}$, is influenced by three attribute ratings, namely, the relative distance of the service, the earliest start time of the service, and the service duration. The weight coefficients of the three attributes are φ_0, φ_1, and φ_2, respectively, and the specific formulas are shown in Eqs. (19) and (20) and Table 3.

$$S(i,j) = \varphi_0 t_{i,j} + \varphi_1 \left| e_i - e_j \right| + \varphi_2 \left| w_i - w_j \right| \tag{19}$$

$$q_i = y^p \times crad(L) \tag{20}$$

where y is a uniform distribution; q_i indicates the number of removed services, mainly based on the similarity of service i.

(4) Worst-value removal

The worst-value removal method removes the services with higher removal costs from the current solution. First, the cost reduction (c_i) is calculated for all of the services removed from the current solution; then, the service removal costs are ranked in descending order; next, the number of services removed is determined according to Eq. (20); finally, the set D and the temporary service chain are output, and $crad(D) \geq q$ is required.

3.3. Repair algorithm

Two types of repair algorithms exist: the greedy repair method and the regret-value repair method. The removed services are inserted into the temporary service chain to generate new solutions according to some priority rules, and then, services need to be updated on time during each insertion.

(1) Greedy repair

The specific steps of the greedy repair method are shown in Table 4.

(1) Regret-value repair

The regret-value repair method refers to the greedy repair method to calculate the cost of inserting service i. The concept of regret value is also introduced, and the calculation formula is as follows:

Table 5
Regret-value repair algorithm.

STEP 1: Calculate the cost ($C_{i,h}$) incurred by each service i in the set D inserted into the temporary service chain h.
STEP 2: Cost ascending order and obtain the minimum cost $C^0_{i,h}$ and the next smallest cost ($C^1_{i,h}$) for all of the services;
STEP 3: Calculate all of the service regret values ($C(i)$) according to Eq. (21), sorted in ascending order;
STEP 4: Obtain the service with the smallest regret value, noted as j, and the insertion position (h_j);
STEP 5: Insert service j and update the set D;
STEP 6: Repeat the above steps until $D = \varnothing$;

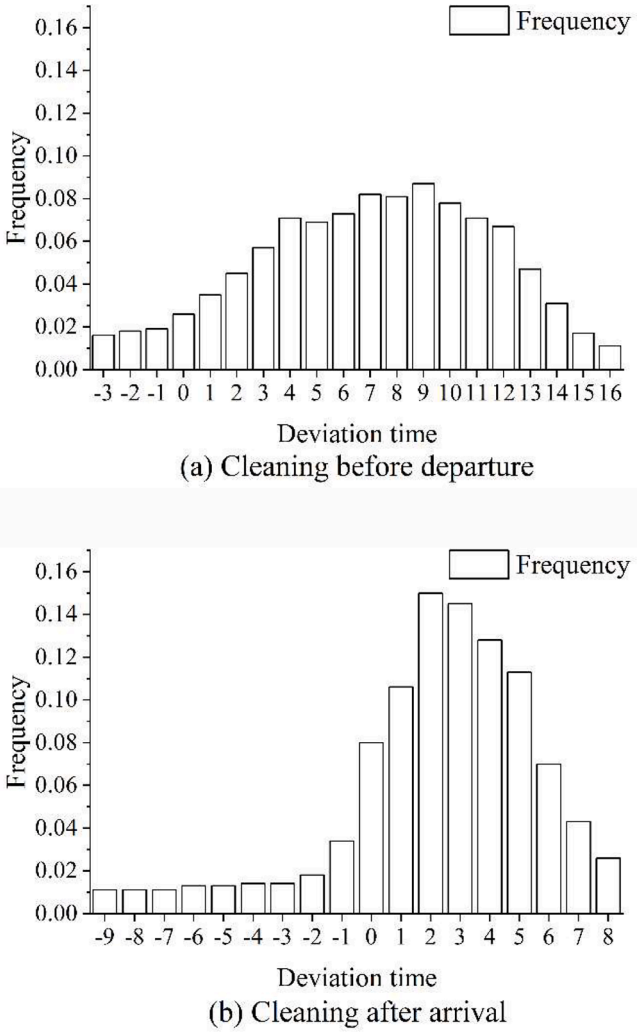

Fig. 5. Deviation of time arrival and planned arrival time.

$$C(i) = C^1_{i,h} - C^0_{i,h}. \tag{21}$$

The specific steps are shown in Table 5.

