Hindawi
Computational Intelligence and Neuroscience
Volume 2022, Article ID 1843675, 12 pages
https://doi.org/10.1155/2022/1843675

Research Article

An Improved Simulated Annealing-Based Decision Model for the Hybrid Flow Shop Scheduling of Aviation Ordnance Handling

Xianglei Meng, Nengjian Wang, Jue Liu, and Qinhui Liu

College of Mechanical and Electrical Engineering, Harbin Engineering University, Harbi 150001, China

Correspondence should be addressed to Qinhui Liu; liuqinhui@hrbeu.edu.cn

Received 20 December 2021; Revised 12 January 2022; Accepted 15 January 2022; Published 2 February 2022

Academic Editor: Daqing Gong

Aviation ordnance handling is critical to the firepower projection of the time-critical cyclic flight operation on aircraft carriers. The complexity of the problem depends on the supply and demand features of ordnance. This paper examines the scheduling of aviation ordnance handling of an operational aircraft carrier under the framework of hybrid flow shop scheduling (HFS) and derives a method based on the simulated annealing (SA) algorithm to get the HFS problem's solution. The proposed method achieves the minimum possible flow time by optimizing the ordnance assignment through different stages. The traditional SA algorithm depends heavily on the heuristic scheme and consumes too much time to compute the optimal solution. To solve the problem, this paper improves the SA by embedding a task-based encoding method and a matrix perturbation method. The improved SA remains independent of the heuristic scheme and effectively propagates the local search process. Since the performance of SA is also influenced by its embedded parameters, orthogonal tests were carried out to carefully compare and select these parameters. Finally, different ordnance loading plans were simulated to reveal the advantage of the improved SA. The simulation results show that the improved SA (ISA) can generate better and faster solution than the traditional SA. This research provides a practical solution to stochastic HFS problems.

1. Introduction

This study focuses on the ordnance dispatching scheduling problem observed onboard the aircraft carrier flight operation, which plays an important role in the air wing firepower projection in its sortie generation [1]. The ordnance handling process involves many stages, equipment, and hundreds of personnel operating in a limited work space [2]; finding an optimal dispatch scheduling for a given ordnance load plan, plus the time critical nature of cyclic flight window requires, is a challenging problem. Traditionally, ordnance dispatching scheduling is made by a human operator's hand in a spreadsheet with experience, which is always nonoptimal, or even leading to delays that left aircraft launching without firepower. Thus, robust optimal scheduling is essential to conduct the ordnance handling procedure. However, such problem has seldomly been studied, which can be casted in the hybrid flow shop (HFS) scheduling framework.

The HFS scheduling problem [3] can be regarded as the combination of the flow shop scheduling (FSS) [4] problem and parallel machine scheduling (PMS) problem, where the former is to decide the job sequences through the shop and the latter is to allocate jobs to machines, given the processing times of each job on each machine according to one or several given criteria, aiming to minimize the makespan [5–7]. For an n jobs m stages problem, there are a total of $(n!)^m$ possible schedules, which proves to be NP-hard [8]. If the numbers n of jobs and m of stages are very small, the optimal schedule may be determined by exhaustion, such as branch and bound (B&B) [9] or integer programming techniques [10]. However, these approaches are not applicable to HFS problems with numerous jobs and stages, due to their enormous computing time and memory occupation.

Thus, for scheduling different HFS configurations, a large number of approximation and heuristics algorithms have been proposed [11, 12]. The computational complexity of HFS propelled scholars to develop many heuristics algorithms to obtain good enough solutions in a short time for medium-to-large problems, such as different scheduling rules [13], but the heuristic methods are too problem-

specific, it often cannot be applied to generalized problem. For the past decades, many general schemes on improving the performance of simple heuristics have been successfully developed, most of which are named as metaheuristics, such as genetic algorithm (GA) [14], ant colony optimization (ACO) [15], tabu search (TS) [16], neural networks (NN) [17], artificial immune systems (AIS) [18], and simulated annealing (SA) [19]. They inhere with higher level of abilities in searching the vast solution space, which have better performance than the simple heuristic methods.

Since different heuristics work effectively for different problems, when it encounters the flow shop scheduling problem, Maaroju [20] tested all the metaheuristic methods and found that the genetic algorithm and simulated annealing outperformed others, for hill climbing, swarm intelligence, and neural networks yielded only marginal improvements. However, the computation time for the GA is larger than that for SA. Thus, the SA-based algorithm is chosen in dealing with the ordnance handling problem under the HFS framework. Simulated annealing origins from the metallurgy technology, where a material cools down from high temperature to get minimum energy state. In the algorithm, the current state s and neighbor states s' are considered, and the algorithm decides the state transition probability from s to s' based on current system energy (known as temperature). This process continues until a good enough state has been found or the computation threshold has been reached. Such mechanism guarantees approximating to global optimum without getting stuck in local minimum for solving large complex optimization problems. However, the traditional SA algorithm has several defects [21], which include heuristic-dependent, parameter-specific, and long computation time; thus, the performance of the algorithm is yet to be improved. To overcome the above defects, this paper presents a Monte Carlo [22] perturbation method, which directly perturbs the solution matrix in each iteration of SA cooling, eliminating the dependence on any heuristic method, whereas SA performance also depends on cooling parameters; this paper carefully plans the calibration of these parameters to accelerate the computation process by adding double thresholds and setting the memory method of the SA.

The organizations of this paper are as follows: Section 2 introduces the ordnance handling process in detail; Section 3 discusses the methodology of the SA and improves the SA by embedding a new decoding method and matrix perturbation method; Section 4 evaluates the improved SA algorithm through computational experiments; Section 5 summarizes the research findings and gives the directions of future research.

2. Ordnance Handling Procedure

The ordnance handling procedure is specified in a daily loading plan, which lists the amount and types of weapons (throughout this paper, ordnance and weapon are used interchangeably) to be loaded onto the corresponding aircraft. Figure 1 shows the layout of aircraft carrier decks, where the construction and transfer of ordnance origin from the magazines located in lower decks to the awaiting aircraft

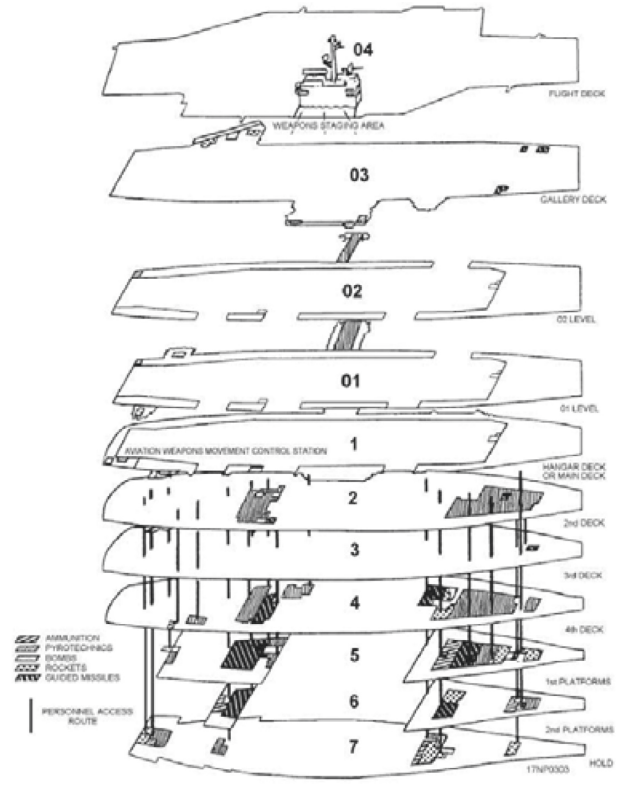


Figure 1: The layout of the ordnance handling routes aboard an aircraft carrier.

