Hindawi
Scientific Programming
Volume 2020, Article ID 8811391, 14 pages
https://doi.org/10.1155/2020/8811391

Research Article

A DE-LS Metaheuristic Algorithm for Hybrid Flow-Shop Scheduling Problem considering Multiple Requirements of Customers

Yingjia Sun[1] **and Xin Qi**[2]

[1]*University of Science and Technology of China, Hefei 230009, China*
[2]*Baoshan District People's Government, Shanghai 201999, China*

Correspondence should be addressed to Yingjia Sun; sunyingjia@joyintech.com

Received 14 April 2020; Revised 5 May 2020; Accepted 3 June 2020; Published 15 July 2020

Academic Editor: Lu Zhen

Copyright © 2020 Yingjia Sun and Xin Qi. 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.

In this paper, we address a hybrid flow-shop scheduling problem with the objective of minimizing the makespan and the cost of delay. The concerned problem considers the diversity of the customers' requirements, which influences the procedures of the productions and increases the complexity of the problem. The features of the problem are inspired by the real-world situations, and the problem is formulated as a mixed-integer programming model in the paper. In order to tackle the concerned problem, a hybrid metaheuristic algorithm with Differential Evolution (DE) and Local Search (LS) (denoted by DE-LS) has been proposed in the paper. The differential evolution is a state-of-the-art metaheuristic algorithm which can solve complex optimization problem in an efficient way and has been applied in many fields, especially in flow-shop scheduling problem. Moreover, the study not only combines the DE and LS, but also modifies the mutation process and provides the novel initialization process and correction strategy of the approach. The proposed DE-LS has been compared with four variants of algorithms in order to justify the improvements of the proposed algorithm. Experimental results show that the superiority and robustness of the proposed algorithm have been verified.

1. Introduction

Flow shop involves a large amount of jobs and machines, and the jobs need to be processed in machines, which formulates a series of stages in the factories. Flow-shop scheduling problem (FSP) [1, 2] has been widely investigated by the researchers, which aims to find the optimal schedule in the production procedures. The FSP, with reference to many constraints of resources, the number of jobs and machines, is a complex combination optimization problem [3, 4]. The research on FSP can be traced back to 1954 [5], in which an exact algorithm has been proposed to deal with the proposed three-machine flow-shop scheduling problem. Then, many scholars have devoted enormous effort and time to the research of FSP. For example, Riahi and Kazemi [6] have proposed a no-wait flow-shop scheduling problem, which is a variant of FSP, considering the makespan and flow time. In the paper, the authors have provided the mathematical formulation and a hybrid metaheuristic algorithm combined ant colony optimization and simulated annealing algorithm. Marichelvam et al. [7] address a flow-shop scheduling problem with minimization of the makespan and total flow time, which is a biobjective flow-shop scheduling problem. In order to solve the proposed problem, a hybrid monkey search algorithm based on a subpopulation has been studied.

It is worth mentioning that hybrid flow-shop scheduling problem (HFSP) is one type of FSP, which need to schedule the jobs in machines and complete the assignments of machines [8–10]. In HFSP, there are parallel machines in a stage, which has higher requirements for the schedule Because redundancy and shortage of machines both can reduce

the efficiency of the production and increase the cost. In addition, hybrid flow-shop scheduling problem has been proved the NP-hard [11, 12]. For instance, Yu et al. [13] propose a HFSP to minimize the total tardiness in consideration of several practical assumptions. Moreover, a genetic algorithm has been utilized to tackle the proposed problem, which involves a novel decoding approach to get the objective. Pan et al. [14] provide nine algorithms to solve the proposed HFSP, and the objective of the concerned problem is to minimize the makespan. Zhang et al. [12] study the HSFP with a lot streaming, and the objective is to minimize the total flow time. A mathematical formulation is provided for the studied problem, and an effective modified birds optimization algorithm has been applied to obtain the results. In this paper, a hybrid flow-shop scheduling problem has been studied, which considers many stages with parallel machines and multiple requirements of customers. Different requirements of customers indicate that the manufacturer needs to produce different types of productions in a horizon. The diversity of productions reflects on the procedures in the manufacturer. For example, two productions need to be processed in different stages. The first one has two procedures and the other one has three procedures. The objective of the proposed problem is to minimize the makespan and the cost of delay. Thus, the schedule should not only consider the flow time but also take into account the time requirements of the customers, which is inspired by real-world situations.