3.4. Accepting new solutions and algorithm termination

After generating $s^{'}$, it is still necessary to judge whether $s^{'}$ can be accepted or not. To keep the algorithm from falling into the local optimum, we introduce the annealing mechanism to the algorithm in this paper. The Metropolis criterion is used to calculate the probability (F) that the new solution is accepted, and the calculation is as follows:

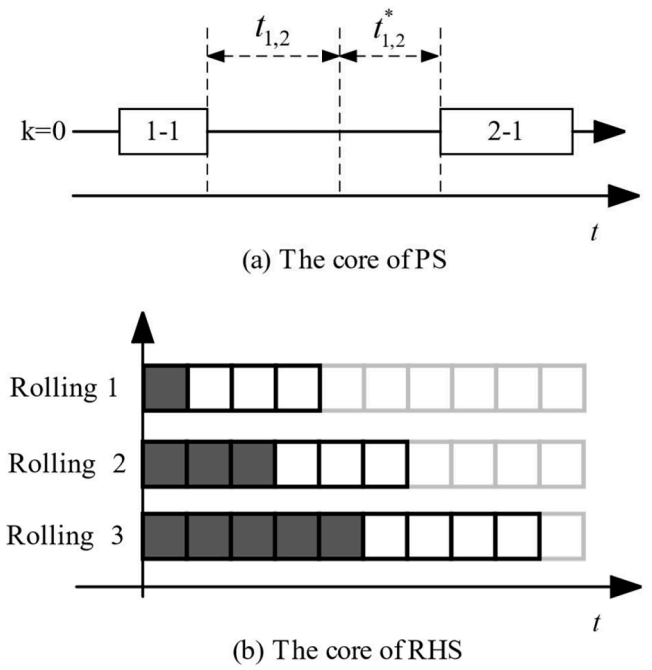

Fig. 6. The core of different scheduling methods.

$$F = \begin{cases} 1 & f(s^*) > f(s') \\ e^{\frac{f(s^*)-f(s')}{T}}, & f(s^*) \le f(s') \end{cases}, \tag{22}$$

$$T = T \times \varepsilon, \forall \varepsilon \in (0,1), \tag{23}$$

where T denotes the annealing temperature, T_0 is the initial temperature, and ε denotes the cooling ratio. Finally, this algorithm uses the maximum number of iterations as the termination criterion of the algorithm.

4. Numerical example analysis

Cabin cleaning is an important ground support service in the process of flight transit. According to different flight statuses, passenger cabin cleaning can usually be divided into three types: preflight cleaning, transit cleaning, and postflight cleaning. Among them, preflight cleaning is based on the scheduled departure time of the flight and starts and ends within a period of time before the scheduled departure of the flight. The other two types of cleaning are based on the scheduled arrival time of the flight and need to start and end within a period of time after the flight has arrived. The resource types required for these three types of cleaning are all of the cabin cleaning personnel. The quantity required and the specific duration of operations depend on the attributes of the flight, including the airline, model size, and international/domestic flights. In the cabin cleaning resource scheduling problem, owing to the non-negligible resource transfer time, the scheduling instruction needs to be issued before the service is fully determined; that is, service uncertainty is bound to exist. Fig. 5 shows a histogram of the deviation of actual arrival time and planned start time of cabin cleaning based on historical data. The horizontal coordinate is the time axis with a minimum scale of minutes, and the vertical coordinate is the frequency of the actual and planned time deviation statistics. The frequency distribution of preflight cleaning arrival time deviation is shown in Fig. 5(a), where the service arrival time uncertainty was mainly affected by the crew arrival time and tight preflight service completion. In Fig. 5(b), the frequency distribution of the deviations of transit and post-arrival time is shown. The service arrival time uncertainty was mainly affected by the flight arrival time and the completion time of the immediately preceding service. Without considering the influence of special factors,

Table 6
Parameter initialization.