on flight deck following a series of stages. In stage I, the ordnances are retrieved from magazines by bomb skids and delivered to lower-stage elevators. In stage II, the ordnances are lifted to the hangar deck by lower-stage elevators. In stage III, the ordnances are transferred to the staging area of the hangar deck, assembled in that area, and moved to upper-stage elevators. In stage IV, the ordnances are transferred to the flight deck by upper-stage elevators. In stage V, the ordnances are moved directly to and loaded on the aircraft waiting on the flight deck. The flowchart of this procedure is shown in Figure 2. For a common aircraft carrier layout (Figure 1), there are at least 4 magazines in the delivering stage, 8 elevators in the lower-lifting stage, 2 assemble centers in the assembling stage, 4 elevators in the upper-lifting stage, and around 10 aircraft spots in the loading stage. According to Gupta [8], the two-stage flow shop problem with one stage containing a single machine can be NP-hard. Thus, the ordnance handling problem is far from trivial, especially for ordnance officers in making timely decisions of the flight operations.

This paper examines the ordnance handling problem in the HFS framework. The definition of hybrid flow shop system is as follows: in a factory, the set of n jobs $J = \{1, 2, \ldots, j, \ldots, n\}$ is going to be processed through m stages $M = \{1, 2, \ldots, i, \ldots, m\}$ in sequence, while each stage i contains $M_i = \{1, 2, \ldots, k, \ldots, m_i\}$ identical machines, and the processing time of job j on machine k is $p_{jk} \geq 0$. The objective is always to minimize makespan. Similarly, in the ordnance handling problem, n batches of weapons are

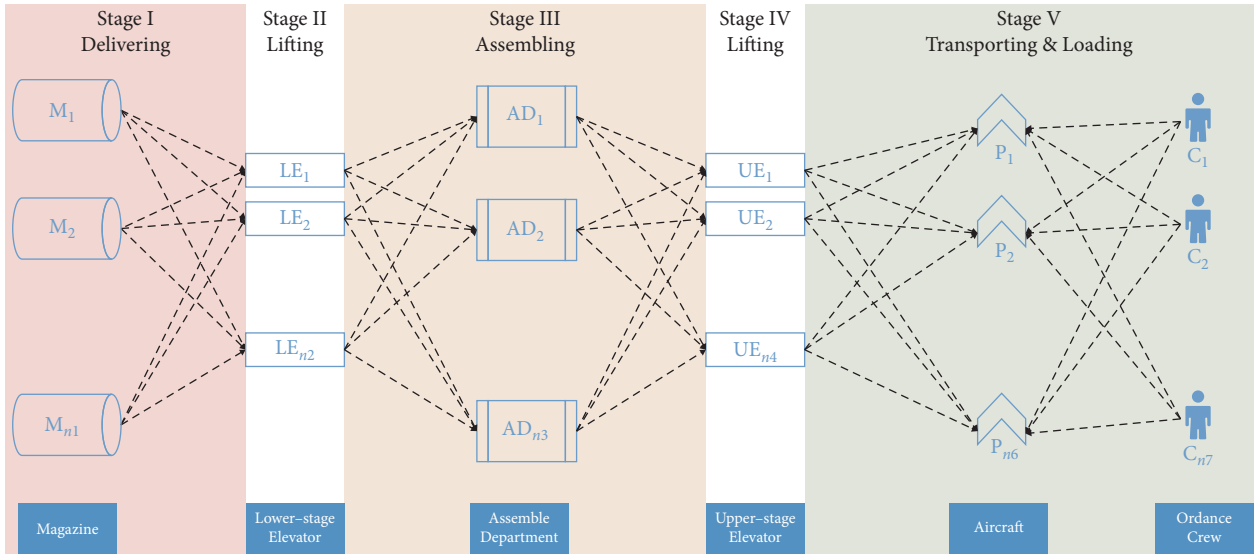


FIGURE 2: Flowchart of ordnance handling process. Note: M = magazine; LE = lower-stage elevator; AD = assembly department; UE = upper-stage elevator; P = aircraft parking spot.

considered as jobs; they also shall be processed in the same order through m stages by facility k (as shown in Figure 2) with processing time p_{jk}, and the objective is to decide the weapons' sequences and the allocations of weapons to facilities to get the minimum flow time. This is a combinatorial optimization problem with $(n!)^m$ possible schedules, which is considered as NP-hard so that it is difficult to find the optimal solution in polynomial time. For a simple case of 10 batches of weapons in our problem, there can be $(10!)^5 = 6.3 \times 10^{32}$ different schedules for the ordnance officer to choose, which is beyond human mind's reach in conducting the time-critical flight operations.

The ordnance configuration of aircraft on the carrier depends on the specific mission [23]. It is assumed that each transfer equipment or personnel only transfers one type of ordnance at a time [24]. For each type of ordnance, the number loaded in one skid is denoted as r_w Thus, the batches of ordnances needed to complete the task of all aircraft can be determined by

$$\text{task} = \left\{ \begin{array}{llllll} 1 & (1, \text{num}_1) & (2, \text{num}_2) & \ldots & (w, \text{num}_w) & \\ 2 & (1, \text{num}_1) & (2, \text{num}_2) & \ldots & (w, \text{num}_w) & \ldots \quad (W, \text{num}_W) \\ & & & \ldots & & \\ a & (1, \text{num}_1) & (2, \text{num}_2) & \ldots & (w, \text{num}_w) & \ldots \quad (W, \text{num}_W) \end{array} \right\}, \quad w = 1, 2, \ldots W, \tag{1}$$

where a is the number of aircraft to be loaded, w is the type of ordnance, and num_w is the actual number of ordnance types w.

$$\xi = \sum_{i=1}^{a} \sum_{j=1}^{w} \frac{\text{num}_w}{r_w}, \quad i = 1, 2, \ldots a, \; j = 1, 2, \ldots w, \tag{2}$$

where r_w is the number of type w ordnances in one skid, a is the number of aircraft, and w is the number of ordnance types to be loaded on the aircraft. That is, an ordinance should be transferred to the required aircraft, once being retrieved from the magazine.

