Hindawi
Journal of Applied Mathematics
Volume 2023, Article ID 6680897, 17 pages
https://doi.org/10.1155/2023/6680897

Research Article

INEH-VNS Algorithm Solved Automatic Production System Scheduling Problem under Just-in-Time Environment

Li Qingxiang,[1] **Zhao Xiaofei**,[2] **He Yude,**[3,4] **and Yin Shaojun**[5]

[1]*College of Artificial Intelligence, Chongqing University of Arts and Sciences, Yongchuan 402160, China*
[2]*School of Economics and Management, Chongqing University of Arts and Sciences, Yongchuan 402160, China*
[3]*School of Management, Hebei GEO University, Shijiazhuang 050031, Hebei Province, China*
[4]*Strategy and Management Base of Mineral Resources in Hebei Province, China*
[5]*College of Science, Xichang University, Xichang 615000, Sichuan Province, China*

Correspondence should be addressed to Zhao Xiaofei; 20080012@cqwu.edu.cn

Received 4 June 2023; Revised 5 October 2023; Accepted 6 October 2023; Published 17 November 2023

Academic Editor: Wei-Chiang Hong

Copyright © 2023 Li Qingxiang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Automatic production system scheduling problem under a just-in-time environment is researched in this paper. The automatic production system is composed of many tanks and one robotic, the tank of the researched problem is responsible for processing the job, and the robotic moves the job from one tank to the other tank. The difference between the researched problem and the classic shop scheduling problem is that the former must consider job scheduling and the robotic move sequence, but the latter considers only job scheduling. For optimizing simultaneously job scheduling and robotic move sequence in the proposed problem and minimizing total earliness/tardiness, an improved NEH (Nawaz-Enscore-Ham) and variable search (INEH-VNS) algorithm are developed. In the proposed method, firstly, to obtain initial solution, an improved NEH is shown. Secondly, for computing value of the objective function, the double procedure method is constructed. Thirdly, according to the properties of the proposed problem, three neighborhood structures, adjacent exchange, random insertion, and job exchange, are investigated. To test the performance of the INEH-VNS, 100 instances are randomly generated. When the run time is the same, compared with CPLEX 12.5, the INEH-VNS algorithm can find high-quality approximate optimal solution, a special big scale. Compared with the G-VNS algorithm, the average improvement rate of the approximate optimal solution is 45.9%, and the average stability rate of the INEH-VNS algorithm enhances 75.04%. That is to say, the INEH-VNS algorithm is outstanding and more effective.

1. Introduction

The competition between enterprises in the market is becoming increasingly intense. Therefore, the concept of just-in-time (JIT), which is paid attention to by many industries and has been widely adopted over the years to improve production efficiency, stems from the production field [1, 2]. One of the main concepts in JIT production systems is that the makespan is equal to the due date, avoiding unnecessary inventory and late delivery; that is to say, the makespan is not more than the due date, and the makespan is not less than the due date [3]. If the makespan is less than the due date, many questions, e.g., inventory cost, product damage, and economic depreciation, will be generated. If the makespan is more than the due date, a variety of adverse consequences, e.g., customer loss, contract default, and commercial reputation damage, will arise [4].

Automatic production system which is applied to printed circuit board (PCB) industry, wafer fabrication, automobile manufacturing industry, and steel industry [5–9] is an advanced intelligent manufacturing system. The products which are manufactured by the automatic production system are easy to depreciate, so JIT production mode is adopted. In published literatures, the classical shop scheduling problem considering due date is not researched in robotic move sequence [10–13]. The automatic production system scheduling problem is studied by robotic move sequence but is not considered the due date [5–9]. To the best of our knowledge, the scheduling problem considering JIT environment and robotic move sequence is studied seldom. Due to the significance of JIT production in industrial environment to overcome variations from the demand providing customer satisfaction and widely applied of automatic production system in industrial environment, in this research, an improved NEH and variable neighborhood search (INEH-VNS) that optimizes the proposed problem is developed for improving production efficiency of the automatic production system and alleviating the contradiction between supply and demand.

In this paper, the automatic production system scheduling problem considering due date is investigated. The proposed automatic production system is composed of a number of tanks, including one input station, one output station, and one robotic which is controlled by computer. All jobs which will be processed are loaded into the input station and are processed in each tank in order; finally, all jobs which have been finished are loaded into the output station. Each job is moved from one tank to the other tank. The due date of each job is given.

To satisfy the due date of the job as much as possible, the INEH-VNS is designed for optimizing total earliness/tardiness of more than one tank's automatic production system scheduling problem. Our contributions are listed in the following:

(1) According to the properties of the proposed problem, the improved NEH is developed for generating initial solution

(2) To compute the value of the objective function, the double course method is proposed

(3) According to the solution properties of the proposed problem, three neighborhood constructs are shown for designing variable neighborhood search (VNS).

(4) In the previous work, the automatic production system scheduling problem with one tank under JIT environment is studied. In this paper, the INEH-VNS is developed for solving the automatic production system scheduling problem with more than one tank under JIT environment and optimizing simultaneously the job scheduling and the robotic move sequence

The remainder of this paper is organized as follows. Section 2 describes the literature on scheduling problems minimizing total earliness/tardiness. Section 3 presents the INEH-VNS. Section 4 exhibits computational results. Finally, the conclusions and the challenge work are given in Section 5.

2. Literature Review

The shop scheduling under JIT environment means the due date and the makespan are as consistent as possible, so the objective of the shop scheduling problem under JIT environment is to minimize total earliness/tardiness. In this section, the literatures which the objective is to optimize total earliness/tardiness are described.

For the single-machine scheduling problem under JIT environment, the mixed integer programming model is developed by Woodruff and Spearman when the due date is given [14]. Coleman researched the problem with job scheduling sequence-dependent setup times and investigated the mixed integer programming model [15]. To obtain the total weighted earliness/tardiness of job scheduling sequence-dependent switching cost, the nonlinear mathematic programming model is constructed [16]. For the problem with quadratic earliness/tardiness penalties, the mixed integer programming model with uncertainty due date is shown [17]. The exact and heuristic methods are proposed [18] and improved the result of literature [17]. According to literature [17], the mathematic programming with considering learning effect and release dates is exhibited [19]. Kayvanfar et al. researched the single machine scheduling problem with parallel machine under JIT environment and minimized total earliness/tardiness [11]. The single machine scheduling problem considering general due date is studied; if the objective function is to minimize the number of tardiness jobs, the problem is polynomially solvable; if the objective function is to optimize the number of weighted tardiness jobs, the problem is NP hard; if the problem is extended to allow job rejection, the problem is NP hard [20]. For multi-scenario scheduling problem, to obtain the number of weighted JIT jobs, the problem is solvable in polynomial time [21]. Otherwise, if the objective function is a total weighted makespan, the problem is solvable in (is the number of jobs) [22]. When the due dates are assigned to each job depending on its order, and the objective is to minimize the total penalty for the earliness and tardiness of each job, the problem is NP hard [16]. For solving the single machine scheduling problem under JIT environment, the decomposition algorithm is proposed, so the upper bound and the lower bound are obtained, and the optimal solution is found by the proposed branch and bound; for optimizing total weighted tardiness, four heuristic methods are designed [23]. Mosheiov et al. introduced pseudopolynomial dynamic programming algorithms for minimizing the number of tardy jobs when the jobs allowed rejection [20]. Alidaee reviewed the published mathematic programming and then introduced future work [12].

For the two machines scheduling problem under JIT environment, the problem with simultaneous consideration of common due date assignment, convex resource allocation, and learning effect is solvable in polynomial time for minimizing the total resource consumption cost [24]. Shi and Wang researched the problem with common due window assignment, learning effect, and resource allocation, and they proofed the problem is solvable in polynomial time for optimizing the linear weighted sum of job earliness, tardiness, due window size, and resource cost [25]. The results are the same if the slack due date substitutes for common due date [26, 27]. Based on the results of literature [25, 27], for minimizing the scheduling cost and the resource consumption cost, the scheduling problem with different due window assignment and learning effects is investigated, and the polynomial algorithm is proposed [28]. The classical scheduling problem models considering due date assignment are extended and solvable in polynomial time [29].

For more than two machines scheduling problem under JIT environment, Jiang et al. addresses the problem considering position-dependent weights and discusses the common due date assignment model and slack due date assignment model. The polynomial time algorithm is developed for minimizing total cost [30]. Lv et al. proposed polynomial time algorithms for the problem in literature, the computational time complexity is reduced from O(n2logn) to O(nlogn) [31]. For the problem with two distinct due dates, Koulamas et al. show that the proportionate flow shop problem with variable machine speeds remains NP-hard; they then show that the no-wait problem with ordered jobs and a maximal last machine is solvable in time; They also show that the corresponding problem with the minimum number of tardy jobs is solvable in time [32].

In summary, for the flow shop scheduling problem under JIT environment, mathematics model, optimization algorithm, and computational complexity are studied, but the robot moving job between the operations is not considered. Thus, the results of the literatures [14–32] cannot be applied to the proposed problem in this paper.

For the automatic production system scheduling problem, Zhao and Guo [9] and Zhao and Guo [33] constructed optimization algorithm, respectively, for optimizing the production cycle. Wu et al. proposed a differential evolution algorithm for solving robotic cell scheduling problem with batch-processing machines [34]. Majumder et al. researched robotic cell scheduling problem with sequence-dependent setup times and addressed a bacterial foraging optimization algorithm [35]. Elmi et al. showed an integer programming for cyclic flow shop robotic cell scheduling problem with multiple part types [36]. The objective of the literatures [9, 33–36] is to minimize the production cycle. Wang et al. studied the automatic production system scheduling problem under JIT environment, developed a mixed integer programming, and proposed a hybrid guided tabu search algorithm to optimize the makespan and total weighted tardiness [37]. The difference between the proposed problem and the problem of literature [37] is that the parallel-tank scheduling problem with hoist and group constraints is researched in literature [37], but more than one tank flow shop scheduling problem with hoist is investigated in this paper, so the method of the literature [37] cannot be applied in this paper. The method for solving the proposed problem is worth studying.

Since the automatic production system scheduling problem is NP-hard, all kinds of algorithms, e.g., chemical reaction optimization [9], branch and bound algorithm [33], differential evolution algorithm [34], bacterial foraging optimization algorithm [35], and hybrid guided tabu search algorithm [37], are proposed for obtaining the optimal solution. There are two shortcomings in these algorithms; one is it is difficult to find a solution for big scale problem, and the other is it is troublesome to find parameter values for giving problem. Variable neighborhood search that is proposed by Mladenović and Hansen [38] does not involve parameters and can find optimal solution of the problem. Variable neighborhood search is applied to solve various combinatorial optimization problem [39–42]. In current, it is seldom found that variable neighborhood search is used to solve the automatic production system scheduling problem under just-in-time environment. In addition, for enhancing the performance of the proposed algorithm, the improved NEH that is developed generates initial solution. In this paper, an improved NEH and variable neighborhood search, namely, INEH-VNS, are investigated for minimizing total earliness/tardiness and finding the optimal job scheduling and robotic move sequence.