Due to the NP-hard of the HFSP, in many prior studies, metaheuristic algorithms have been applied to tackle the HFSP [6, 11, 13], such as greedy algorithm, simulated annealing algorithm, genetic algorithm, etc. Metaheuristic algorithms can obtain the solution of complexity optimization problem in available time, which have been developed in many fields, i.e., manufacturing industry [15–17], airline industry [18–21], energy sources [22–25], etc. In addition, Differential Evolution (DE), firstly proposed by Storn and Price [26], is a powerful evolutionary algorithm for global optimization. Like most evolutionary algorithms, DE is a population-based algorithm involving three main operations [25, 27, 28], i.e., mutation operations, crossover operation, and selection operation. These operations update the solution by the specific mechanism and help the algorithm to find the global optimal solution. Recently, DE algorithm has been widely studied by many researchers and has shown good performance in solving many optimization problems, especially in flow-shop scheduling problem. For example, Zhang and Xing [29] have utilized DE to solve a distributed limited-buffer flow-shop scheduling problem, with minimization of the makespan. In the study, two heuristics and two differential evolution metaheuristics have been proposed for various situations. Zhou et al. [30] presented a hybrid differential evolution named DE-eEDA, which combined DE with an estimation of distribution algorithm. The authors have applied the proposed DE-eEDA to tackle the reentrant hybrid flow-shop scheduling problem with the objective of minimizing the total weighted completion time. In this paper, a hybrid differential evolution with local search algorithm (denoted by DE-LS) has been proposed to solve the studied problem, which has modified the mutation process and provides the novel initialization process and correction strategy of the approach. Moreover, the local search has been utilized in the specific situation which can improve the solution by several operators, i.e., 2-opt and swap. Moreover, the comparison between previous studies works and this study is provided in Table 1, which compares the studies from three aspects, i.e., types of problems, objectives, and approaches.

The contributions of our paper can be organized as follows: (1) A hybrid flow-shop scheduling problem with consideration of multiple requirements of customers has been considered, with reference to diverse productions and different procedures. (2) The novel mixed-integer programming model has been provided in the paper to formulate the studied problem. (3) The hybrid metaheuristic algorithm with Differential Evolution and Local Search (DE-LS) has been proposed to tackle the proposed problem, which involves the modified mutation process, initialization process, correction strategy, etc. (4) The proposed DE-LS has been compared with four variant algorithms, and the robustness and superiority of DE-LS have been verified.

The remaining components of this paper can be summarized as follows. The problem description and mathematical formulation are provided in Section 2. In Section 3, the proposed hybrid algorithm DE-LS has been illustrated detailedly, with the encoding strategy and correction strategy, initialization, etc. The results of computational experiments can be seen in Section 4. The conclusion of the paper is presented in Section 5.

2. Problem Description and Mathematical Formulation

The flow-shop scheduling problem has been proved as NP-hard [35]. Moreover, the hybrid flow-shop scheduling problem with two machines in one stage has also been proved NP-hard [36]. In this paper, a hybrid flow-shop scheduling problem with consideration of several machines in one working processing and more than one stage is being studied, which is NP-hard. This study focuses on the multiple demands of customers in the hybrid flow-shop scheduling problem, which is based on the real-world situations. Generally, the manufacturers will revive many bookings from different customers, which have different requirements about the productions. For example, some customers need the productions adding one or more processes based on the basic productions. On the contrary, some of them may need the productions omitting one or more processes based on the basic productions. The differences between customers directly influence the efficiency and the working patterns of the manufacturers. Thus, the paper considers the differential demands of the customers. Moreover, the paper aims to improve the efficiency of the manufacturers and reduce the delay cost. The notations used in the paper can be found in Table 2.