2.1. Stage I: Weapons Retrieving. Multiple magazines are located in the bow and aft of the carrier, and the ordnances can be transferred by multiple elevators. The ordnances are firstly retrieved from magazines by skids with the setup time T_0. From the same magazine, the ordnances should be retrieved with an interval no shorter than t_{int}. Then, the skids deliver the ordnances to lower-stage elevators. The time consumed to transfer ordnances from magazines to lower-stage elevators can be expressed as

$$T_M^L = \begin{bmatrix} t_1^1 & t_1^2 & \cdots & t_1^L \\ t_2^1 & t_2^2 & \cdots & t_2^L \\ \vdots & \vdots & \vdots & \vdots \\ t_M^1 & t_M^2 & \cdots & t_M^L \end{bmatrix}, \tag{3}$$

where M is a magazine and L is a lower-stage elevator.

2.2. Stage II: Weapons Buildup. The ordnances are loaded onto lower-stage elevators and lifted vertically with constant speed to the hangar deck. The time consumed in this stage (lifting time) is denoted as T_L.

2.3. Stage III: Weapons Assembling. The ordnances are preassembled in the staging area of the hangar deck, with a sufficient lead time to meet the short turnaround time of the flight schedule. The assembling time T_K^{ass} varies with the types of ordnances. Note that the assembling time of the staff fluctuates in the real world. Therefore, the interval of assembling time was set to $[-T_{f1}, T_{f1}]$. The real assembling time is denoted as $T_K^{ass} + t_1$, where t_1 is a random number within $[-T_{f1}, T_{f1}]$.

2.4. Stage IV: Weapons Striking Up. The ordnances are transferred to the flight deck by upper-stage elevators. The time consumed in this stage (transport time) is denoted as T_U.

2.5. Stage V: Weapons Loading. Some ordnance crew members on the flight deck transport the ordnances from the upper-stage elevators to the aircraft. The time consumed in this stage can be expressed as

$$T_U^A = \begin{bmatrix} t_1^1 & t_1^2 & \cdots & t_1^A \\ t_2^1 & t_2^2 & \cdots & t_2^A \\ \vdots & \vdots & \vdots & \vdots \\ t_U^1 & t_U^2 & \cdots & t_U^A \end{bmatrix}, \tag{4}$$

where U is an upper-stage elevator and A is an aircraft.

The other ordnance crew members load the ordnances onto the aircraft. It is assumed that the different groups consume the same time to load the same ordnance and different types of ordnances need different time to be loaded. The time needed to load each type of ordnance is denoted as T_K^{load}. The interval of the loading time was set as $[-T_{f2}, T_{f2}]$. The real loading time is denoted as $T_K^{\text{load}} + t_2$, where t_2 is a random number within $[-T_{f2}, T_{f2}]$.

The ordnance crew members can be shared across groups. The loading cannot proceed unless ordnance crew members are available. The walking time for interstation transfer between different aircraft can be expressed as

$$T_A^A = \begin{bmatrix} t_1^1 & t_1^2 & \cdots & t_1^A \\ t_2^1 & t_2^2 & \cdots & t_2^A \\ \vdots & \vdots & \vdots & \vdots \\ t_A^1 & t_A^2 & \cdots & t_A^A \end{bmatrix}. \tag{5}$$

Referring to the standard three-field notation for scheduling problems, our problem can be described as follows: $FH5$, $(PM^{(k)})_k^2 = 1$, $(RM^{(k)})_{k=3}^4 | prmu, M_j^{(5)}, \text{block} | C_{\max}$. Specifically, $FH5$ is a five-stage HFS problem: stage I involves $M^{(1)}$ identical magazines that store ordnances; stage II involves $M^{(2)}$ lower-stage elevators to transport the ordnances; stage III has $M^{(3)}$ identical assembling personnel to assemble the ordnances; stage IV has $RM^{(4)}$ independent upper-stage elevators to transport the ordnances; and stage V has $RM^{(5)}$ independent aircraft to be loaded. Note that *prmu* indicates that the ordnances are handled in the same order in every stage; $M_j^{(5)}$ (eligibility constraint) means that the handling of ordnance j is limited to the aircraft set M in stage V; *block* indicates that the capacity of buffer between stages is constrained, for instance, the weapons have to wait in the current stage till enough room is released for the next stage of handling.

In total, the completion time for batch i of ordnances can be calculated by

$$C_i = T_M^L + T_L + T_K^{ass} + t_1 + T_U + T_U^A + T_A^A + T_K^{\text{load}} + t_2 + T_{\text{wating}}, \quad i = 1, 2, \ldots \xi, \tag{6}$$

where T_{wating} is the whole waiting time in the transporting process, for an ordinance cannot be handled unless machines or ordnance crew members are available.

The general objective of ordnance handling is to complete all transporting operations as efficiently as possible within the specified time and to generate a reasonable schedule for ordnance handling. Therefore, for our ordnance handling problem, the minimization of makespan is set as goal, so that sufficient ordnances can be loaded to the awaiting aircraft to fly in the next flying window.

$$\text{Object} = \min\left(C_{\max}\right). \tag{7}$$

3. Improved Simulated Annealing Algorithm

The SA is a technique capable of searching for good solutions to various combinatorial problems in material science and physics. The pseudocode of the algorithm is as follows.

The SA includes three major functions: state generation, state acceptance, and temperature update. The first is to make perturbations of the given initial solution, in order to search for the optimal solution effectively in the vast solution space. The second decides whether to accept a newly generated solution with a certain probability in case of trapping in the local minimum. The third offers a cooling scheme that mimics the physical annealing process to get a stable state for the problem. The SA performance can be augmented by adjusting various parameters and operators [25], such as initial temperature, descent gradient of temperature, and termination rule, which have to be adjusted manually. This paper mainly improves the search efficiency (timeliness) of the SA, without sacrificing the optimization quality. Thus, encoding scheme, initial solution, neighborhood search structure (NSS), type of cooling schedule, and two criteria (internal cycle termination and external cycle termination) are modified as described in the following subsections.

3.1. Task Encoding. This paper presents a task-based encoding method for the ordnance handling problem, where ordnances are coupled with aircraft. Each job of the HFS is defined as an operation o^{xy}, where x is the type of ordnance and y is the aircraft to be loaded. Then, a solution can be expressed as

Input: A initial S solution and a cost value function $F(x)$.
Output: A S' solution that minimizes the cost value function $F(x)$.
$T \leftarrow$ initializing Temperature//a method for assigning an initial temperature
while $T > B$ not freezing **do**//a definition of "frozen,"
  for $i = 1$ to C do while not at equilibrium do//a definition of "equilibrium,"
    S'$\leftarrow$ new permutation of S.
    If $F(S') < F(S)$ or Random value $< e^{(F(S')-F(S))/DT}$ then $S \leftarrow S$'//a selection criterion
  end
    $T \leftarrow$reduced temperature//a way of calculating the next temperature
end

Algorithm 1: Standard simulated annealing.

$$S = \begin{pmatrix} o_1^{xy} & m_{11} \cdots & m_{n1} \\ o_2^{xy} & m_{21} \cdots & m_{n1} \\ \vdots & \vdots & \vdots \\ o_i^{xy} & m_{n1} \cdots & m_{nn} \end{pmatrix}, \tag{8}$$

where the first column is a permutation of the task sequence, the following columns are the corresponding stages, and m_1 to m_n are machines assigned randomly to execute the tasks.