3. Problem Description

The automatic production system is composed of one input station T_0, m tanks T_1, T_2,$\cdots$,T_m, one output station T_{m+1}, and single robot in this paper. The input station T_0 contains all jobs which will be processed, the output station T_{m+1} holds all jobs which have been finished, and all tanks T_1, T_2, $\cdots$, T_m process jobs. The robot moves the job from one tank to the other tank. n jobs are processed by the automatic production system at the same time. Firstly, each job is unloaded from the input station T_0. Then, the job is successively processed in tank T_1, T_2, $\cdots$, T_m. Finally, the job is loaded into the output station T_{m+1}, that means the job is finished. The processing time of each job is not all same in tank $T_i (i = 1, 2, \cdots, m)$. So, the proposed problem in this paper is a flow shop scheduling problem with single robot. It is difficult that job scheduling and robotic move sequence are simultaneously optimized in current researched problem. At the beginning of the period, the input station contains all unprocessed jobs, and all tanks and the output station are empty, which is different from published literatures automatic production system scheduling problem, because the tanks of published literatures automatic production system scheduling problem are not all empty. At any time, the robot can hold only one job at most, and the tank can contain only one job at most. Furthermore, processing interruptions and preemption are not considered. In this paper, three operations: (i) the robot unloads job from tank T_i; (ii) the robot carries the job from tank T_i to tank T_{i+1}; (iii) the robot loads the job into tank T_{i+1} $(i = 0, 1, 2, \cdots, m)$, are called robotic activity. The robot moves from tank T_i to tank $T_k (i, k = 0, 1, 2, \cdots, m, m + 1)$ is named void move. For satisfying the due date of job as much as possible, the objective of total earliness/tardiness is proposed in this paper. To optimize the objective value, three types of constraint, processing time constraints, robot-containing constraints, and tank capacity constraints, must be considered.

There is an obvious difference between the proposed problem and the classic shop scheduling problem in that job scheduling and the robotic move sequence of the proposed problem are simultaneously optimized. Since the flow shop scheduling problem for two machines is NP-hard to obtain total earliness/tardiness [43], the proposed problem is also NP-hard. The notations that are used in this paper are defined as follows:

Decision variables:

$t_{i,j}$: the robotic activity start time ($j = 1, 2, \cdots, n$; $i = 0, 1, 2, \cdots, m$).

Intermediate variables:

$y_{i,j,k,l}$: binary variables. When $t_{i,j} < t_{k,l}$ is true, $y_{i,j,k,l} = 1$, otherwise $y_{i,j,k,l} = 0$ ($j, l = 1, 2, \cdots, n$; $i, k = 0, 1, 2 \cdots, m$).

TT: total earliness\tardiness time.

T_j: earliness\tardiness time of the jth($j = 1, 2, \cdots, n$). job.

C_j: makespan of the jth($j = 1, 2, \cdots, n$). job.

Parameters or notations:

$l_{i,j}$: the processing time lower bound of the jth ($j = 1, 2, \cdots, n$) job in tank T_i($i = 1, 2, \cdots, m$).

d_j: the due date of the jth($j = 1, 2, \cdots, n$) job.

$mt_{i,j}$: the time of robotic activity($j = 1, 2, \cdots, n$;$i = 0, 1, 2, \cdots, m$).

$em_{i,k}$: the time of void move($i, k = 0, 1, 2, \cdots, m, m + 1$).

Other notations:

δ: $\{1, 2,\cdots,n\} \longrightarrow \{1, 2,\cdots,n\}$, the jobs permutation.

σ: $\{0, 1,\cdots,n(m + 1) - 1\} \longrightarrow \{0, 1,\cdots,n(m + 1) - 1\}$, the robotic activities permutation.

$\sigma(k)$: the kth robotic activity in feasible solution($k = 0, 1, \cdots, n(m + 1) - 1$).

$[\sigma(k)]$: the kth robotic activity corresponds to the subscript of the tank in feasible solution($k = 0, 1, \cdots, n(m + 1) - 1$).

$\lfloor\sigma(k)/(m + 1)\rfloor + 1$: the kth robotic activity corresponds to the job in feasible solution($k = 0, 1, \cdots, n(m + 1) - 1$).

$t_{\sigma(k)}$: the start time of robotic activity $\sigma(k)$ and $t_{\sigma(0)} = 0$ ($k = 0, 1, \cdots, n(m + 1) - 1$).

The mixed integer programming is shown.

$$\text{Minimize} \quad TT = \sum_{j=1}^{n} \left|C_j - d_j\right|. \tag{1}$$

$$\text{Subject to} \quad t_{i+1,j} - t_{i,j} \geq \text{lm}_{i,j} + l_{i,j}, j = 1, 2, \cdots, n\, ; i = 0, 1, 2, \cdots, m - 1, \tag{2}$$

$$t_{i,j} - t_{k,l} \geq \text{lm}_{k,l} + \text{em}_{k+1,i} - M\left(1 - y_{i,j,k,l}\right), j, l = 1, \cdots, n\, ; i, k = 0, 1, \cdots, m\, ; i \neq k \,\text{and}\, i \neq k + 1 \,\text{or}\, j \neq l, \tag{3}$$

$$t_{k,l} - t_{i,j} \geq \text{lm}_{i,j} + \text{em}_{i+1,k} - \text{M}y_{i,j,k,l}, j, l = 1, \cdots, n\, ; i, k = 0, 1, \cdots, m\, ; i \neq k \,\text{and}\, i + 1 \neq k \,\text{or}\, j \neq l, \tag{4}$$

$$y_{i,h,i-1,h} + y_{i-1,h,i,j} + y_{i,j,i-1,j} + y_{i-1,j,i,h} = 3, i = 1, 2, \cdots, m, j, h = 1, 2, \cdots, n \,\text{and}\, j \neq h, \tag{5}$$

$$C_j = t_{m,j} + mt_{m,j}, j = 1, 2, \cdots, n, \tag{6}$$

$$\text{mt}_{i,j} \geq \text{em}_{i,i+1} \quad j = 1, \cdots, n\, ; i = 0, 1, \cdots, m, \tag{7}$$

$$\text{em}_{i,p} \leq \text{em}_{i,k} + \text{em}_{k,p} \quad i, p, k = 0, 1, \cdots, m + 1\, ; i \neq p\, ; i \neq k\, ; p \neq k, \tag{8}$$

$$t_{i,j} \geq 0, j = 1, 2, \cdots, n\, ; i = 0, 1, 2, \cdots, m, \tag{9}$$

$$y_{i,j,k,l} = 0 \,\text{or}\, 1, j, l = 1, 2, \cdots, n, i, k = 0, 1, 2, \cdots, m. \tag{10}$$

The objective function that is to minimize total earlines-s\tardiness is shown by equality (1). Inequality (2) is a processing time constraint. Inequality (3) and Inequality (4) are robots containing constraints. Equality (5) is the tank capacity constraint. Makespan is computed by equality (6). Inequality (7) and Inequality (8) are triangle inequality constraints. Inequality (9) and Inequality (10) are nonnegative constraints and binary variables, respectively.

4. INEH-VNS

Variable neighborhood search that improves local search algorithm is composed of initial solution, neighborhood structure, and stop condition. Next initial solution and neighborhood structure of the proposed algorithm that are designed are introduced.

4.1. INEH. Since the job scheduling and the robot move sequence of the proposed problem are optimized, if the job scheduling is discussed first of all and, secondly, the robot move sequence is optimized, it is hard to find the optimal solution of the proposed problem, vice versa. In addition, since the job scheduling and the robot move sequence are not one-to-one, it is not easy to find the optimal solution to the current researched problem if the job scheduling or the robot move sequence represents the initial solution. In this paper, to obtain initial solution, the robotic activity sequence [12] that is composed of the job scheduling and the robot move sequence is applied. It has been proven that the NEH heuristic is an effective heuristic\method to address flow-shop scheduling problems [44]. The improved NEH is developed for constructing the initial solution, and the detailed process is as follows:

Step 1. Since the time which the job moves from tank T_i to tank T_{i+1} is considered. In order to apply NEH method, the processing time of the job, called $l'_{i+1,j}$, is revised. The detailed process is shown in equation (3).

$$l'_{i+1,j} = l_{i+1,j} + \text{mt}_{i+1,j}. \tag{11}$$

Step 2. Compute the sum of the processing time of the jth ($j = 1, 2, \cdots, n$) job in all tanks, named sum_j. It means equation (4) holds

$$\text{sum}_j = \sum_{i=1}^{m} l'_{i,j}(j = 1, 2,\cdots,n). \tag{12}$$

Step 3. All sum_j($j = 1, 2, \cdots, n$) are arranged in descending order; then, the job sequence set J is obtained.

Step 4. The first two robotic activities of the first job of set J are inserted in the first two places of the initial solution, namely, S. The last two robotic activities of the last job of set J are inserted in the last two places of initial solution S.

Step 5. The first robotic activity of each job, except the first job and the last job, in set J is inserted in the place of initial solution S in turn.

Step 6. According to property 1 (the appendix), the remainder robotic activities of each job in set J are inserted in the initial solution S; then, the initial solution S is obtained.

Example 1. The number of jobs is four, and the number of tanks, including the input station and output station, is five. The processing time of jobs is listed in Table 1. The robotic activities are shown in Table 2. The time of the void move $\mathrm{em}_{i,k}$ is equal to $2|i-k|$ $(i,k=0,1,2,\cdots,5)$. The time of robotic activity $mt_{i,j}$ is 6 $(i=0,1,2,3,4,j=1,2,3,4)$. The due date of jobs is $(d_1,d_2,d_3,d_4)=(4,3,2,1)$.

According to Step 1, Step 2, and Step 3, the set $J=\{1,2,3,4\}$. Figures 1(a) and 1(b) are obtained according to Step 4 and Step 5, respectively. Figures 1(c) and 1(b) are obtained according to Step 6. Finally, the initial solution S, Figure 1(d), is given.