To better understand the proposed problem, an example of the processes of the manufacturing and the delivering is shown in Figure 1. Firstly, the suppliers provide the original

Table 1: Comparison between previous studies and this study.

References	Types of problems	Objectives	Approaches
Pan et al. [1]	Lot-streaming FSP	Minimize the total weighted earliness and tardiness penalties	Discrete artificial bee colony
Riahi and Kazemi [6]	No-wait FSP	Minimize the makespan	A hybrid ant colony optimization and simulated annealing algorithm
Marichelvam et al. [7]	FSP	Minimize the makespan and total flow time	A hybrid monkey search algorithm based on subpopulation
Ruize and Stützle [31]	Permutation FSP	Minimize the makespan	Iterated greedy algorithm
Xu et al. [32]	Permutation FSP	Minimize the makespan and total weighted tardiness	Iterated local search
Lei and Zheng [11]	HFSP	Minimize the total tardiness, maximum tardiness, and makespan	A novel neighborhood search with global exchange
Zhang et al. [12]	HFSP	Minimize the total flow time	Effective modified migrating birds optimization
Yu et al. [13]	HFSP	Minimize the total tardiness	Genetic algorithm
Pan et al. [14]	HFSP	Minimize the makespan	Nine effective metaheuristics
Tasgetiren et al. [33]	Blocking FSP	Minimize the makespan	Iterated greedy algorithm
Yagmahan and Yenisey [34]	FSP	Minimize the makespan and total flow time	Ant colony system algorithm
Zhang and Xing [29]	FSP	Minimize the makespan	Differential evolution
Zhou et al. [30]	HFSP	Minimize the total weighted completion time	A hybrid differential evolution with an estimation of distribution algorithm
This study	HFSP	Minimize the makespan and cost of delay	A hybrid differential evolution and local search

Table 2: Notations in the paper.

N	The total number of jobs
N	The index of jobs
C	The total number of customers
C	The index of customers
M	The total number of machines
m	The index of machines
J	The total number of working processes
j	The index of working processes
k	The number of production types
c_k	The demands number of customer c for production with type k
D_c	The latest delivery time of customer c
Q_{jm}	1, if the machine m in the j working process; 0, otherwise
P_{kjm}	The processing time of production with type k in the j working process in machine m
B_{njm}	The begin time of job n in the j working process in machine m
C_{njm}	The completion time of job n in the j working process in machine m
Max_m	The maximum number of jobs can be assigned to machine m in the same working process
ρ_c	The penalty cost of customer c if the latest delivery time is not satisfied

materials, and then the following processes can be started. Due to the different requirements of the customers, the jobs need to be processed with the specific sequences of the machines. For example, job 2 and job 3 need to be processed in the first process, while job 1 and job N do not need that. Jobs with different processing sequences can manufacture the productions with multiple types, which aims to satisfy the differential demands of customers.

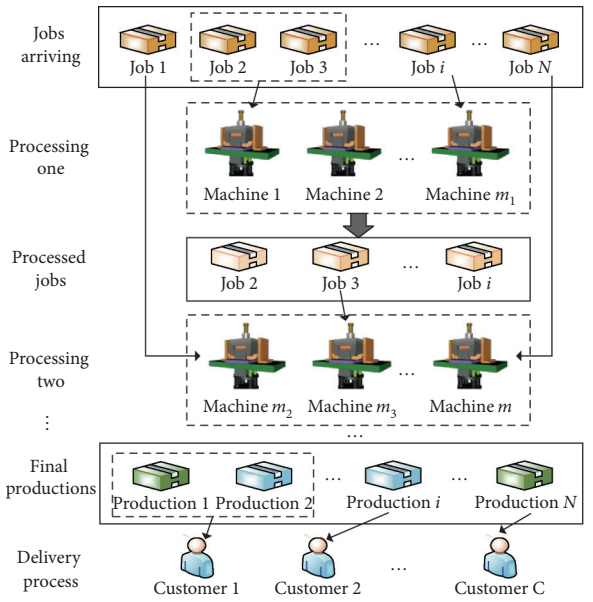


Figure 1: An example of the manufacturing and delivery.