3.2. Initial Solution. As mentioned above, the initial solution has a great impact on the final solution of the SA. According to the heuristic methods mentioned in [26], we generate the initial solution by the following rule: first, the permutation of tasks is determined by assigning each type of weapon x to its corresponding aircraft y, defined as operation o_i^{xy}, according to the ordnance loading plan P. Then, the tasks are assigned to the earliest available machine. If there are more than one earliest available machines, one of them will be chosen randomly. The initial solution will be generated as $S_{i\times m}$ with i tasks through m stages.

For example, Table 1 shows the ordnance loading plan of the aircraft waiting to operate in the next fly window. There are three aircraft and four types of ordnances. Aircraft 1 requires 2 skids of type 1 ordnances and 1 skid of type 2 ordnances.

Here, the operations are permutated by aircraft number (equal priority) in ascending order: $o_i^{xy} = \{o_1^{11}, o_2^{11}, o_3^{21}, o_4^{12}, o_5^{32}, o_6^{43}\}$. First, a random permutation of jobs (tasks) is generated. Then, the optimal available elevator is assigned to each operation. The final plan can be written as

$$\begin{pmatrix} 3 & 1 & 1 & 1 & 1 & 1 \\ 5 & 2 & 2 & 2 & 2 & 2 \\ 1 & 3 & 3 & 1 & 3 & 3 \\ 4 & 4 & 4 & 2 & 4 & 1 \\ 2 & 1 & 1 & 1 & 1 & 2 \\ 6 & 2 & 2 & 2 & 2 & 3 \end{pmatrix}. \tag{9}$$

The meaning of such matrix can be explained as follows: taken the third row for illustration, operation o_1^{11} is extracted out of the third magazine, transferred by the third lower-stage elevator to the first staging area, moved by the third upper-stage elevator to the flight deck, and loaded to the corresponding aircraft by the No. 3 ordnance crew.

3.3. Neighborhood Perturbation. The NSS can generate a new solution by slightly modifying the current candidate solution. Traditionally, many different NSSs are adopted in each iteration of SA computation, such as swap, shift, and reversion [27], to realize the task permutation in the first stage. Then, the same heuristic rules are applied in the following stages to generate a new solution S'. The quality of the solution depends on the selected heuristic rules. If the rules are too greedy, the algorithm may fall into the local optimum, being unable to get convergence to the final optimal result.

The quality of the SA solution is highly sensitive to the selection of candidate solutions. Therefore, the perturbation scheme is crucial to the good performance of the SA algorithm. To ease the dependence of neighborhood search on heuristics, this paper proposes a Monte Carlo perturbation technique, which directly perturbs the initial solution matrix. The initial solution is changed significantly in one step, eliminating the effect by heuristic methods. The matrix perturbation is described as follows.

The following is an example of the matrix perturbation process: for instance, a 6×3 matrix, a rectangle $R_{a\times b}$ randomly generated a size of 4×3 matrix.

$\begin{pmatrix} 3 & 1 & 2 \\ 4 & 2 & 1 \\ 6 & 3 & 2 \\ 2 & 1 & 3 \\ 1 & 2 & 1 \\ 5 & 3 & 3 \end{pmatrix}$ which covers the matrix of $\begin{pmatrix} 4 & 2 & 1 \\ 6 & 3 & 2 \\ 2 & 1 & 3 \\ 1 & 2 & 1 \end{pmatrix}$.

Then, matrix R is reversed as $R' = \begin{pmatrix} 1 & 2 & 1 \\ 2 & 1 & 3 \\ 6 & 3 & 2 \\ 4 & 2 & 1 \end{pmatrix}$.

Then, the new solution can be obtained as $S' = \begin{pmatrix} 3 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 3 \\ 6 & 3 & 2 \\ 4 & 2 & 1 \\ 5 & 3 & 3 \end{pmatrix}$.

Table 1: Ordnance loading plan.

Aircraft type of ordnance	1	2	3
1	2	1	—
2	1	—	—
3	—	1	—
4	—	—	1

(1) Randomly generate a rectangle $R_{a\times b}$, with width $a \in (0, i]$ and length $b \in (0, m]$;
(2) Put R in the solution matrix S and mark the four vertices $(o_i^{xy}, o_{i+a}^{x,y}, o_{i+a}^{x+b,y+b}, o_i^{x+b,y+b})$;
(3) The subsequence of the R area is reversed, $o_i^{xy} \longrightarrow o_{i+a}^{x,y}$ and $o_i^{x+b,y+b} \longrightarrow o_{i+a}^{x+b,y+b}$;
(4) Get the new solution $S' = \begin{pmatrix} o_{i+a}^{xy} & \cdots & o_{i+a}^{x+b,y+b} \\ \vdots & \ddots & \vdots \\ o_i^{xy} & \cdots & o_i^{x+b,y+b} \end{pmatrix}$.

Algorithm 2: Matrix perturbation.

Hence, the matrix perturbation process is completed:

$$\begin{pmatrix} 3 & 1 & 2 \\ 4 & 2 & 1 \\ 6 & 3 & 2 \\ 2 & 1 & 3 \\ 1 & 2 & 1 \\ 5 & 3 & 3 \end{pmatrix} \longrightarrow \begin{pmatrix} 3 & 1 & 2 \\ 1 & 2 & 1 \\ 2 & 1 & 3 \\ 6 & 3 & 2 \\ 4 & 2 & 1 \\ 5 & 3 & 3 \end{pmatrix}.$$

3.4. Cooling Schedule. The SA behavior can be regulated by the temperature and its descent gradient. To avoid the local optimum trap, inferior solutions may be accepted depending on the falling temperature, under the mechanism of cooling schedule. Here, the exponential cooling rate is adopted:

$$T_l = \frac{(T_0 - T_f)(N+1)}{N(l+1)} + T_0 - \frac{(T_0 - T_f)(N+1)}{N}; \quad l = 1, 2, \ldots, N, \tag{10}$$

where T_0 and T_f are initial temperature and final temperature, respectively, and N is the number of temperatures between T_0 and T_f.