4.2. Double Procedure Method. In order to calculate the makespan $c_j (j=1,2,3,4)$, the double procedure method is developed. Firstly, the relation between the makespan c_j $(j=1,2,3,4)$ and the start time of robotic activity is built. Secondly, the start time of robotic activity is computed that is separated into two procedures: one is the complete time of job on the tank, and the other is the time of void move. If the complete time of job on the tank is more than the time of the void move, the start time of the robotic activity is equal to the complete time of job on the tank, vice versa. If $[\sigma(k)]=m$ is true, the makespan $c_{\lfloor\sigma(k)/(m+1)\rfloor+1}=t_{\sigma(k)}+\mathrm{mt}_{m,\lfloor\sigma(k)/(m+1)\rfloor+1}$ holds, where the $t_{\sigma(k)}$ is obtained according equation (5), equation (14), and equation (15).

$$t_{\sigma(k)}{}' = t_{\sigma(k\text{-}1)} + \mathrm{mt}_{[\sigma(k\text{-}1)],\lfloor\sigma(k\text{-}1)/(\mathrm{m}+1)\rfloor+1} + l_{[\sigma(k)],\lfloor\sigma(k)/(\mathrm{m}+1)\rfloor+1} \left\lfloor\frac{\sigma(k)}{m+1}\right\rfloor = \left\lfloor\frac{\sigma(k\text{-}1)}{m+1}\right\rfloor, \tag{13}$$

$$t_{\sigma(k)}{}'' = t_{\sigma(k\text{-}1)} + \mathrm{mt}_{[\sigma(k\text{-}1)],\lfloor\sigma(k\text{-}1)/(\mathrm{m}+1)\rfloor+1} + \mathrm{em}_{[\sigma(k\text{-}1)]+1,[\sigma(k)]} \left\lfloor\frac{\sigma(k)}{m+1}\right\rfloor \neq \left\lfloor\frac{\sigma(k\text{-}1)}{m+1}\right\rfloor, \tag{14}$$

$$t_{\sigma(k)} = \max\left\{t_{\sigma(k)}{}', t_{\sigma(k)}{}''\right\}. \tag{15}$$

Example 2. The relation between the robotic activities in feasible solution is exhibited in Figure 2. The start time of each robotic activity is computed as follows. So, the makespan c_1 is equal to 62.

$t_{\sigma(1)}=t_{\sigma(0)}+mt_{0,1}+l_{1.1}=0+6+2=8$, $t_{\sigma(2)}=t_{\sigma(1)}+mt_{1,1}+l_{2.1}=8+6+4=18$, $t_{\sigma(3)}=t_{\sigma(2)}+mt_{2,1}+l_{3.1}=18+6+6=30$, $t_{\sigma(4)}=t_{\sigma(3)}+mt_{3,1}+em_{3+1.0}=30+6+4\times 2=44$, $t_{\sigma(5)}{}'=t_{\sigma(3)}+mt_{3,1}+l_{4.1}=30+6+8=44$, $t_{\sigma(5)}{}''=t_{\sigma(4)}+mt_{0,2}+\mathrm{em}_{1,4}=44+6+6=56$, $t_{\sigma(5)}=\max\left(t_{\sigma(5)}{}', t_{\sigma(5)}{}''\right)=\max(44,56)=56c_1=t_{\sigma(5)}+mt_{4,1}=56+6=62$.

After the double procedure method is applied, $c_2=140$, $c_3=192$, and $c_4=260$ are true.

4.3. Neighborhood Structure. The feasible solution properties of the proposed problem are applied, and three neighborhood structures, adjacent exchange, random insertion, and job exchange, are shown.

4.3.1. Adjacent Exchange. The $S=(\sigma(0),\sigma(1),\cdots,\sigma(k-1),\sigma(k),\sigma(k+1),\sigma(k+2),\cdots,\sigma(n(m+1)-1))$ represents a feasible solution of the current researched problem, and $\sigma(k)$ and $\sigma(k+1)$ are two robotic activities. If the inequality $\lfloor\sigma(k)/(m+1)\rfloor\neq\lfloor\sigma(k+1)/(m+1)\rfloor$ holds, the robotic activities, $\sigma(k)$ and $\sigma(k+1)$, are exchanged. New solution $S'=(\sigma(0),\sigma(1),\cdots,\sigma(k-1),\sigma(k+1),\sigma(k),\cdots,\sigma(n(m+1)-1))$ is obtained and is feasible. For improving convergence, when the robotic activities are exchanged, the job of smaller processing time is moved from right to left. According to the following steps, the neighborhood structure of adjacent exchange is realized.

Step 1. The place, called k, is randomly selected in feasible solution. There is a robotic activity $\sigma(k)$ at place k.

Step 2. If the inequality $\lfloor\sigma(k)/(m+1)\rfloor\neq\lfloor\sigma(k+1)/(m+1)\rfloor$ holds, Step 3 is implemented. Otherwise, Step 4 is implemented.

Step 3. If the inequality $l_{[\sigma(k+1)],\lfloor\sigma(k+1)/(m+1)\rfloor+1} < l_{[\sigma(k)],\lfloor\sigma(k)/(m+1)\rfloor+1}$ is true, the robotic activities, $\sigma(k)$ and $\sigma(k+1)$, are exchanged. Otherwise, Step 4 is implemented.

Step 4. If the inequality $\lfloor\sigma(k)/(m+1)\rfloor\neq\lfloor\sigma(k-1)/(m+1)\rfloor$ holds, Step 5 is implemented. Otherwise, Step 1 is implemented.

Step 5. If the inequality $l_{[\sigma(k-1)],\lfloor\sigma(k-1)/(m+1)\rfloor+1} > l_{[\sigma(k)],\lfloor\sigma(k)/(m+1)\rfloor+1}$ is true, the robotic activities, $\sigma(k)$ and $\sigma(k-1)$, are exchanged. Otherwise, Step 1 is implemented.

Example 3. A feasible solution is shown in Figure 3(a). The place, $k=6$, is selected, and there are two robotic activities $\sigma(6)=4$, $\sigma(7)=6$ at place $k=6$ and $k+1=7$, respectively. The inequality $\lfloor\sigma(6)/(4+1)\rfloor\neq\lfloor\sigma(6+1)/(4+1)\rfloor$ is true, and the inequality $l_{[\sigma(6+1)],\lfloor\sigma(6+1)/(4+1)\rfloor+1} < l_{[\sigma(6)],\lfloor\sigma(6)/(4+1)\rfloor+1}$ holds. After the robotic activities $\sigma(6)$and $\sigma(7)$ are exchanged, new solution which is shown in Figure 3(b) is obtained and is feasible.

4.3.2. Random Insertion. For exploring field, the neighborhood of random insertion is developed. Since there is only one tank T_{i+1} between successive two tanks T_i $(i=0,1,\cdots,m-1)$, or there is only one tank T_{i-1} between successive two tanks $T_i (i=1,\cdots,m)$, the neighborhood structure of random insertion which is constructed is listed as follows:

Table 1: Processing time.

Job	Tank			
	T_1	T_2	T_3	T_4
1	2	4	6	8
2	4	6	8	10
3	6	8	10	12
4	8	10	12	14

Table 2: Robotic activity.

Job	Tank				
	T_0	T_1	T_2	T_3	T_4
1	0	1	2	3	4
2	5	6	7	8	9
3	10	11	12	13	14
4	15	16	17	18	19

Step 1. The place, denoted k, is randomly selected in feasible solution.

Step 2. The tanks, $[\sigma(k)]+1$ and $[\sigma(k)]-1$, which are located in the right-most at the left of the place k are found. The tanks $[\sigma(k)]+1$ and $[\sigma(k)]-1$ correspond to the places k_1 and k_2, respectively.

Step 3. The tanks, $[\sigma(k)]+1$ and $[\sigma(k)]-1$, which are located in the left-most at the right of the place k are found. The tanks $[\sigma(k)]+1$ and $[\sigma(k)]-1$ correspond to the places $\bar{k}_1$ and $\bar{k}_2$, respectively.

Step 4. Let $k' = \max(k_1, k_2)$ and $k'' = \min(\bar{k}_1, \bar{k}_2)$.

Step 5. If $k' = k'' - 1$ is true, go to Step 1. Otherwise, Step 6 is implemented.

Step 6. A random number $\bar{k}$ is obtained between k' and k'', and $\bar{k} \neq k$ holds.

Step 7. The robotic activity $\sigma(k)$ is inserted at the place $\bar{k}$. The relative position of the remainder robot activities is unchanged.

Example 4. The value of k is 9, and $\sigma(9)=8$. The result is shown in Figure 4(a). According to Step 2, $k_1=6$, $\sigma(k_1)=4$, and $k_2=8$, $\sigma(k_2)=7$ are shown in Figure 4(b). According to Step 3, $\bar{k}_1=12$, $\sigma(\bar{k}_1)=9$, $\bar{k}_2=13$, and $\sigma(\bar{k}_2)=12$ are shown in Figure 4(c). Figure 4(d) shows the results of $k'=8$ and $k''=12$, and $k' \neq k''-1$ is true. Finally, the place $\bar{k}=11$ is selected that is shown in Figure 4(e). Figure 4(f) shows that the robotic activity $\sigma(k)=8$ is inserted at the place $\bar{k}=11$.

4.3.3. Job Exchange. Two neighborhood structures, neighborhood exchange and random insertion, change the robotic move sequence, but the job scheduling does not change. Therefore, to find a much better solution, the neighborhood of job exchange is investigated for changing job sequence. After the robotic activities of the job $\delta(k)$ and the robotic activities of the job $\delta(h)$ have corresponded to swap in feasible solution, new solution is obtained and is feasible. To enhance algorithm efficiency, the job that the due date is smaller is prior processed. The following steps show the process that job exchange is constructed.

Step 1. The two different jobs, $\delta(k)$ and $\delta(h)$, are randomly selected. Without loss of generality, the job $\delta(k)$ is processed before the job $\delta(h)$.

Step 2. The due dates of job $\delta(k)$ and $\delta(h)$ are $d_{\delta(k)}$ and $d_{\delta(h)}$, respectively.

Step 3. If $d_{\delta(k)} \geq d_{\delta(h)}$ holds, implement Step 4; otherwise, implement Step 1.

Step 4. The robotic activities of the job $\delta(k)$ and the robotic activities of the job $\delta(h)$ correspond to swap.

Example 5. Figure 5(a) shows the randomly selected two jobs, $\delta(1)$ (the shadow is the slash) and $\delta(2)$ (the shadow is the square), respectively. The due dates of job $\delta(1)$ and $\delta(2)$ are $d_{\delta(1)}=4$ and $d_{\delta(2)}=3$, respectively. Since $d_{\delta(1)} > d_{\delta(2)}$ is true, the robotic activities of the job $\delta(1)$ and the robotic activities of the job $\delta(2)$ corresponded to swap. New feasible solution is obtained and is shown in Figure 5(b).

As mentioned above, INEH-VNS flowchart is shown in Figure 6. The detailed procedure of INEH-VNS is illustrated as algorithm in Algorithm 1.

5. Computational Result

To test the performance of the proposed method, the computational experiments are shown in this section. Extensive experiments are conducted on a set of problems. All experiments are implemented using Microsoft Visual Studio 2010 and are run on an Intel(R) Core(TM)i5-4344 CPU@3.20 GHz and RAM 8.0 GB PC.

In order to evaluate the performance of the proposed method, we develop the other algorithms, namely, G-VNS and simulated annealing (SA), respectively. There are two differences between the G-VNS and the proposed method. One is all the due dates $d_j (j=1,2,\cdots,n)$ are arranged in descending order, and then the job sequence set J is obtained for generating the initial solution of the G-VNS; the other is the $d_{\delta(k)} \geq d_{\delta(h)}$ which is replaced by $\text{sum}_{\delta(k)} \geq \text{sum}_{\delta(h)}$ in Step 3 of Section 4.3.3, where $\text{sum}_{\delta(k)}$ is computed by equation (4). The initial solution of SA is generated by the INEH method, and the proposed neighborhood structures in this study are applied in the SA.