Parameter	Value of parameter	Parameter	Value of parameter
α	1000	q	[5 %, 20 %]
β	1	p	3
γ	1	λ	0.3
δ'	1	$\varphi_0, \varphi_1, \varphi_2$	20, 3, 1
$\sigma_0, \sigma_1, \sigma_2$	0.4, 0.4, 0.2	T_0	100,000
IMax	10,000	ε	0.98
mMax	50		

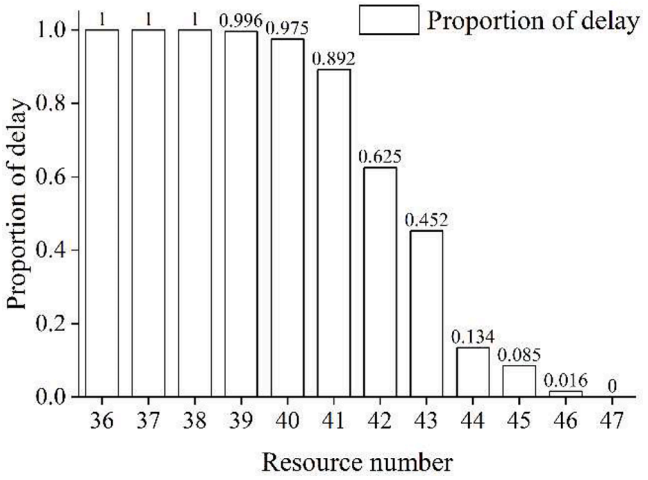

Fig. 7. Percentage of service delays with different numbers of resources, simulation times of 3000.

we attempted to approximate the discrete probability density functions of arrival times of different types of cabin cleaning services using the frequency distribution in Fig. 5.

To verify the effectiveness of the PRS decision framework considering service uncertainty coverage, we introduced proactive scheduling (PS) [14,15] and rolling horizon scheduling (RHS) [25] for comparison. Zhu et al. [14,15] deal with different types of resource scheduling problems. The optimization objectives of both are particularly similar, maximizing the idle time between services. As shown in Fig. 6(a), $t_{1,2}$ are the necessary transfer times for services $j = 1$ and $j = 2$, and $t^*_{1,2}$ is the idle time. The vertical coordinates in Fig. 6(b) indicate the number of rolls. The black square indicates the time period of the last rolled decision. The black box indicates the time period to join the current decision. Gray boxes indicate time periods that have not yet been considered. The data of flights between 12:00 and 14:00 on a certain day in the cabin cleaning operation room of an airport in South China were selected as an arithmetic example; accordingly, there were 76 cleaning services.

Combined with the parameter settings from literature [26,27], we set the model and algorithm parameters as in Table 6.

Since the number of resources has a relatively large impact on the robustness of the scheme, the number of configured resources was determined by pre-experiments. In the pre-experiments, 1000 tests were conducted on different quantities of resources using the first come first serve (FCFS) scheduling rule. In Fig. 7, the horizontal coordinate indicates the number of different resources, and the vertical coordinate indicates the number of tests with service delays. The results indicate that when the number of resources was less than 38, service delays occurred in every test; as the number of resources increased, the number of tests with service delays decreased continuously; when the number of resources reached 47, all of the service uncertainties could be handled by FCFS. Initially, if we determined the range of the resource allocation quantity {38, 44} with a step size of 2, we have $crad(G) \in \{38, 40, 42, 44\}$.