In order to study the proposed problem, the following assumptions have been made in this paper:

(a) The job will not be interrupted when it starts

(b) The production with the same type is homogeneous

(c) The machine can process the jobs list with different types of jobs

(d) The arrival times of all jobs are known and the same.

Decision variables are as follows:

$$
\begin{aligned}
x_{njm} &= \begin{cases} 1, & \text{if the job } n \text{ is processed with machine } m \text{ in } j \text{ working process,} \\ 0, & \text{otherwise,} \end{cases} \\
y_{nk} &= \begin{cases} 1, & \text{if the job } n \text{ will be processed to production with type } k, \\ 0, & \text{otherwise,} \end{cases} \\
z_{nc} &= \begin{cases} 1, & \text{if the job } n \text{ is assined to customer } c, \\ 0, & \text{otherwise,} \end{cases} \\
\varphi_{inm} &= \begin{cases} 1, & \text{if the job } i \text{ is processed before the job } n \text{ in machine } m, \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{1}
$$

The MIP formulation is

$$\text{Min } C_{\max} + \sum_{c=1}^{C} \left(\sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{j=1}^{J} \max\left\{ C_{njm} - D_c, 0 \right\} z_{nc} \right) \rho_c, \tag{2}$$

subject to

$$C_{\max} = \max_{n \in \{1,2,\ldots,N\}} C_{njm}, \quad \forall j \in \{1,2,\ldots,J\}, \forall m \in \{1,2,\ldots,M\}, \tag{3}$$

$$\sum_{n=1}^{N} y_{nk} z_{nc} \geq c_k, \quad \forall c \in \{1,2,\ldots,C\}, \forall k \in \{1,2,\ldots,K\}, \tag{4}$$

$$\sum_{n=1}^{N} x_{njm} \leq \text{Max}_m, \quad \forall j \in \{1,2,\ldots,J\}, \forall m \in \{1,2,\ldots,M\}, \tag{5}$$

$$\sum_{m=1}^{M} x_{njm} \leq 1, \quad \forall n \in \{1,2,\ldots,N\}, \forall j \in \{1,2,\ldots,J\}, \tag{6}$$

$$\sum_{k=1}^{K} y_{nk} \leq 1, \quad \forall n \in \{1,2,\ldots,N\}, \tag{7}$$

$$\sum_{c=1}^{C} z_{nc} \leq 1, \quad \forall n \in \{1,2,\ldots,N\}, \tag{8}$$

$$C_{njm} = B_{njm} + \sum_{k=1}^{K} P_{kjm} y_{nk}, \quad \forall n \in \{1,2,\ldots,N\}, \forall j \in \{1,2,\ldots,J\}, \forall m \in \{1,2,\ldots,M\}, \tag{9}$$

$$C_{njm} \leq C_{\max}, \quad \forall n \in \{1,2,\ldots,N\}, \forall j \in \{1,2,\ldots,J\}, \forall m \in \{1,2,\ldots,M\}, \tag{10}$$

$$C_{ijm} \leq C_{njm} \varphi_{inm}, \quad \forall i,n \in \{1,2,\ldots,N\}, \forall j \in \{1,2,\ldots,J\}, \forall m \in \{1,2,\ldots,M\}, \tag{11}$$

$$x_{njm}, y_{nk}, z_{nc}, \varphi_{inm} \in \{0,1\}, \quad \forall i,n \in \{1,2,\ldots,N\}, \forall j \in \{1,2,\ldots,J\}, \forall m \in \{1,2,\ldots,M\}, \forall k \in \{1,2,\ldots,K\}, \forall c \in \{1,2,\ldots,C\}. \tag{12}$$