For single machine scheduling problem under JIT environment, the due dates obey the uniform distribution of $[P(1-H-R/2), P(1-H+R/2)]$, where P is the lower bound, H is the tardiness factor, and R is the due date

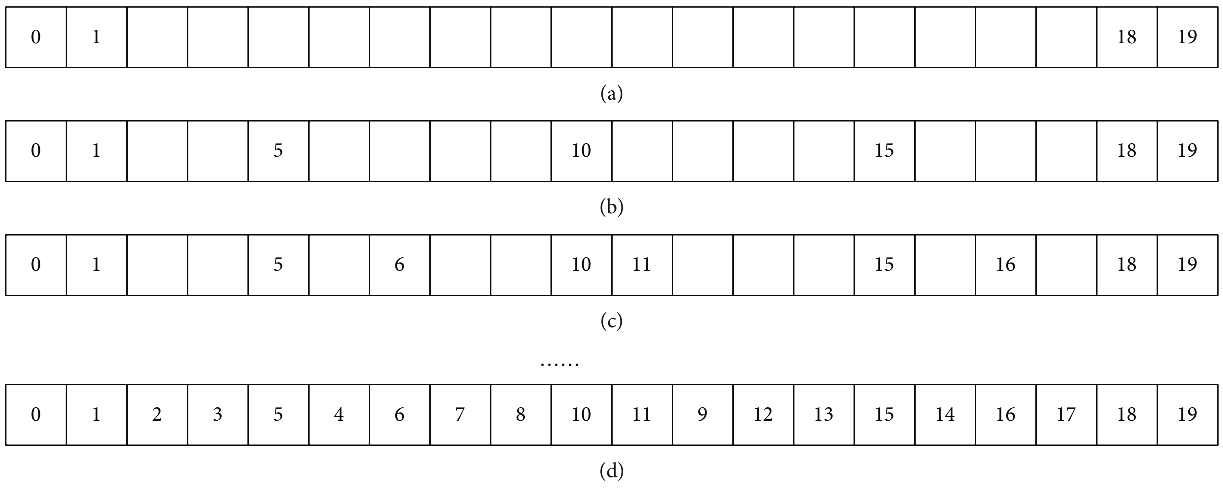

Figure 1: Procedure of producing initial solution.

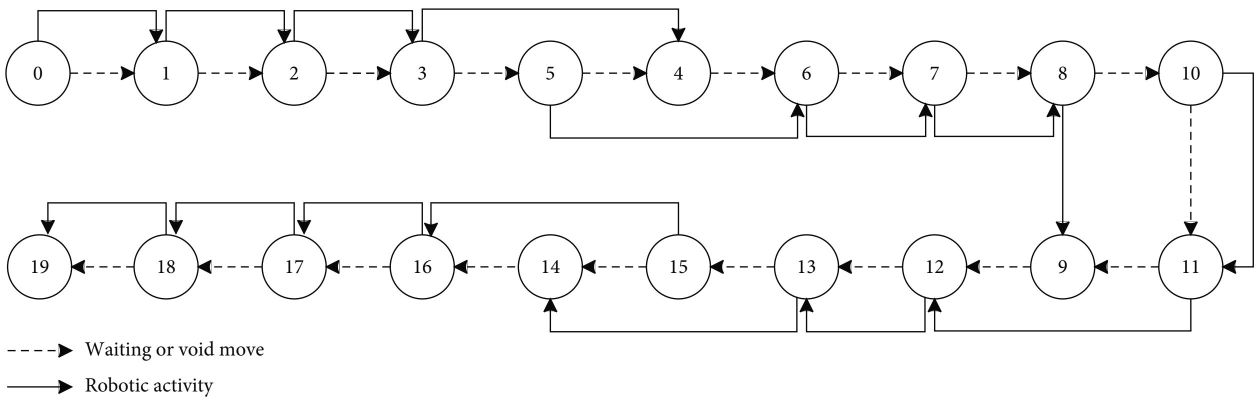

Figure 2: Robotic activity relationship of feasible solution.

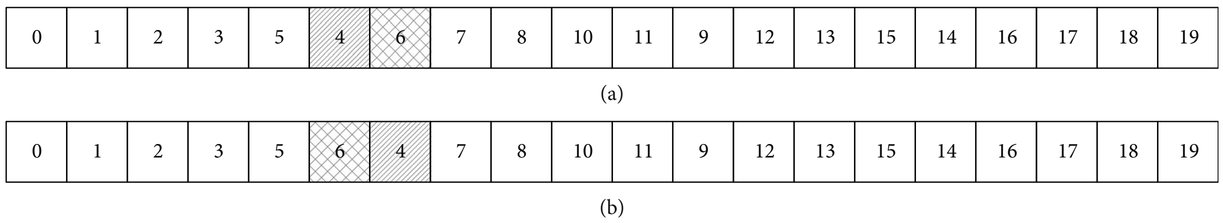

Figure 3: Schematic diagram of adjacent exchange.

range [45]. The computational method of the lower bound P is developed for m machine scheduling problem under JIT environment by Taillard [46]. The computational method which is proposed by Taillard does not consider the time of robotic activity. In this paper, to obtain a lower bound P, equation (3) is used. Then, the tardiness factor H and the due date range R are 0.2 and 0.6, respectively [47].

Since bench marks are not found for the proposed problem, one hundred instances are generated according to literature [48]. $l_{i,j}$ is an integer and $l_{i,j} \sim U[20, 99]$. The robotic move time $\mathrm{mt}_{i,j} = 6$ for all $i = 0, 1, 2, \cdots, m, j = 1, 2, \cdots, n$. The void move time $\mathrm{em}_{i,k} = 2|i - k|$ for all $i, k = 0, 1, 2, \cdots, m, m + 1$. The number of tanks m is an even number between 2 and 20. The number of jobs n varies between 5 and 50 and is an integer multiple of 5.

On the test of the instances, the proposed algorithm, called INEH-VNS, compares with the G-VNS, CPLEX12.5, and SA. Termination condition is the maximum computational time does not exceed 600 seconds. The other parameters of CPLEX 12.5 are default. Each instance is run 10 times, and the average of each instance, namely, $AVGE_i$ ($i \in \{\text{G-VNS, INEH-VNS, CPLEX, SA}\}$), is recorded. The performance measure is defined by equation (8), equation (9), and equation (10).

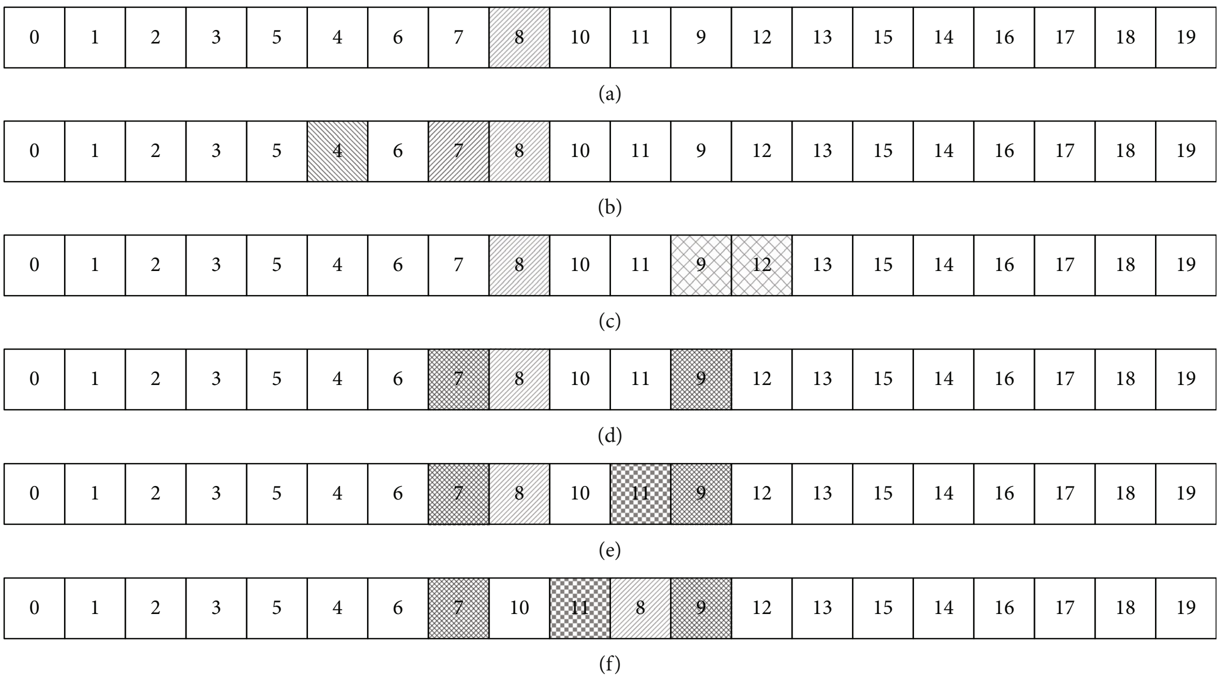

FIGURE 4: Schematic diagram of random insertion.

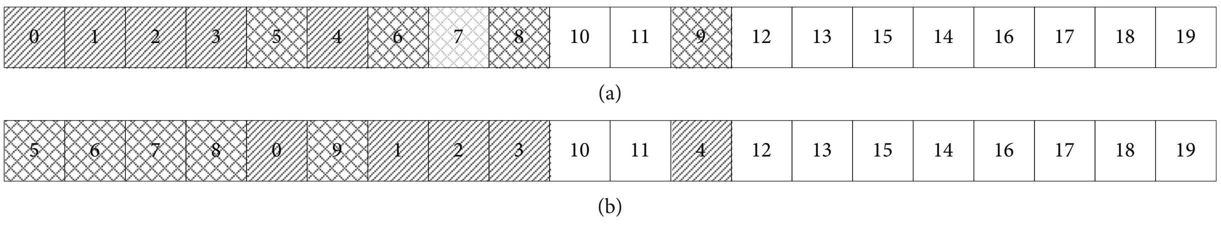

FIGURE 5: Schematic diagram of job exchange.

$$\begin{aligned} \mathrm{IR1} &= \frac{\mathrm{AVGE}_{\mathrm{CPLEX}}\text{-}\mathrm{AVGE}_{\text{INEH-VNS}}}{\mathrm{AVGE}_{\mathrm{CPLEX}}} \times 100\%, \\ \mathrm{IR2} &= \frac{\mathrm{AVGE}_{\mathrm{SA}}\text{-}\mathrm{AVGE}_{\text{INEH-VNS}}}{\mathrm{AVGE}_{\mathrm{SA}}} \times 100\%, \qquad (16) \\ \mathrm{IR3} &= \frac{\mathrm{AVGE}_{\text{G-VNS}}\text{-}\mathrm{AVGE}_{\text{INEH-VNS}}}{\mathrm{AVGE}_{\text{G-VNS}}} \times 100\%. \end{aligned}$$

5.1. Comparison Approximate Optimal Solution. The proposed algorithm compares with CPLEX 12.5, the G-VNS, and the SA, and the results are displayed in Tables 3–7. The results indicate high-quality solutions that can be found by the INEH-VNS.