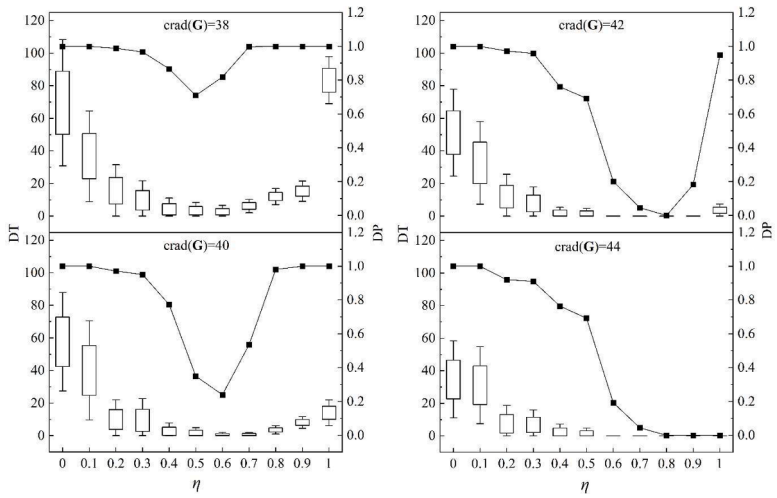

Fig. 8. Comparison of simulation results for η with different resource numbers.

4.1. PRS decision framework validity analysis

In the PS, merely considering the uncertainty coverage, the RHS constantly reschedules and updates the best scheduling scheme in the current state using time triggering or event triggering. Three metrics—delay probability (DP), average delay cost (ADC), and average transfer cost (ATC)—were used to evaluate the simulation results. The ratio between the number of simulations with service delays and the total number of simulations is DP. The ratio between the cost of all of the service delays and the total number of simulations is ADC. The ratio between the cost of all of the resource transfers and the number of simulations is ATC. After several simulations, the statistical results of the three metrics are as shown in 0.

In the pre-experiment, there was still a service delay in 13.4 % of simulations when the number of allocated resources was 44. After using the PS, RHS, and PRS, the value of the DP fell, although the optimization of the PS was not obvious. For different numbers of resources, the PRS outperformed the RHS to varying degrees.

The optimization objectives of the forward scheduling model include the costs generated by delay time DT and the transfer time (TT), which correspond to the ADC and ATC in 0 after processing. Since $\alpha \gg \beta = \gamma$, the optimization process focuses on the DT in the solution process. The combined ADC and ATC show that when the resource allocation was 38 and 40, the resources were relatively tight and PRS performed better; when the resource allocation was 42 and 44, the resources amounted to a relatively surplus and the possibility of service delays was reduced, the TT became the focus of optimization, and the RHS performed better at this time.

In summary, a more fine-grained utilization of resource free time (FT) by the PRS could obtain a scheduling scheme with better robustness when resources were relatively tight. RHS scheduling could optimize resource transfer time by rescheduling when resources amounted to a relative surplus

$\boldsymbol{\eta}$ is an indicator of the degree by which the algorithm considers uncertainty. The intuitive understanding is that when $\boldsymbol{\eta} = 0$, the scheduling decision process does not consider the problem uncertainty and that the service occupies only $\boldsymbol{w}_i$ on resources at this time. When $\boldsymbol{\eta} = 1$, the scheduling decision process completely considers the problem uncertainty, and the service occupies $\boldsymbol{w}_i + (\boldsymbol{l}_i - \boldsymbol{e}_i)$ on resources at this time. Fig. 8 shows how the DP and DT were affected by the value of $\boldsymbol{\eta}$ for four different resource configurations. The effect of $\boldsymbol{\eta}$ taking values on the DP and DT influenced by a different number of resource configurations varied widely. The basic pattern was first decreasing and then increasing. $\boldsymbol{\eta}$ was described as $\boldsymbol{\eta}^*$ when DP decreased to the lowest point. When $\boldsymbol{crad}(\boldsymbol{G}) = 38$, $\boldsymbol{\eta}^* \cong 0.5$; when $\boldsymbol{crad}(\boldsymbol{G}) = 42$, $\boldsymbol{\eta}^* \cong 0.8$. Thus, $\boldsymbol{\eta}^*$ took different values for different resource allocation quantities.