The SA needs to accept the new state through probability judgment, in order to avoid the local minimum. When the initial temperature is sufficiently high, the cooling is slow enough (i.e., each temperature is held for a sufficiently long time), and the final temperature approaches zero; the SA will converge to the global optimal solution with the probability of 1. However, it is very difficult to fulfil the global convergence condition. Besides, the current state may be worse than some intermediate states in the search trajectory, owing to the probability acceptance mechanism. Thus, the SA algorithm often converges to an approximate optimal solution, or a solution poorer than the best intermediate solution. The search efficiency is inevitably affected. To preserve the best-known state and improve search efficiency, this section makes the following improvements to the SA:

(1) Memorize the best intermediate solution in the search process and update it immediately. The improvement of memory turns the SA into an intelligent algorithm.
(2) Set up two thresholds, internal cycle threshold and external cycle threshold, to reduce the computing load while maintaining optimality. The internal cycle threshold refers to the number of cycles that the new solution of continuous disturbance does not generate a better solution at a certain temperature, while the external cycle threshold refers to the number of cycles that the new solution generated by continuous cooling does not generate a better solution. The two thresholds are determined as follows.

First, determine whether the number of internal cycles reaches the threshold; if yes, lower the temperature by one step; otherwise, judge if it conforms to Markov chain. If not, reconduct the process of state generation, state acceptance, and algorithm termination; otherwise, lower the temperature by one step. Second, determine whether the number of external cycles reaches the threshold; if yes, terminate the algorithm and obtain the final solution; otherwise, judge whether the algorithm meets the termination conditions. If yes, terminate the algorithm and obtain the final solution; otherwise, reconduct the process of state generation, state acceptance, and algorithm termination. Terminate the algorithm once the number of iterations i surpasses the prior fixed constant *MaxIter*. In our experiments, *MaxIter* was set to 10^4.

The flow of the improved SA is shown in Figure 3.

4. Experiments

To test the effectiveness of SA-based algorithm, we first evaluate the control factors of the SA and suggest a good parameter setting. Then, the solution quality and efficiency of the ISA were verified through several experiments. The algorithms are implemented in our previously published carrier-based flight operations simulation [28], which is written by C++ and ran on Microsoft Windows operating system with 4 GB RAM and dual core CPU.

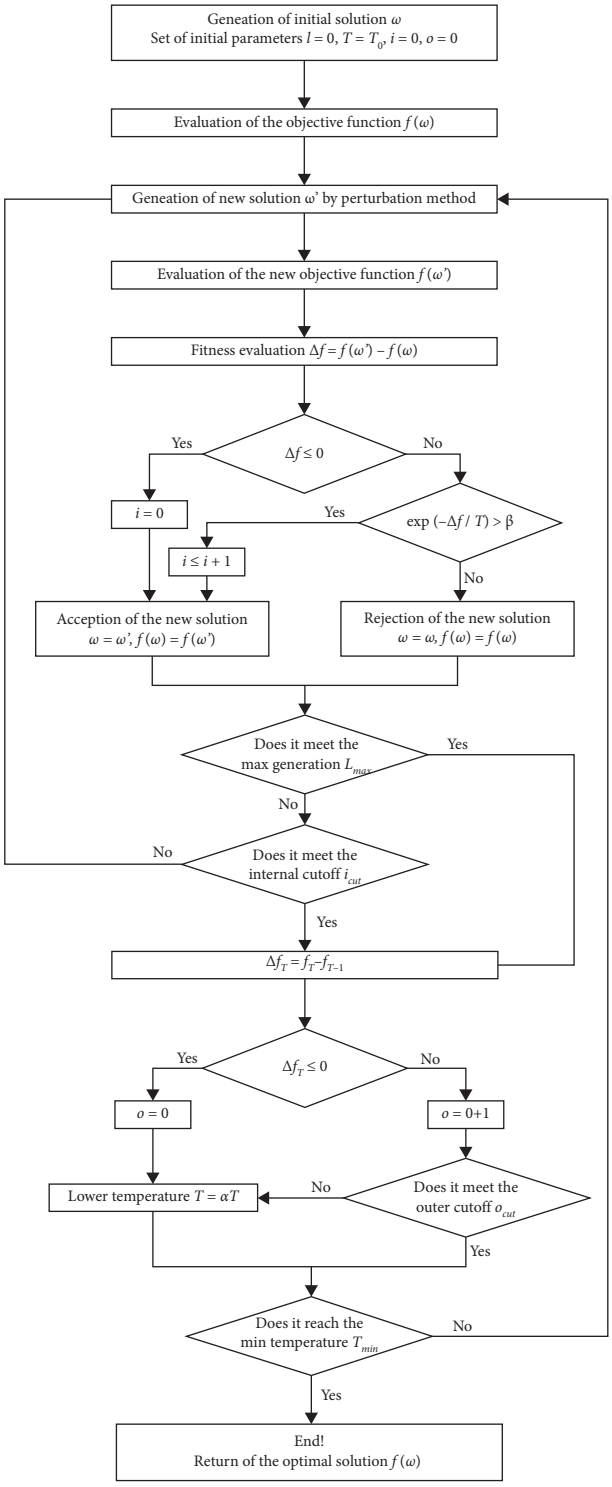


Figure 3: Flow of the improved SA.

4.1. Parameter Tuning. The efficiency of the SA algorithm is greatly affected by the design of parameters and operators. The full factorial design tests all possible combinations. Such an approach becomes too laborious in the face of numerous factors. Taguchi utilized orthogonal arrays to examine lots of decision variables in a few tests [29] and measured the importance of each factor by its influence on algorithm performance, using the signal-to-noise ratio: 10 log 10 $(\text{objective})^2$. Following Taguchi's method, the SA control factors were configured as follows: initial solution, initial temperature, cooling rate, and number of neighborhood searches in every temperature. Table 2 shows the different levels of these factors.

Hence, the SA has one 3-level factor and three 4-level factors. The best design among the orthogonal arrays is L16. Thus, additional transform was performed to fit L16 (Table 3).

The relative percentage deviation (RPD) was also adopted to measure the performances:

$$\text{RPD} = \frac{\text{Alg}_{\text{sol}} - \text{Min}_{\text{sol}}}{\text{Min}_{\text{sol}}} \cdot 100\%, \tag{11}$$

where the best solution obtained for one instance is denoted as Min_{sol}, while the objective value is marked as Alg_{sol}. Table 4 lists the S/N ratios and RPD values of each level of the factor value. The results show that A(3), B(4), C(4), and D(4) are the best levels of the factors.

4.2. Experimental Settings. The size of the test instances was set to $n = \{15, 30, 45, 60\}$ tasks, which corresponds to a common mission of strike sorties of 5, 10, 15, and 20 aircraft, respectively. The processing time for jobs on each machine was generated by triangle distribution with mean time according to [30]. There are resource constraints of five types of weapons, four lower-stage elevators, 10 assembling crews, four upper-stage elevators, and six loading crew members, see Tables 5–9 for comprehensive data.

4.3. Makespan Analysis. The test of problem uses an ordnance loading plan, ranging from 5 to 20 aircraft. At first, the experiment tries to solve a standard small size problem with optimal solution and preliminarily demonstrates the adaptability and feasibility of the ISA. Then, the proposed ISA was adopted to solve larger size problems and compared with the other methods to reveal its superiority.

To compare the ISA with the SA, control factors were configured as those in the preceding section. The permutation of machines was arranged in ascending order. The initial temperature $T_0 = 10^3$, final temperature $\varepsilon = 0.5$, cooling rate $\alpha = 0.99$, and length of Markov chain L = 2000.