The INEH-VNS algorithm compares with CPLEX 12.5, and the INEH-VNS algorithm is more effective for large-scale problems. The approximate optimal solutions of all instances are found by the INEH-VNS. Otherwise, 27% of the instances of the approximate optimal solutions are solved by CPLEX 12.5. Furthermore, in the instances in which the approximate optimal solutions are discovered by CPLEX 12.5, the approximate optimal solutions of eleven instances that are solved by INEH-VNS are better than the optimal solutions obtained by CPLEX 12.5. The best improvement rate is 77.96%, the worst improvement rate is 0.44%, and the average of $IR1$ is 34.55%. The best improvement rate and the worst improvement rate are shown in bold.

The INEH-VNS algorithm compares with the SA algorithm, and the INEH-VNS algorithm is more effective than the G-VNS algorithm. That is to say, if the run time is the same, better approximate solutions are found by the INEH-VNS algorithm in all instances. The best improvement rate is 77.78%, the worst improvement rate is 7.09%, and the average of $IR2$ is 54.18%. The best improvement rate and the worst improvement rate are exhibited by bold and shadow. The G-VNS algorithm compares with the SA algorithm, The G-VNS algorithm is better than the SA algorithm. 85 out of 100 instances of the solution quality which is found by the G-VNS are better than the solution quality which is solved by the SA algorithm. We find that if the number of tank is more than 4, the G-VNS algorithm is more competitive than the SA algorithm. The INEH-VNS algorithm and the G-VNS algorithm are more effective than the SA algorithm.

The INEH-VNS algorithm compares with the G-VNS algorithm, and the INEH-VNS algorithm is more

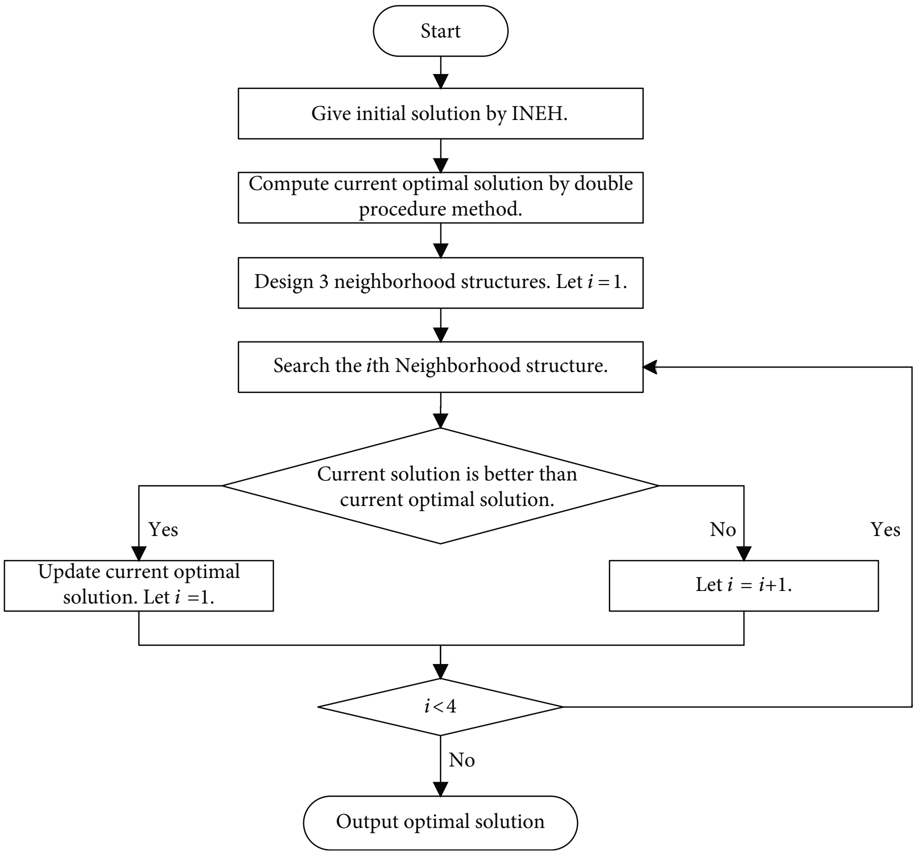
Start
Give initial solution by INEH.
Compute current optimal solution by double procedure method.
Design 3 neighborhood structures. Let $i = 1$.
Search the ith Neighborhood structure.
Current solution is better than current optimal solution.
Yes
No
Yes
Update current optimal solution. Let $i = 1$.
Let $i = i + 1$.
$i < 4$
No
Output optimal solution

FIGURE 6: INEH-VNS flowchart.

competitive than the G-VNS algorithm. The approximate optimal solutions of all instances are obtained by the INEH-VNS algorithm and the G-VNS algorithm, but 94 out of 100 instances of the solution quality which is found by the INEH-VNS are better than the solution quality which is solved by the G-VNS. The biggest *IR*3 is 69.86%, the smallest *IR*3 is 0.59%, and the average of *IR*3 is 45.9%. The biggest *IR*3 and the smallest *IR*3 are shown by shadow. The results show job exchange that is designed according to descending order due date is more effective.

5.2. Stability Comparison. In this subsection, the stability of the INEH-VNS algorithm, the G-VNS algorithm, and the SA algorithm is discussed. For evaluating algorithm stability, the standard deviation is used. Since CPLEX 12.5 can find 27% of the instances of the approximate optimal solutions, CPLEX 12.5 is not considered in this subsection. The results are shown in Tables 7–11. The stability rate, called *SR*1 and *SR*2, is computed according to equation (17) and equation (18), where STSA, STG-VNS, and STINEH-VNS represent the standard deviation of the SA algorithm, the G-VNS algorithm, and the INEH-VNS algorithm, respectively.

$$\mathrm{SR1} = \frac{\text{STSA-STINEH-VNS}}{\text{STSA}} \times 100\%, \tag{17}$$

$$\mathrm{SR2} = \frac{\text{STG-VNS-STINEH-VNS}}{\text{STG-VNS}} \times 100\%. \tag{18}$$

In Tables 8–12, the stability of the INEH-VNS algorithm and the G-VNS algorithm is much better than the stability of the SA algorithm. The stability of the INEH-VNS algorithm is much better than the stability of the G-VNS algorithm except for six instances. In the remainder instances, the standard deviation of the G-VNS algorithm is more than the standard deviation of the INEH-VNS. When $n = 5$ and $m = 6$ or $m = 18$, the same solution is obtained by the INEH-VNS algorithm; thus, the stability of the INEH-VNS is the best. The results are shown in bold. The standard deviation of the INEH-VNS algorithm, compared with the standard deviation of the G-VNS algorithm, is the worst improvement.

The rate is 3.46%, and the result is shown by shadow. That is to say, when $n = 20$ and $m = 4$, there is little difference for the stability of the INEH-VNS algorithm and the G-VNS algorithm. In these instances where the standard deviation of the INEH-VNS algorithm is more than the standard deviation of the G-VNS, the average of *SR*2 is 75.04%. These results show that job exchange that is designed according to descending order due date is more stable; that is to say, the INEH-VNS is better convergent.

5.3. Convergence. Because CPLEX 12.5 can find 27% of the instances of the approximate optimal solutions and the performance of the SA algorithm is poor, in this subsection, the convergence of the INEH-VNS algorithm and the G-VNS algorithm is described. When the number of

```
Algorithm INEH-VNS
Input: processing time of job, the time of robotic activity, the time of void move, the number of jobs $n$, the number of tanks $m$, the due date of job, the time of void move, neighborhood structures, $N_1$ (Adjacent exchange), $N_2$ (Random insertion), and $N_3$ (Job exchange). Let $k = 1, 2, 3$.
Output: optimal solution $S$, and optimal objective value $TT$.
//First stage: generate initiation solution by INEH.
For $i$=0 to $m$
    For $j$=0 to $n$
        Record the amend processing time of job as $l'_{i+1,j}$
    End for
End for
For $i$=0 to $m$
    For $j$=0 to $n$
        Calculate the sum of processing time of each job in all tanks as $sum_j$
    End for
End for
For $i$=0 to $m$
    For $j$=0 to $n$
        Descending order $sum_j$, obtain the job sequence set $J$.
    End for
End for
For $i$=0 to $n-1$
    For $j$=$i+1$ to $n$
        According to property 1, initiation solution $S_0$ is obtained.
    End for
End for
The double procedure method is proposed, the objective value $TT_0$ is computed. Let $S = S_0$ and $TT = TT_0$.
//Second stage: find optimal solution by VNS.
Repeat
    For $k = 1_{\text{to } 3}$
        Find the best neighbor $S'$ of $S$ in $N_{k;}$
        When the solution is $S'$, the objective value is $TT^{`}$;
        If $TT^{`} < TT$, then $S = S^{`}$ and $k = 1$;
        Otherwise $k = k + 1$;
    End for
Until Stopping criteria.
```

Algorithm 1: The pseudocode of the INEH-VNS.

Table 3: The result of the comparison between the INEH-VNS and CPLEX 12.5, the G-VNS, and the SA for $n = 5$ and $n = 10$.

m	5							10						
	CPLEX	SA	G-VNS	INEH-VNS	IR1(%)	IR2(%)	IR3(%)	CPLEX	SA	G-VNS	INEH-VNS	IR1(%)	IR2(%)	IR3(%)
2	493	635	439	590	-19.68	**7.09**	-34.40	925	1681	1186	1179	-27.46	29.85	0.59
4	1091	1488	1091	1214	-11.27	18.40	-11.27	1883	3406	2443	2551	-35.48	25.10	-4.42
6	1063	1854	1125	1305	-22.77	29.63	-16.00	2526	3815	3211	1933	23.48	49.33	39.80
8	1463	2611	1624	1803	-23.24	30.94	-11.02	4882	7943	4946	3322	31.95	58.18	32.83
10	1780	3077	2094	2144	-20.45	30.32	-2.39	6523	11452	7100	5139	21.22	55.12	27.62
12	1978	4435	2493	2172	-9.81	51.02	12.88	—	16925	10206	7215	—	57.37	29.31
14	2499	6188	3381	3236	-29.49	47.71	4.29	—	20124	12297	7886	—	60.81	35.87
16	1436	5246	2689	2158	-50.28	58.86	19.75	—	22493	14487	8077	—	64.09	44.25
18	2841	7008	4302	3549	-24.92	49.36	17.50	—	32611	20270	11912	—	63.47	41.23
20	3720	9997	5325	4663	-25.35	53.35	12.43	—	33530	22108	11689	—	65.14	47.13

The symbol bold' indicates the best improvement rate and the worst improvement rate.