In the airport ground resource scheduling problem, the resource shift time could be divided into three parts: operation time, TT, and FT. η essentially finely utilized the FT in response to service uncertainty. When the number of resources was relatively tight, the FT ratio was low and η was adjusted, which had a small impact on DP and DT; see $crad(D) = 38$ in Fig. 8. As the number of resources increased, the FT ratio gradually increased. When $\eta \leq \eta^*$, increasing η could effectively use FT to handle problem uncertainty. When $\eta \geq \eta^*$, increasing η caused services to over-occupy resources and resources to become relatively tight; see $crad(D) = 40$ and $crad(D) = 42$ in Fig. 8. Moreover, when the number of resources amounted to a relatively surplus, the FT ratio was high enough to satisfy $\eta = 1$ (see $crad(D) = 44$ in Fig. 8) when $\eta^* \cong 1$.

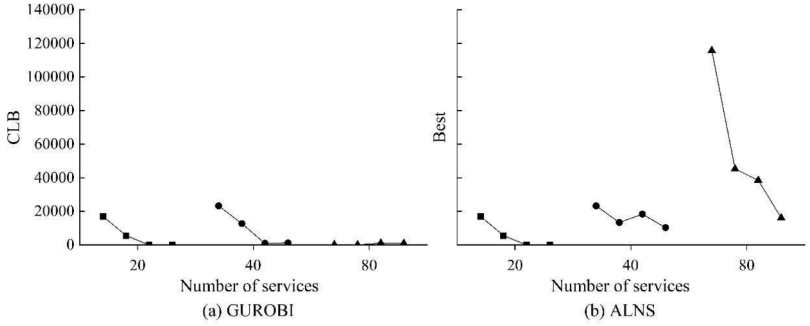

Fig. 9. Comparison of boundary solutions using different algorithms.

Table 7
Simulation of experimental results of different strategies.

$crad(G)$	η	DP			ADC			ATC		
		RHS	PS	PRS	RHS	PS	PRS	RHS	PS	PRS
38	0.50	0.81	0.99	0.71	5364.02	9145.96	4025.52	173.95	214.54	182.78
40	0.60	0.50	0.87	0.27	1909.23	3020.94	1043.76	153.03	255.04	165.31
42	0.80	0.02	0.44	0.00	0.00	961.13	0.00	130.52	216.45	137.75
44	0.90	0.00	0.02	0.00	0.00	447.27	0.00	116.36	218.11	122.16

Table 8
Results of Gurobi and ALNS under different-scale problems.

$crad(N-\{0\})$	$crad(G)$	Gurobi			ALNS		
		CS	CLB	Time (s)	Best	AVG	Time (s)
20	23	**17,047.1**	17,047.1	36.4	**17,047.1**	17,047.1	3.4
	25	**5583.2**	5583.2	2.6	**5583.2**	5583.2	2.7
	38	**0.3**	0.3	2	**0.3**	0.3	19.2
	40	**0.0**	0.0	2.1	**0.0**	0.0	20.4
40	23	**23,254.6**	23,254.6	740.6	**23,254.6**	23,257.7	62.6
	25	**12,744.5**	12,743.3	1568.6	13,343.8	16,119.5	59.4
	38	19,750.9	1016.9	3600.0	**18,340.6**	18,469.4	3.8
	40	**10,340.2**	1269.5	3600.0	**10,340.2**	10,584.9	2.6
80	23	–	234.3	3600.0	**115,759.6**	121,608.6	18.8
	25	–	199.2	3600.0	**45,331.1**	48,169.3	17.7
	38	–	1134	3600.0	**38,531.2**	46,641.3	107.4
	40	–	1044.4	3600.0	**16,038.0**	18,671.9	123.4