The base case includes 15 tasks. As the temperature declined (Figure 4(a)), the ISA converged to the optimal solution in 2.705 s, faster than the SA, which converged to the optimal solution in 2.965 s (Figure 5(a)). Although the initial scheduling time of the ISA was higher than that of SA, its faster convergence procedure suggests the good performance of the ISA. Note that the computing time of the ISA was 85.3637 s, much shorter than that (143.7861 s) of the SA.

Next, the ISA performance on larger problems was tested, whereas more tasks bring a greater computing effort for the heuristic. Table 10 lists the average scheduling time and computing time of different tasks, with each task running for 30 times. Figure 6 shows the corresponding plot. The results showed that the ISA achieved a shorter

Table 2: The SA control factors.

Factors	Levels	Types
Initial solution (A)	3	(A1) randomness (A2) task-based (A3) ascending order
Initial temperature (B)	4	$B1 = 50$, $B2 = 100$, $B3 = 500$, $B4 = 1000$
Cooling rate (C)	4	$C1 = 0.85$, $C2 = 0.9$, $C3 = 0.95$, $C4 = 0.99$
Markov chain (d)	4	$D1 = 100$, $D2 = 500$, $D3 = 1000$, $D4 = 2000$

Table 3: Orthogonal array of L16 of our algorithm test.

Experiment	A	B	C	D
1	1	2	3	3
2	2	4	1	2
3	3	4	3	4
4	1	2	1	1
5	1	3	1	4
6	2	1	3	1
7	3	1	1	3
8	2	3	3	2
9	1	1	4	2
10	2	3	2	3
11	3	3	4	1
12	3	1	2	4
13	1	4	2	1
14	2	2	4	4
15	3	2	2	2
16	1	4	4	3

Table 4: Results of the orthogonal test.

Factor with level	Completion time	
	Mean S/N ratio	Mean RPD
Random initial (A1)	−73.6779	2.399
Descending initial (A2)	−73.6634	2.216
Ascending initial* (A3)	−73.6504*	2.060*
$T0 = 50$ (B1)	−73.6656	2.239
$T0 = 100$ (B2)	−73.6638	2.223
$T0 = 500$ (B3)	−73.6669	2.252
$T0 = 1000$* (B4)	−73.6629*	2.212*
Alpha = 0.85 (C1)	−73.7266	2.962
Alpha = 0.9 (C2)	−73.6791	2.397
Alpha = 0.95 (C3)	−73.6547	2.113
Alpha = 0.99* (C4)	−73.5987*	1.453*
$L = 100$ (D1)	−73.7602	3.360
$L = 500$ (D2)	−73.6716	2.310
$L = 1000$ (D3)	−73.6344	1.869
$L = 2000$* (D4)	−73.5930*	1.387*

Table 5: Ordnance loading plan.

Ordnance type	Spot task									
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
1	4(2)									
2		4(2)								
3			2(1)	2(1)	2(1)	2(1)	2(1)	2(1)	2(1)	2(1)
4			2(1)	2(1)	2(1)	2(1)	2(1)	2(1)	2(1)	2(1)
5			6(2)	6(2)	6(2)	6(2)				
6							2(2)	2(2)	2(2)	2(2)

Table 6: Ordnance assembling time (min).

Ordnance type	1	2	3	4	5	6
Loading time/skid	13	8	7	7	11	6

Table 7: Ordnance transport time on flight deck.

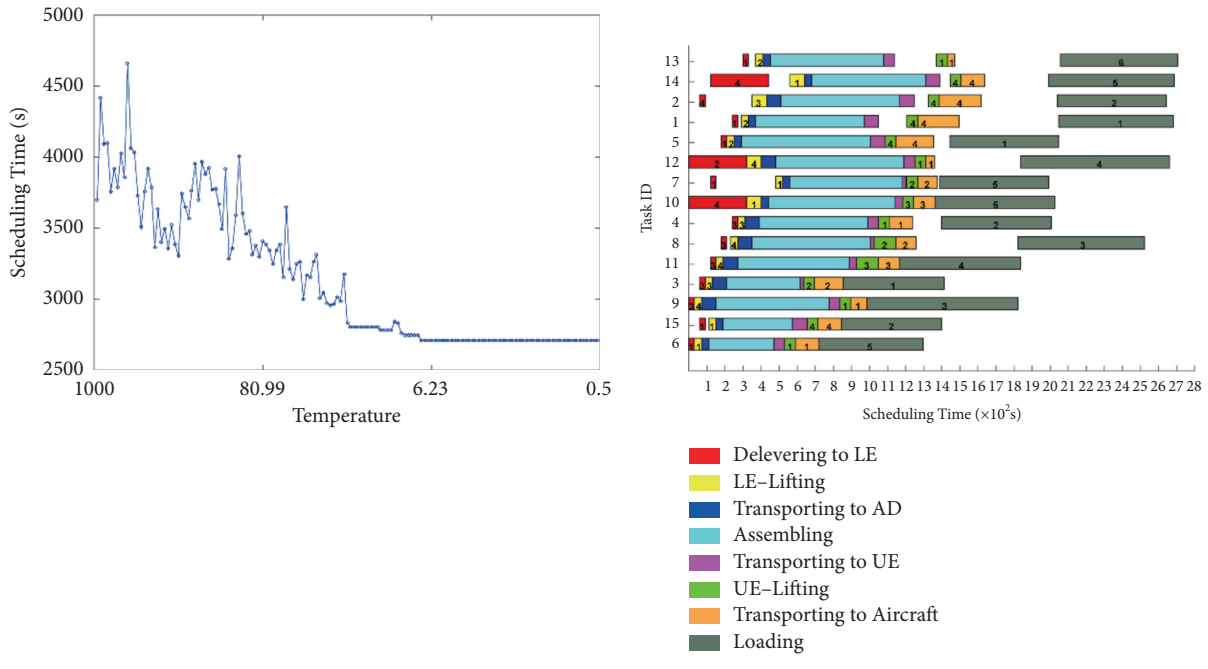
Time (s)	Spot									
Upper-stage elevator number	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
1	245	199	148	30	61	107	214	213	402	342
2	106	60	190	92	66	67	96	159	345	286
3	381	337	262	166	127	78	74	91	267	208
4	443	398	320	228	189	134	118	80	217	161

Table 8: Ordnance loading time (s).

Ordnance type	1	2	3	4	5	6
Time (skid/s)	660	600	480	480	780	360

Table 9: Ordnance crew walking time (s).