Table 4: The result of the comparison between the INEH-VNS and CPLEX 12.5, the G-VNS, and the SA for $n = 15$ and $n = 20$.

m	15							20						
	CPLEX	SA	G-VNS	INEH-VNS	IR1(%)	IR2(%)	IR3(%)	CPLEX	SA	G-VNS	INEH-VNS	IR1(%)	IR2(%)	IR3(%)
2	1867	3334	2973	2331	-24.85	30.08	21.59	4365	6463	6462	4950	-13.40	23.41	23.40
4	7451	7169	6328	4694	37.00	34.52	25.82	13103	9114	10063	6525	50.20	28.40	35.16
6	7703	8805	8294	5335	30.74	39.41	35.68	—	12299	14348	6048	—	50.82	57.85
8	11659	18178	11604	7231	37.98	60.22	37.69	—	32291	21171	10649	—	67.02	49.70
10	—	24791	15983	9276	—	62.58	41.96	—	44868	29839	16187	—	63.92	45.75
12	—	34570	20709	12069	—	65.09	41.72	—	59627	39092	23384	—	60.78	40.18
14	—	42426	27015	16720	—	60.59	38.11	—	69337	46135	24621	—	64.49	46.63
16	—	52705	35570	22062	—	58.14	37.98	—	86514	58817	33019	—	61.83	43.86
18	—	62078	42921	23315	—	62.44	45.68	—	104738	76923	43351	—	58.61	43.64
20	—	70017	50865	26879	—	61.61	47.16	—	122469	91142	49460	—	59.61	45.73

Table 5: The result of the comparison between the INEH-VNS and CPLEX 12.5, the G-VNS, and the SA for $n = 25$ and $n = 30$.

m	25							30						
	CPLEX	SA	G-VNS	INEH-VNS	IR1(%)	IR2(%)	IR3(%)	CPLEX	SA	G-VNS	INEH-VNS	IR1(%)	IR2(%)	IR3(%)
2	6744	9846	11394	8103	-20.15	17.70	28.88	10112	12305	15603	10068	**0.44**	18.18	35.47
4	36771	13595	17490	8103	**77.96**	40.40	53.67	—	18518	24649	12287	—	33.65	50.15
6	—	23188	21934	9638	—	58.44	56.06	—	28782	34238	12232	—	57.50	64.27
8	—	47217	33205	12605	—	73.30	62.04	—	61529	45596	14232	—	76.87	68.79
10	—	68084	45098	23307	—	65.77	48.32	—	95193	64206	30880	—	67.56	51.90
12	—	98962	59518	32134	—	67.53	46.01	—	121323	82199	40165	—	66.89	51.14
14	—	105430	71535	35534	—	66.30	50.33	—	150899	105295	50646	—	66.44	51.90
16	—	135521	93677	48914	—	63.91	47.78	—	186305	135841	72702	—	60.98	46.48
18	—	158791	117724	66081	—	58.38	43.87	—	220702	168404	88466	—	59.92	47.47
20	—	181473	140627	75785	—	58.24	46.11	—	253343	202082	104241	—	58.85	48.42

The symbol bold' indicates the best improvement rate and the worst improvement rate.

Table 6: The result of the comparison between the INEH-VNS and CPLEX 12.5, the G-VNS, and the SA for $n = 35$ and $n = 40$.

m	35							40						
	CPLEX	SA	G-VNS	INEH-VNS	IR1(%)	IR2(%)	IR3(%)	CPLEX	SA	G-VNS	INEH-VNS	IR1(%)	IR2(%)	IR3(%)
2	13006	17059	22483	14108	-8.47	17.30	37.25	26736	25286	33131	22323	16.51	11.72	32.62
4	—	21931	34534	16376	—	25.33	52.58	—	30620	46964	24170	—	21.06	48.54
6	—	46902	49152	17002	—	63.75	65.41	—	65436	64193	22612	—	65.44	64.77
8	—	93408	65086	22189	—	76.25	65.91	—	109027	80392	24229	—	**77.78**	69.86
10	—	127233	87192	36513	—	71.30	58.12	—	168067	115275	44716	—	73.39	61.21
12	—	165049	113481	52317	—	68.30	53.90	—	211854	148882	68293	—	67.76	54.13
14	—	196886	144152	71225	—	63.82	50.59	—	259912	191018	92834	—	64.28	51.40
16	—	232382	176042	83334	—	64.14	52.66	—	310703	237617	118029	—	62.01	50.33
18	—	295100	222768	115001	—	61.03	48.38	—	377706	295988	149591	—	60.39	49.46
20	—	336981	268031	135693	—	59.73	49.37	—	431811	347982	176065	—	59.23	49.40

The symbol bold' indicates the best improvement rate and the worst improvement rate.

tanks is 50, and the number of jobs is 10, 12, 14, 16, 18, and 20, respectively, the convergence of the INEH-VNS algorithm and the G-VNS algorithm is shown in Figures 7(a)–7(f). In Figure 7, the INEH-VNS algorithm and the G-VNS algorithm cause a quick decrease in the objective function value at the beginning and continue reducing this function with lower speed until it converges to its best solution with a reduced speed of approximately zero. But the best solution which is found by the INEH-VNS algorithm is much better than the best solution which is found by the G-VNS algorithm. That is to say, the proposed algorithm is effective.

Table 7: The result of the comparison between the INEH-VNS and CPLEX 12.5, the G-VNS, and the SA for $n = 45$ and $n = 50$.

m	45							50						
	CPLEX	SA	G-VNS	INEH-VNS	IR1(%)	IR2(%)	IR3(%)	CPLEX	SA	G-VNS	INEH-VNS	IR1(%)	IR2(%)	IR3(%)
2	51314	27826	40584	24157	52.92	13.18	40.48	—	37055	53160	29196	—	21.21	45.08
4	—	37668	59833	26187	—	30.48	56.23	—	49054	78949	39258	—	19.97	50.27
6	—	90543	82649	30006	—	66.86	63.69	—	110265	103603	41566	—	62.30	59.88
8	—	142668	103064	34014	—	76.16	67.00	—	173608	129906	39274	—	77.38	69.77
10	—	203423	144021	55515	—	72.71	61.45	—	244018	172848	59123	—	75.77	65.79
12	—	256067	185770	79818	—	68.83	57.03	—	318390	234713	98828	—	68.96	57.89
14	—	324433	232567	99217	—	69.42	57.34	—	396303	294291	134595	—	66.04	54.26
16	—	370196	282836	126261	—	65.89	55.36	—	465536	361976	165256	—	64.50	54.35
18	—	463879	364988	178594	—	61.50	51.07	—	551850	438044	207390	—	62.42	52.66
20	—	519073	427214	199962	—	61.48	53.19	—	657109	537185	264502	—	59.75	50.76

Table 8: Comparison standard deviation between the INEH-VNS algorithm, the G-VNS algorithm, and the SA algorithm when $n = 5$ and $n = 10$.

Tank	5					10				
	STSA	STG-VNS	STINEH-VNS	SR1(%)	SR2(%)	STSA	STG-VNS	STINEH-VNS	SR1(%)	SR2(%)
2	96.74	0.00	5.10	94.73	—	293.99	18.80	87.20	70.34	**-363.73**
4	194.45	0.00	7.04	96.38	—	449.42	104.29	84.29	81.24	19.17
6	209.63	14.71	0.00	100.00	100.00	945.32	120.58	3.43	99.64	97.16
8	293.84	34.58	4.00	98.64	88.43	883.43	141.86	82.66	90.64	41.73
10	516.10	50.94	5.37	98.96	89.45	1573.53	171.76	244.73	84.45	**-42.48**
12	756.24	68.63	13.03	98.28	81.01	1695.16	282.55	80.25	95.27	71.60
14	912.01	104.50	2.72	99.70	97.40	2385.22	302.74	105.86	95.56	65.03
16	644.08	79.07	71.60	88.88	9.44	1601.24	324.34	71.04	95.56	78.10
18	893.98	128.16	0.00	100.00	100.00	2005.54	290.72	44.44	97.78	84.71
20	605.46	162.93	22.58	96.27	86.14	2134.83	456.86	57.60	97.30	87.39

The symbol bold' indicates the best improvement rate and the worst improvement rate.

Table 9: Comparison standard deviation between the INEH-VNS algorithm, the G-VNS algorithm, and the SA algorithm when $n = 15$ and $n = 20$.

Tank	15					20				
	STSA	STG-VNS	STINEH-VNS	SR1(%)	SR2(%)	STSA	STG-VNS	STINEH-VNS	SR1(%)	SR2(%)
2	190.55	91.40	153.56	19.41	**-68.01**	341.35	163.34	45.46	86.68	72.17
4	1067.12	207.40	121.74	88.59	41.30	1037.31	264.67	255.50	75.37	3.46
6	1757.07	280.04	21.84	98.76	92.20	1687.32	569.50	8.34	99.51	98.53
8	2515.67	472.49	162.06	93.56	65.70	2422.99	636.79	144.54	94.03	77.30
10	1639.26	399.43	152.27	90.71	61.88	3144.65	743.05	203.85	93.52	72.57
12	2277.20	392.36	302.40	86.72	22.93	2789.35	628.26	145.73	94.78	76.80
14	2613.79	613.04	288.78	88.95	52.89	5319.97	1386.70	400.55	92.47	71.11
16	3802.75	622.98	240.98	93.66	61.32	4752.51	1023.85	19.95	99.58	98.05
18	3613.17	312.13	317.07	91.22	**-1.58**	3108.78	994.65	415.17	86.65	58.26
20	4172.77	615.93	138.41	96.68	77.53	4102.68	2193.45	379.71	90.74	82.69

The symbol bold' indicates the best improvement rate and the worst improvement rate.

Table 10: Comparison standard deviation between the INEH-VNS algorithm, the G-VNS algorithm, and the SA algorithm when $n = 25$ and $n = 30$.

Tank			25					30		
	STSA	STG-VNS	STINEH-VNS	SR1(%)	SR2(%)	STSA	STG-VNS	STINEH-VNS	SR1(%)	SR2(%)
2	543.68	293.57	25.77	95.26	91.22	699.23	366.14	111.56	84.05	69.53
4	1760.64	506.39	25.77	98.54	94.91	2241.29	960.99	422.20	81.16	56.07
6	4451.15	831.56	113.10	97.46	86.40	3995.50	452.35	64.38	98.39	85.77
8	3077.30	989.86	43.46	98.59	95.61	5126.49	1549.31	85.86	98.33	94.46
10	3922.83	1371.23	218.17	94.44	84.09	5546.11	1147.74	306.76	94.47	73.27
12	6034.24	1100.04	305.91	94.93	72.19	5229.98	1766.38	326.26	93.76	81.53
14	7276.99	900.98	120.48	98.34	86.63	7243.47	1828.94	327.07	95.48	82.12
16	7872.45	1591.45	377.41	95.21	76.29	7659.39	1815.99	949.75	87.60	47.70
18	7401.14	712.86	369.40	95.01	48.18	10273.96	1327.83	554.84	94.60	58.21
20	6099.01	2310.39	318.55	94.78	86.21	6201.25	2110.95	399.79	93.55	81.06

Table 11: Comparison standard deviation between the INEH-VNS algorithm, the G-VNS algorithm, and the SA algorithm when $n = 35$ and $n = 40$.