4.2. Algorithm effectiveness analysis

The problem size in the PRS decision framework is often largely due to the range of decision periods. For this reason, the improved ALNS algorithm was designed in this paper to handle the problem. The Gurobi solver and the improved ALNS algorithm were introduced for comparison. To verify the effectiveness of the algorithm designed in this paper, we designed two-factor multilevel experimental arithmetic cases. Among them, we chose the service quantity factor $crad(N-\{0\}) \in \{20, 40, 80\}$ and the resource quantity factor $crad(G) \in \{23, 25, 38, 40\}$. The optimization objective of the proactive stage was used as the main evaluation index of the two solution methods. The test results of all of the cases are shown in Table 8. The results of the Gurobi solver had three parts: current solution (OS), current lower bound (CLB), and solution time. The results of the improved ALNS algorithm had three parts: best solution, average value, and solution time. Since the model optimization goal is to minimize the cost, it is better to obtain a smaller cost algorithm. During the experiment, the upper limit of Gurobi solution time was set to 3600 s, so there were two conditions for Gurobi to terminate: first, the optimal solution was found; second, the upper limit of time was reached and the current optimal solution was output. As shown in Table 8, when we were solving the small-scale problem, both algorithms could find the optimal solution in a short time; when solving the medium-scale problem, we found that the two algorithms did not produce considerably different results but that the improved ALNS took less time than Gurobi. When the number of services increased to 80, Gurobi could not find a feasible solution in the limited time.

In Fig. 9, the trend plots of the CLB and Best for the two solution methods are shown for different combinations of experiments. With the same number of services, it is theoretically possible to reduce the target fetch as the resources increase (when resource allocation cost is not considered). This corollary is somewhat verified in Fig. 9(b), but it is difficult to see this in Fig. 9(a). The CLB was constantly proved and updated because the solver enabled both the approximate and exact algorithms, with the approximate algorithm focusing on the update of the solution and the exact algorithm focusing on the proof of the solution. However, in large-scale problems, the CLB was proved and updated at a reduced rate. As shown in Fig. 9(a), when the number of services was 40 and the number of resources exceeded 38, the CLB might not yet be a true lower bound on the feasible solution.

Gurobi could find feasible solutions of higher quality for smaller scale problems and show the gap between the current optimal solution and the theoretical optimal solution, but it was limited by the size of the solution; ALNS could obtain better performance at different scales, especially when the problem size was large (Table 7).

5. Conclusion

In this paper, forward and reactive scheduling strategies were introduced to address the arrival time uncertainty of airport ground protection services. In the proactive scheduling stage, the uncertainty coverage concept was introduced, considering the limitation of traditional buffer time. The improved ALNS algorithm was designed to solve the foreground scheduling model effectively and quickly. The actual effect of the strategy in an uncertain environment was simulated and tested using actual data from South China International Airport as an experimental arithmetic example. The test results indicate the following: (1) the PRS can better utilize the idle time of resources when resources are relatively tight and develop a benchmark plan with better robustness. (2) Using η test results can reflect the rationality of resource quantity allocation. (3) The improved ALNS algorithm is more efficient to solve compared with Gurobi, especially among large-scale problems. Therefore, the forward and reactive strategies can better handle the uncertainty of airport ground protection services and at the same time guide the allocation and utilization of airport ground protection resources.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] Zhao PX, Gao WQ, Han X, Luo WH. Bi-objective collaborative scheduling optimization of airport ferry vehicle and tractor. Int J Simul Model 2019;18(2):355–65.

[2] Ip WH, Wang D, Cho V. Aircraft ground service scheduling problems and their genetic algorithm with hybrid assignment and sequence encoding scheme. IEEE Syst J 2013;7(2):649–57.

[3] Chen B, Tang K, Yang Y. Airport resource scheduling optimization based on planning time window. In: Proceedings of the CCDC; 2021. p. 3971–8.

[4] Padrón S, Guimarans D, Ramos JJ. A bi-objective approach for scheduling ground-handling vehicles in airports. Comput Oper Res 2016;71(7):34–53.