Spot	Spot									
	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}
A_1	0	47.53946	168.2	222.6	266.9	331	366	446.9	639	578.8
A_2	47.5	0	142.9	181.2	225.6	289.4	324.9	404.6	596.4	536.1
A_3	168.2	142.9	0	97.8	131.5	189.8	220	302.5	492	432.9
A_4	222.6	181.2	97.8	0	44.4	108.5	143.7	224.3	416.5	356.3
A_5	266.9	225.6	131.5	44.4	0	64.1	99.3	180.1	372.3	312.1
A_6	331	289.4	189.9	108.5	64.1	0	36.1	116	308.2	278.2
A_7	366	324.9	220	143.7	99.3	36.1	0	82.6	274	214.1
A_8	446.9	404.6	302.5	224.3	180.1	116	82.6	0	192.2	132
A_9	639	596.4	492	416.5	372.3	308.1623	274.0073	192.1666	0	60.3
A_{10}	578.8	536.1	432.9	356.3	312.1	278.2	214.1	132	60.3	0



(a) (b)

Figure 4: Optimal schedule of 15 tasks (5 aircraft) derived by the ISA. (a) Convergence curve. (b) Gantt chart.

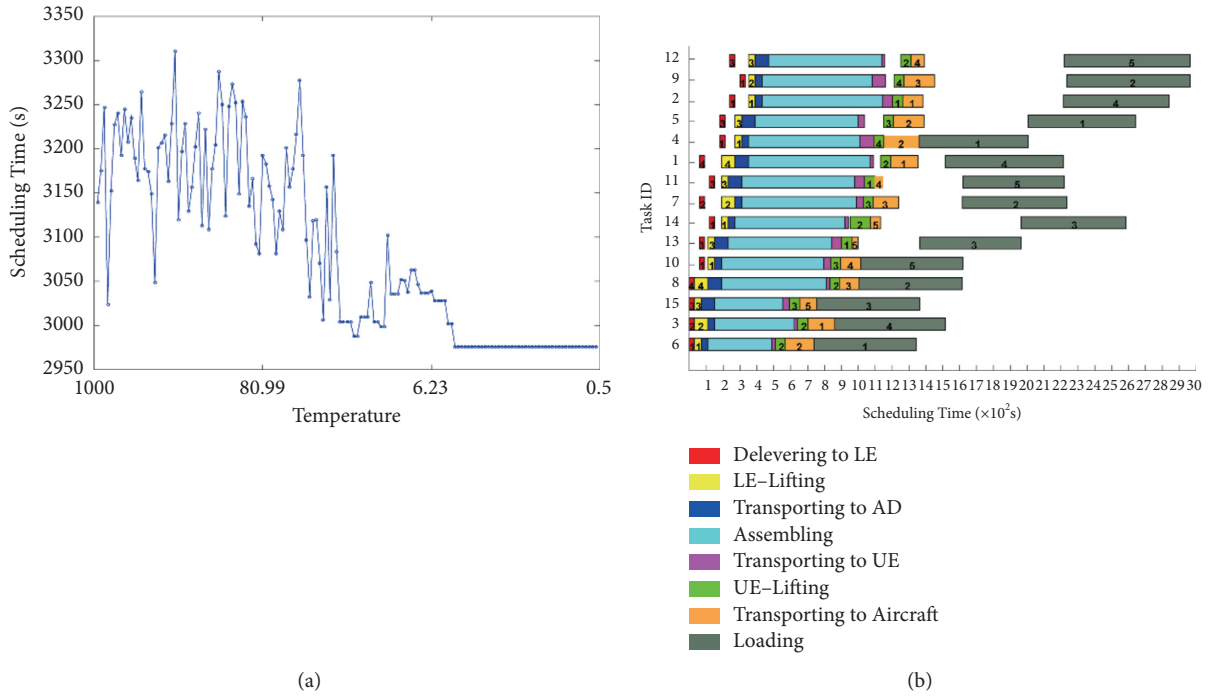


Figure 5: Optimal schedule of 15 tasks (5 aircraft) derived by the SA. (a) Convergence curve. (b) Gantt chart.

Table 10: Comparison between the ISA and the SA.

Task numbers	Average scheduling time (s)		Average computing time (s)	
	ISA	SA	ISA	SA
15	2965	3100	60	150
30	4200	4500	150	260
45	6100	6500	200	370
60	8000	8600	260	450

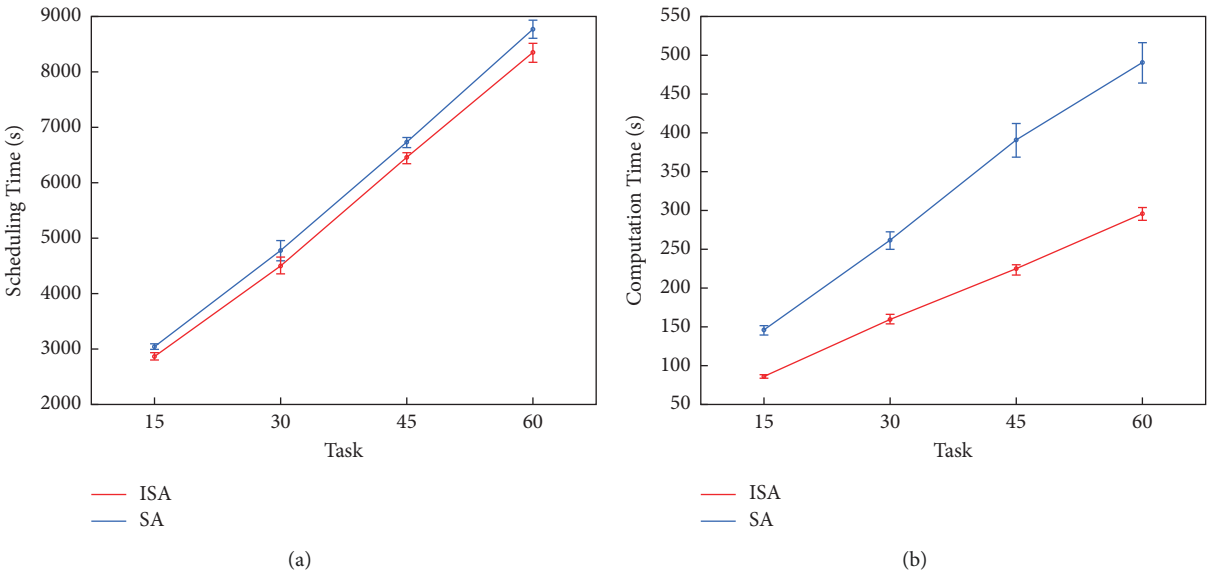


Figure 6: Performance of the ISA and the SA on medium and large problems. (a) Scheduling time. (b) Computing time.

scheduling time and a faster computing speed than the SA. Comparing with the SA, the average scheduling time of the ISA reduces from 300 to 600 seconds in each ordnance turnaround cycle for 30 to 60 tasks and the average computing time saves from 110 to 190 seconds. Note that there are usually ten or more cycles in a general flight day; the scheduling time saved by implementing ISA equals one more group sortie generation, which in turn enhanced the firepower capacity of carrier air wing.

The results also show that the bottleneck of ordnance handling is the loading process, where the number of loading crews heavily influences the aircraft turnaround time. When there is high intensity of surge operations, more loading crews should be arranged to handle ordnances.