Tank			35					40		
	STSA	STG-VNS	STINEH-VNS	SR1(%)	SR2(%)	STSA	STG-VNS	STINEH-VNS	SR1(%)	SR2(%)
2	529.61	330.32	252.09	52.40	23.68	1378.74	493.97	271.28	80.32	45.08
4	1708.50	943.56	450.31	73.64	52.28	1878.89	1365.57	792.22	57.84	41.99
6	4659.21	1185.72	184.05	96.05	84.48	3779.64	2004.88	169.03	95.53	91.57
8	4679.38	1811.13	182.94	96.09	89.90	6302.77	3901.98	97.62	98.45	97.50
10	5455.54	1609.81	343.82	93.70	78.64	10878.84	1753.24	184.96	98.30	89.45
12	9451.82	1779.00	183.22	98.06	89.70	7263.92	2330.18	250.07	96.56	89.27
14	7831.62	1735.54	790.83	89.90	54.43	12433.64	4733.82	599.02	95.18	87.35
16	12810.56	2148.04	530.44	95.86	75.31	8770.79	1560.77	573.16	93.47	63.28
18	6587.30	2266.52	1267.52	80.76	44.08	11822.65	3134.82	268.54	97.73	91.43
20	7386.41	2372.08	428.91	94.19	81.92	12216.95	3018.54	901.23	92.62	70.14

Table 12: Comparison standard deviation between the INEH-VNS algorithm, the G-VNS algorithm, and the SA algorithm when $n = 45$ and $n = 50$.

Tank			45					50		
	STSA	STG-VNS	STINEH-VNS	SR1(%)	SR2(%)	STSA	STG-VNS	STINEH-VNS	SR1(%)	SR2(%)
2	888.95	576.72	300.87	66.15	47.83	1969.96	633.32	585.14	70.30	7.61
4	4153.02	2662.17	399.54	90.38	84.99	2497.47	2121.86	587.25	76.49	72.32
6	3911.70	2237.33	353.38	90.97	84.21	8525.72	2815.48	500.19	94.13	82.23
8	2732.91	3789.85	161.71	94.08	95.73	9229.39	3257.93	141.18	98.47	95.67
10	10637.93	4163.80	41.10	99.61	99.01	6375.13	3456.54	375.50	94.11	89.14
12	8225.88	4089.40	345.49	95.80	91.55	16666.92	3580.72	291.88	98.25	91.85
14	15088.96	3974.43	59.25	99.61	98.51	18254.07	3403.23	176.78	99.03	94.81
16	10226.13	3178.66	250.23	97.55	92.13	16266.27	3395.58	414.28	97.45	87.80
18	10488.95	5661.82	311.76	97.03	94.49	22244.65	4447.87	358.74	98.39	91.93
20	10474.91	7102.92	590.66	94.36	91.68	17813.45	6935.89	1116.90	93.73	83.90

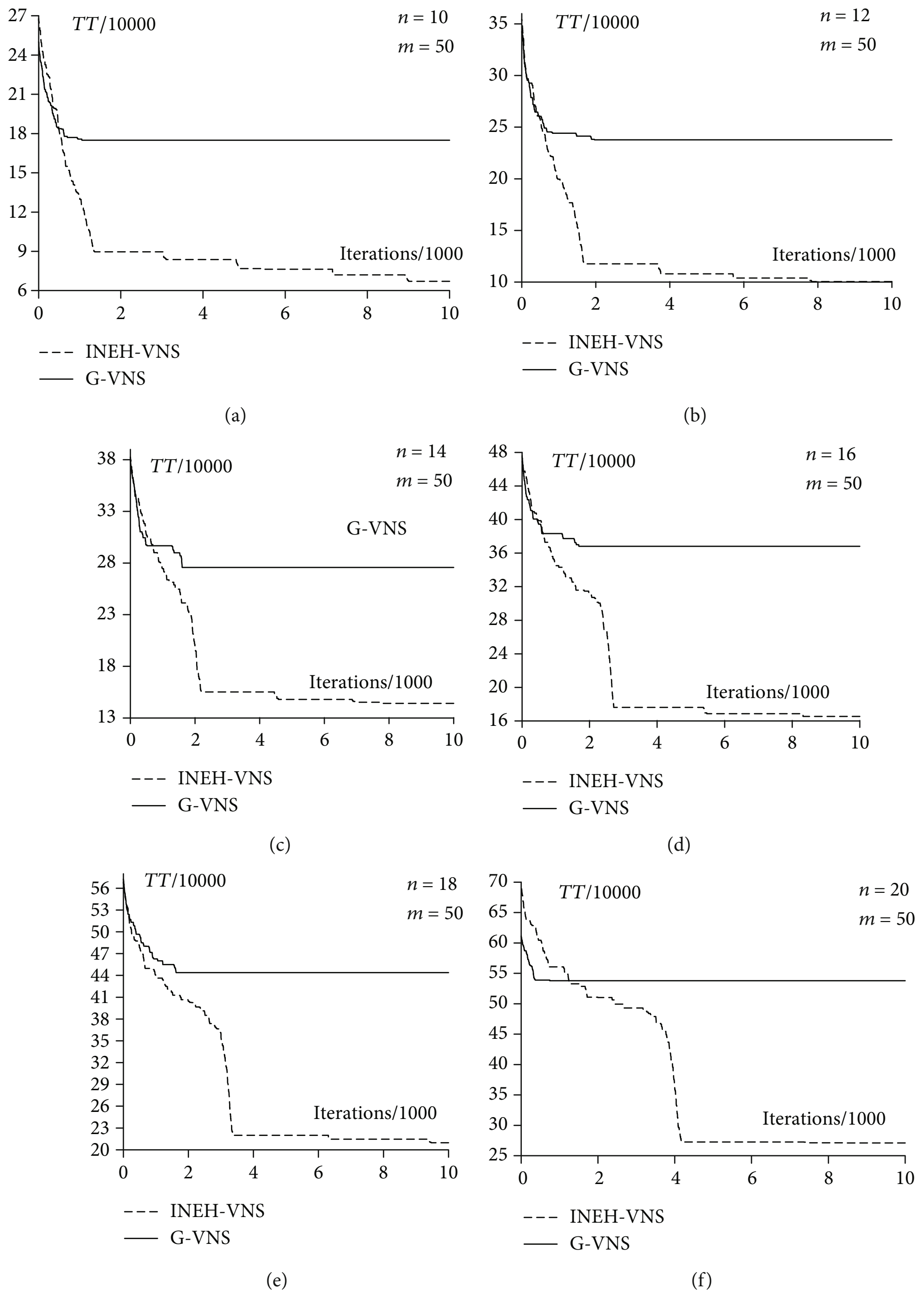

FIGURE 7: Convergence of the INEH-VNS algorithm and the G-VNS algorithm.

6. Conclusions

In this article, the automatic production system scheduling problem under just-in-time environment is investigated, and the INEH-VNS algorithm is developed for minimizing total earliness/tardiness time.

In the proposed algorithm, the improved NEH robotic activity method is investigated to obtain initial solution. The double procedure method is designed to compute the value of the objective function. According to the property of the current researched problem, three neighborhood structures, adjacent exchange, random insertion, and job exchange, are discussed for finding approximate optimal solution. Compared with CPLEX 12.5, the G-VNS algorithm, and the SA algorithm based on solving randomly generating instances, the proposed algorithm is outstanding. Next, we will research multiobjective automatic production system scheduling problems under just-in-time environment.

Appendix

Property 1. If $S = (t_{\sigma(0)}, t_{\sigma(1)}, \cdots, t_{\sigma(n(m+1)-2)}, t_{\sigma(n(m+1)-1)})$ is a solution of the proposed problem, $[\sigma(0)] = 0$, $[\sigma(1)] = 1$, $[\sigma(n(m+1)-2)] = m-1$, and $[\sigma(n(m+1)-1)] = m$, $\lfloor \sigma(0)/(m+1) \rfloor + 1 = \lfloor \sigma(1)/(m+1) \rfloor + 1$, $\lfloor \sigma(n(m+1)-2)/(m+1) \rfloor + 1 = \lfloor \sigma(n(m+1)-1)/(m+1) \rfloor + 1$ all hold.

Proof. Because at the beginning of the period, the input device is the full exclusion of all tanks and the output device. The job $\lfloor \sigma(0)/(m+1) \rfloor + 1$ is moved to the tank T_1; that is to say, the tank T_1 is occupied. According to condition (iii) of definition 1, the other job cannot be loaded into the tank T_1. The robot stays in the tank T_1. After the job $\lfloor \sigma(0)/(m+1) \rfloor + 1$ is processed completely on the tank T_1, the robot moved the job $\lfloor \sigma(0)/(m+1) \rfloor + 1$ from tank T_1 to tank T_2. So, $\lfloor \sigma(0)/(m+1) \rfloor + 1 = \lfloor \sigma(1)/(m+1) \rfloor + 1$, $[\sigma(0)] = 0$, and $[\sigma(1)] = 1$ are all true. All jobs are done, the output device is full, and all tanks and the input device are empty. The last two robotic activities of the solution must implement the job $[\sigma(n(m+1)-2)]$ from tank T_{m-1} to tank T_m, and the robot stays in the tank T_m; when the job is done on the tank T_m, the robot moves the job $[\sigma(n(m+1)-2)]$ from tank T_m to tank T_{m+1}. That is to say, $\lfloor \sigma(n(m+1)-2)/(m+1) \rfloor + 1 = \lfloor \sigma(n(m+1)-1)/(m+1) \rfloor + 1$, $[\sigma(n(m+1)-2)] = m-1$, and $[\sigma(n(m+1)-1)] = m$ all hold. □

Data Availability

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

Disclosure

The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Conflicts of Interest

The authors declare no conflict of interest.

Authors' Contributions

L.Q. and Z.X. were responsible for the conceptualization. H.Y. was responsible for the methodology. L.Q. was responsible for the software. Z.X. and H.Y. were responsible for the validation. Z.X. was responsible for the formal analysis. L.Q. was responsible for the investigation. H.Y. and Y. S. were responsible for the resources. H.Y. and Y. S. were responsible for the data curation. L.Q. and Y. S. were responsible for the writing—original draft preparation. Z.X. and H.Y. were responsible for the writing—review and editing. Z.X. and Y. S. were responsible for the visualization. Z.X. was responsible for the supervision. Z.X. and L.Q. were responsible for the project administration. Z. X and L.Q. were responsible for the funding acquisition. All authors have read and agreed to the published version of the manuscript.