[5] Zhao PX, Han X, Wan D. Evaluation of the airport ferry vehicle scheduling based on network maximum flow model. Omega 2021;99.

[6] Xing Z, Yi L. Research of algorithms for aircraft ground deicing operation scheduling model. In: Proceedings of the WCICA; 2010. p. 5973–7.

[7] Schmidt M, Paul A, Cole M. Challenges for ground operations arising from aircraft concepts using alternative energy. J Air Transp Manag 2016;56(pt. B):107–17.

[8] Ng KKH, Lee CKM. Review on meta-heuristics approaches for airside operation research. Appl Soft Comput 2018.

[9] Schmidt M. A review of aircraft turnaround operations and simulations. Prog Aerosp Sci 2017;92(7):25–38.

[10] Alexander EI, Brownlee A. A fuzzy approach to addressing uncertainty in airport ground Movement optimization. – ScienceDirect Transp Res Part C Emerg Technol 2018;92(2018):150–75.

[11] Liang J, Wu J, Gao Z, Sun H, Yang X, Lo HK. Bus transit network design with uncertainties on the basis of a metro network: a two-step model framework. Transp Res Part B Methodol 2019;126.

[12] Diepen G, Pieters BFI. Robust planning of airport platform buses. Comput Oper Res 2013;40(3):747–57.

[13] Padrón S, Guimarans D. A stochastic approach for planning airport ground support resources. Int Trans Oper Res 2022;29(6):3316–45.

[14] Zhu S, Sun H, Guo X. Cooperative scheduling optimization for ground-handling vehicles by considering flights' uncertainty. Comput Ind Eng 2022;169. https://doi.org/10.1016/j.cie.2022.108092.

[15] Gök YS, Guimarans D, Stuckey PJ, Tomasella M, Ozturk C. Robust resource planning for aircraft ground operations. In: Proceedings of the CPAIOR 2020; 2020. p. 222–38.

[16] Xiang X, Liu C, Miao L. Reactive strategy for discrete berth allocation and quay crane assignment problems under uncertainty. Comput Ind Eng 2018;126(12):196–216.

[17] Cui W, Lu Z, Li C. A proactive approach to solve integrated production scheduling and maintenance planning problem in flow shops. Comput Ind Eng 2017;115(1):342–53.

[18] Su X, Han W, Wu Y. A proactive robust scheduling method for aircraft carrier flight deck operations with stochastic durations. Complexity 2018;2018.

[19] Kang L, Meng Q, Tan KC. Tugboat scheduling under ship arrival and tugging process time uncertainty. Transp Res Part E Logist Transp Rev 2020;144.

[20] Tan C, He J. Integrated proactive and reactive strategies for sustainable berth allocation and quay crane assignment under uncertainty. Ann Oper Res 2021.

[21] Ropke S, Pisinger D. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp Sci 2006;40.

[22] P. Shaw, "Using constraint programming and local search methods to solve vehicle routing problems," 1998.

[23] Li JP, Bai RB, Shen YD, Qu R. Search with evolutionary ruin and stochastic rebuild: a theoretic framework and a case study on exam timetabling. Eur J Oper Res 2015;242(3):798–806. https://doi.org/10.1016/j.ejor.2014.11.002.

[24] Cai Y, Zhang Z, Guo S. A tree-based tabu search algorithm for the manpower allocation problem with time windows and job-teaming constraints. Atmos Environ 2013;77:9–15.

[25] Stolletz R, Zamorano E. A rolling planning horizon heuristic for scheduling agents with different qualifications. Transp Res Part E Logist Transp Rev 2014;68(8):39–52.

[26] Pisinger D, Ropke S. A general heuristic for vehicle routing problems. Comput Oper Res 2007;34(8):2403–35.

[27] Azi N, Gendreau M, Potvin JY. An adaptive large neighborhood search for a vehicle routing problem with multiple routes. Comput Oper Res 2014;41(1):167–73.