5. Conclusions

This paper treats the aviation ordnance scheduling problem under the HFS framework with multistages, independent parallel machines, and several processing constraints. The simulated annealing algorithm was modified with dual threshold selection to generate faster and better schedules using the proposed matrix perturbation method that keeps the SA independent of the heuristic schemes. The influencing parameters of the improved algorithm are carefully tuned by Taguchi's method. The experimental results demonstrate the effectiveness of ISA, which provides a practical solution to a broad application in dealing with stochastic hybrid flow shop scheduling problems.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] A. Jewell, *Sortie Generation Capacity of Embarked Airwings*, Center For Naval Analyses Alexandria Va, Arlington, VA, USA, 1998.

[2] Naval Air Systems Command, *CV Naval Air Training and Operating Procedures Standardization*, NATOPS Manual, Patuxent River, MD, USA, 2009.

[3] R. Ruiz and J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *European Journal of Operational Research*, vol. 205, no. 1, pp. 1–18, 2010.

[4] H. Emmons and G. Vairaktarakis, *Flow Shop Scheduling: Theoretical Results, Algorithms, and Applications*, Springer Science & Business Media, Berlin, Germany, 2012.

[5] E. Mokotoff, "Parallel machine scheduling problems: a survey," *Asia-Pacific Journal of Operational Research*, vol. 18, no. 2, p. 193, 2001.

[6] M. M. Ahmadian, M. Khatami, A. Salehipour, and T. C. E. Cheng, "Four decades of research on the open-shop scheduling problem to minimize the makespan," *European Journal of Operational Research*, vol. 259, no. 2, pp. 399–426, 2021.

[7] A. O. Bedhief and N. Dridi, "A genetic algorithm for three-stage hybrid flow shop scheduling problem with dedicated machines," *Journal Européen des Systèmes Automatisés*, vol. 53, no. 3, pp. 357–368, 2020.

[8] J. N. D. Gupta, "Two-stage, hybrid flowshop scheduling problem," *Journal of the Operational Research Society*, vol. 39, no. 4, pp. 359–364, 1988.

[9] G.-C. Lee and Y.-D. Kim, "A branch-and-bound algorithm for a two-stage hybrid flowshop scheduling problem minimizing total tardiness," *International Journal of Production Research*, vol. 42, no. 22, pp. 4731–4743, 2004.

[10] T. Sawik, "Integer programming approach to reactive scheduling in make-to-order manufacturing," *Mathematical and Computer Modelling*, vol. 46, no. 11-12, pp. 1373–1387, 2007.

[11] I. Ribas, R. Leisten, and J. M. Framiñan, "Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective," *Computers & Operations Research*, vol. 37, no. 8, pp. 1439–1454, 2010.

[12] G.-C. Lee, "Estimating order lead times in hybrid flowshops with different scheduling rules," *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1668–1674, 2009.

[13] G. Bektur and T. Saraç, "A mathematical model and heuristic algorithms for an unrelated parallel machine scheduling problem with sequence-dependent setup times, machine eligibility restrictions and a common server," *Computers & Operations Research*, vol. 103, pp. 46–63, 2019.

[14] T. Yang, Y. Kuo, and C. Cho, "A genetic algorithms simulation approach for the multi-attribute combinatorial dispatching decision problem," *European Journal of Operational Research*, vol. 176, no. 3, pp. 1859–1873, 2007.

[15] D. Li, C. Liu, and K. Li, "A remanufacturing logistics network model based on improved multi-objective ant colony optimization," *Journal Européen des Systèmes Automatisés*, vol. 52, no. 4, pp. 391–395, 2019.

[16] F. Glover and M. Laguna, "Tabu search, handbook of combinatorial optimization," in *Handbook of Combinatorial Optimization*, pp. 2093–2229, Springer, Boston, MA, USA, 1998.

[17] C. B. Rabah, G. Coatrieux, and R. Abdelfattah, "Boosting up source scanner identification using wavelets and convolutional neural networks," *Traitement du Signal*, vol. 37, no. 6, pp. 881–888, 2020.

[18] D. A. B. Fernandes, M. M. Freire, P. A. P. Fazendeiro, and P. R. M. Inácio, "Applications of artificial immune systems to computer security: A survey," *Journal of Information Security and Applications*, vol. 35, pp. 138–159, 2017.

[19] D. Delahaye, S. Chaimatanan, and M. Mongeau, "Simulated annealing: From basics to applications," *Handbook of Metaheuristics*, vol. 272, pp. 1–35, 2019.

[20] N. Maaroju, "Choosing the Best Heuristic for a NP-Problem," Doctoral Dissertation, Thapar University, Patiala, Punjab, 2009.

[21] A. Franzin and T. Stützle, "Revisiting simulated annealing: A component-based analysis," *Computers & Operations Research*, vol. 104, pp. 191–206, 2019.

[22] K. Wal□dzik and J. Mańdziuk, "Applying hybrid Monte Carlo tree search methods to risk-aware project scheduling problem," *Information Sciences*, vol. 460, pp. 450–468, 2018.

[23] Commander Naval Air Systems Command, *CV Flight/Hangar Deck NATOPS Manual*, NAVAIR 00-80T-120, Patuxent River, MA, USA, 2005.

[24] P. C. Goshopn, "Aviation ordnanceman," *Navel Education and Training Professional Development and Technology Center. Navedtra*, vol. 14313, 2001.

[25] D. Leitold, A. Vathy-Fogarassy, and J. Abonyi, "Empirical working time distribution-based line balancing with integrated simulated annealing and dynamic programming," *Central European Journal of Operations Research*, vol. 27, no. 2, pp. 455–473, 2019.

[26] S. Rajendran, C. Rajendran, and R. Leisten, "Heuristic rules for tie-breaking in the implementation of the NEH heuristic for permutation flow-shop scheduling," *International Journal of Operational Research*, vol. 28, no. 1, pp. 87–97, 2017.

[27] B. Maenhout and M. Vanhoucke, "A perturbation matheuristic for the integrated personnel shift and task rescheduling problem," *European Journal of Operational Research*, vol. 269, no. 3, pp. 806–823, 2018.

[28] N. Wang, X. Meng, Q. Liu, and J. Li, "High level architecture based simulation for aircraft carrier deck operations," in *Proceedings of the 2016 IEEE Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC)*, pp. 765–769, Xi'an, China, October 2016.

[29] J. D. Kechagias, K.-E. Aslani, N. A. Fountas, N. M. Vaxevanidis, and D. E. Manolakos, "A comparative investigation of Taguchi and full factorial design for machinability prediction in turning of a titanium alloy," *Measurement*, vol. 151, p. 107213, 2020.

[30] A. Jewell, M. A. Wigge, C. M. Gagnon, L. A. Lynn, and K. M. Kirk, *USS Nimitz and Carrier Airwing Nine Surge Demonstration*, Center for Naval Analyses Alexandria Va, Arlington, VA, USA, 1998.