Acknowledgments

This research was funded by the Humanities and Social Sciences Project of the Ministry of Education of the People's Republic of China under Grant No. 18YJCZH26. It was also supported by the Social Science Planning PhD Project of Chongqing under Grant No. 2019BS072, and the Science and Technology Research Program of Chongqing Municipal Education Commission under Grant Nos. KJQN201901337, KJQN202001316, and KJQN202101324.

References

[1] R. R. Fullerton and C. S. McWatters, "The production performance benefits from JIT implementation," *Journal of Operations Management*, vol. 19, no. 1, pp. 81–96, 2001.

[2] R. E. White and V. Prybutok, "The relationship between JIT practices and type of production system," *Omega*, vol. 29, no. 2, pp. 113–124, 2001.

[3] K. R. Baker and G. D. Scudder, "Sequencing with earliness and tardiness penalties: a review," *Operations Research*, vol. 38, no. 1, pp. 22–36, 1990.

[4] C. J. Liao and C. C. Cheng, "A variable neighborhood search for minimizing single machine weighted earliness and tardiness with common due date," *Computers & Industrial Engineering*, vol. 48, no. 4, pp. 404–413, 2007.

[5] P. Yan, G. Wang, A. Che, and Y. Li, "Hybrid discrete differential evolution algorithm for biobjective cyclic hoist scheduling with reentrance," *Computers & Operations Research*, vol. 76, no. 12, pp. 155–166, 2016.

[6] A. Amraoui and M. Elhafsi, "An efficient new heuristic for the hoist scheduling problem," *Computers & Operations Research*, vol. 67, no. 3, pp. 184–192, 2016.

[7] Q. Zhu, M. Zhou, Y. Qiao, and N. Wu, "Scheduling transient processes for time-constrained single-arm robotic multi-cluster tools," *IEEE Transactions on Semiconductor Manufacturing*, vol. 30, no. 3, pp. 261–269, 2017.

[8] D. Kim, H. Kim, and T. Lee, "Optimal scheduling for sequentially connected cluster tools with dual-armed robots and a single input and output module," *International Journal of Production Research*, vol. 55, no. 11, pp. 3092–3109, 2017.

[9] X. Zhao and X. Guo, "An effective chemical reaction optimization for cyclic multi-type parts robotic cell scheduling problem with blocking," *Journal of Intelligent & Fuzzy Systems*, vol. 35, no. 3, pp. 3567–3579, 2018.

[10] V. Kayvanfar, I. Mahdavi, and G. H. M. Komak, "A drastic hybrid heuristic algorithm to approach to JIT policy considering controllable processing times," *International Journal of Advanced Manufacturing Technology*, vol. 69, no. 1-4, pp. 257–267, 2013.

[11] V. Kayvanfar, G. H. M. Komaki, A. Alaei, and M. Zandieh, "Minimizing total tardiness and earliness on unrelated parallel machines with controllable processing times," *Computers & Operations Research*, vol. 41, no. 1, pp. 31–43, 2014.

[12] B. Alidaee, H. Li, H. Wang, and K. Womer, "Integer programming formulations in sequencing with total earliness and tardiness penalties, arbitrary due dates, and no idle time: a concise review and extension," *Omega*, vol. 103, article 102446, 2021.

[13] B. Choi and M. Park, "Single-machine scheduling with periodic due dates to minimize the total earliness and tardy

penalty," *Journal of Combinatorial Optimization*, vol. 41, no. 4, pp. 781–793, 2021.

[14] D. Woodruff and M. Spearman, "Sequencing and batching for two classes of jobs with deadlines and setups," *Production and Operations Management*, vol. 1, no. 1, pp. 87–102, 1992.

[15] B. J. Coleman, "Technical note: a simple model for Optimizing the single machine early/tardy problem with sequence-dependent setups," *Production and Operations Management*, vol. 1, no. 2, pp. 225–228, 1992.

[16] Y. W. Chen, Y. Z. Lu, M. Ge, G. K. Yang, and C. C. Pan, "Development of hybrid evolutionary algorithms for production scheduling of hot strip mill," *Computers & Operations Research.*, vol. 39, no. 2, pp. 339–349, 2012.

[17] K. Kianfar and G. Moslehi, "A branch-and-bound algorithm for single machine scheduling with quadratic earliness and tardiness penalties," *Computers & Operations Research*, vol. 39, no. 12, pp. 2978–2990, 2012.

[18] M. Vila and J. Pereira, "Exact and heuristic procedures for single machine scheduling with quadratic earliness and tardiness penalties," *Computers & Operations Research.*, vol. 40, no. 7, pp. 1819–1828, 2013.

[19] S. Muştu and T. Eren, "Minimization of the total weighted tardiness on a single machine scheduling problem with a position based learning effect and unequal release dates," *INFOR: Information Systems and Operational Research*, vol. 59, no. 2, pp. 353–376, 2021.

[20] G. Mosheiov, D. Oron, and D. Shabtay, "Minimizing total late work on a single machine with generalized due-dates," *European Journal of Operational Research*, vol. 293, no. 3, pp. 837–846, 2021.

[21] M. Gilenson and D. Shabtay, "Multi-scenario scheduling to maximise the weighted number of just-in-time jobs," *Journal of the Operational Research Society*, vol. 72, no. 8, pp. 1762–1779, 2021.

[22] J. Wang, B. Cui, P. Ji, and W. W. Liu, "Research on single-machine scheduling with position-dependent weights and past-sequence-dependent delivery times," *Journal of Combinatorial Optimization*, vol. 41, no. 2, pp. 290–303, 2021.

[23] C. F. Liaw, "A branch-and-bound algorithm for the single machine earliness and tardiness scheduling problem," *Computers & Operations Research*, vol. 26, no. 7, pp. 679–693, 1999.

[24] X. Geng, J. Wang, and D. Bai, "Common due date assignment scheduling for a no-wait flowshop with convex resource allocation and learning effect," *Engineering Optimization*, vol. 51, no. 8, pp. 1301–1323, 2019.

[25] H. Shi and J. Wang, "Research on common due window assignment flowshop scheduling with learning effect and resource allocation," *Engineering Optimization*, vol. 52, no. 4, pp. 669–686, 2020.

[26] X. Sun, X. Geng, J. Wang, and F. Liu, "Convex resource allocation scheduling in the no-wait flowshop with common flow allowance and learning effect," *International Journal of Production Research*, vol. 57, no. 6, pp. 1873–1891, 2019.

[27] S. Zhao, "Resource allocation flowshop scheduling with learning effect and slack due window assignment," *Journal of Industrial and Management Optimization*, vol. 17, no. 5, pp. 2817–2835, 2021.

[28] D. Lv and J. Wang, "Study on resource-dependent no-wait flow shop scheduling with different due-window assignment and learning effects," *Asia-Pacific Journal of Operational Research*, vol. 38, no. 6, pp. 145–154, 2021.

[29] B. Mor and G. Mosheiov, "Minmax due-date assignment on a two-machine flowshop," *Annals of Operations Research*, vol. 305, no. 1-2, pp. 191–209, 2021.

[30] C. Jiang, D. Zou, D. Bai, and J. B. Wang, "Proportionate flowshop scheduling with position dependent weights," *Engineering Optimization*, vol. 52, no. 1, pp. 37–52, 2020.

[31] D. Lv and J. Wang, "Study on proportionate flowshop scheduling with due-date assignment and position-dependent weights," *Optimization Letters*, vol. 15, no. 6, pp. 2311–2319, 2021.

[32] J. Koulamas Christos and K. George, "Flow shop scheduling with two distinct job due dates," *Computers & Industrial Engineering*, vol. 163, p. 107835, 2022.

[33] X. Zhao and X. Guo, "Branch and bound algorithm for solving hybrid flow shop robotic cells scheduling problem with blocking," *Journal of Computer Applications*, vol. 35, no. 3, pp. 3516–3529, 2018.

[34] X. Wu, Q. Yuan, and L. Wang, "Multiobjective differential evolution algorithm for solving robotic cell scheduling problem with batch-processing machines," *IEEE Transactions on Automation Science and Engineering*, vol. 18, no. 2, pp. 757–775, 2021.

[35] A. Majumder, D. Laha, and P. N. Suganthan, "Bacterial foraging optimization algorithm in robotic cells with sequence-dependent setup times," *Knowledge-Based Systems*, vol. 172, no. 1, pp. 104–122, 2019.

[36] A. Elmi, A. Nazari, and D. Thiruvady, "Cyclic flow shop robotic cell scheduling problem with multiple part types," *IEEE Transactions on Engineering Management*, vol. 69, no. 6, pp. 3240–3252, 2020.

[37] H. Wang, Z. Guan, C. Zhang, L. Yue, D. Luo, and S. Ullah, "The printed-circuit-board electroplating parallel-tank scheduling with hoist and group constraints using a hybrid guided tabu search algorithm," *IEEE Access*, vol. 7, pp. 61363–61377, 2019.

[38] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & Operations Research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[39] Y. Xu, S. Wandelt, and X. Sun, "Airline integrated robust scheduling with a variable neighborhood search based heuristic," *Transportation Research: Part B*, vol. 149, pp. 181–203, 2021.

[40] M. Kong, J. Xu, T. Zhang, S. Lu, C. Fang, and N. Mladenovic, "Energy-efficient rescheduling with time-of-use energy cost: application of variable neighborhood search algorithm," *Computers & Industrial Engineering*, vol. 156, article 107286, 2021.

[41] J. Wahiba, M. Eddaly, and J. Bassem, "Variable neighborhood search algorithms for the permutation flowshop scheduling problem with the preventive maintenance," *Operational Research*, vol. 21, no. 4, pp. 2525–2542, 2021.

[42] A. Anokić, Z. Stanimirović, T. Davidović, and Đ. Stakić, "Variable neighborhood search based approaches to a vehicle scheduling problem in agriculture," *International Transactions in Operational Research*, vol. 27, no. 1, pp. 26–56, 2020.

[43] C. Koulamas, "The total tardiness problem: review and extensions," *Operations Research*, vol. 42, no. 6, pp. 1025–1041, 1994.

[44] M. Nawaz, E. Enscore, and I. Ham, "A heuristic algorithm for the m -machine, n -job flow-shop sequencing problem," *Omega*, vol. 11, no. 1, pp. 91–95, 1983.

[45] C. Potts and L. Van Wassenhove, “A decomposition algorithm for the single machine total tardiness problem,” *Operations Research Letters*, vol. 1, no. 5, pp. 177–181, 1982.

[46] E. Taillard, “Benchmarks for basic scheduling problems,” *European Journal of Operational Research*, vol. 64, no. 2, pp. 278–285, 1993.

[47] D. Ronconi and L. Henriques, “Some heuristic algorithms for total tardiness minimization in a flowshop with blocking,” *Omega*, vol. 37, no. 2, pp. 272–281, 2009.

[48] H. Kamoun, N. Hall, and C. Sriskandarajah, “Scheduling in robotic cells: heuristics and cell design,” *Operational Research*, vol. 47, no. 6, pp. 821–835, 1999.