Objective (2) involves two phases, which are $C_{\max}$ and the penalty part. The definition of $C_{\max}$ is provided in (3). Equation (3) means that $C_{\max}$ is the maximum makespan in all jobs. The penalty part means that it will have an external cost if the production is delayed to each customer. Constraint (4) denotes that the demands of each customer will be satisfied. Constraint (5) means that the number of jobs assigned to one machine in a working process will not exceed the limitation of the machine. Constraint (6) limits that one job will not be scheduled to more than one machine in each working process. Constraint (7) denotes that only one type can be assigned to a job, and constraint (8) means one job

only can be assigned to one customer. Constraint (9) is the definition of the completion time, which can be calculated by the begin time of the job and the processing time. Constraint (10) means that the completion time of any job is less than $C_{\max}$. Constraint (11) defines that if two jobs are processed in one machine, the completion time of the latter one is larger than the former one. Four decision variables are defined in constraint (12).

3. The Proposed Hybrid Approach DE-LS

In this section, a hybrid metaheuristic algorithm DE-LS has been proposed, which involves a specific encoding strategy, correction strategy, initialization, etc., for the studied problem. In addition, the mutation strategy of the DE has been modified in order to improve the solution. The framework of the proposed DE-LS has also been provided in the section, which describes all processes and operation logic of the proposed algorithm.

3.1. Encoding Strategy and Correction Strategy. The difference of the productions lies in the processes in the manufacturing. For example, there are four kinds of the productions and three processes in the manufacturing totally. Production A needs to be processed by the first two processes, and production B is processed by the last two processes. The example of the relationship between productions and processes is provided in Figure 2.

Therefore, we define the procedures according to the different processes, and each job corresponds to a procedure. The processes in each procedure is defined before. Firstly, each job must be assigned to one kind of procedure, which will indicate one type of production. Thus, the number of each procedure is limited by the requirements of the customers. We provide an example of solution presentation in Figure 3. In the figure, there are seven jobs, and each job has a specific encoding number, which indicates the assigned procedure of the job. For example, the encoding number of job 3 and job 5 is 3, which means that these two jobs will be processed with procedure 3. The last line of the figure is the customers' requirements, which is known as the limitation of the encoding.

The solutions in this paper are integer and have a specific range. However, noninteger and out-of-range solutions will be generated during the processes of solutions computation. A correction strategy of solution is provided in the paper. Firstly, the range of the solution should be determined, which is related to the number of procedures. For example, there are four procedures totally, and each element in the solutions is integer in [1, 4]. Then, correct each element in the solution using round-off method. Moreover, the out-of-range elements are corrected according to the bound of the solution. Finally, count the number of procedures in the solutions and compare them with the customers' requirements. A simple example of the correction strategy is provided in Figure 4. The first line in the picture is the computed solution which is noninteger and did not meet the range of solution. After rounding the solution, a new solution is generated which is integer and met the range of solution. However, another limitation of the solution is the requirements of customers. The number of procedure 4 in the solution is more than the requirements of customers and the number of procedures 2 and 3 is less than the requirements of customers. Hence, randomly select two jobs with procedure 4 (i.e., job 4 and job 7) and change the procedure of them into 2 and 3, respectively. For a better understanding, we provide the pseudocode of the correction strategy in Algorithm 1.

3.2. Initialization. The initial solution should be generated before the solution changing, which is the available solution of the proposed problem. The initialization of the classical DE generated the initial solution using the minimum and maximum values of the solution. In this paper, the first step of the initialization is also to generate random variables for each solution, which are integer and between the range of the solution. Then, the limitation of the customers' requirements should be considered. The correction strategy can be used to make the solution available. The details of the initialization can be found in Algorithm 2.

3.3. Details of the Proposed DE-LS. The details of the proposed DE-LS can be seen in this section, which involves the mutation strategy, crossover, selection, and local search. The first three phases are the part of the classical DE, and the local search in the paper is to improve the searched solution.

3.3.1. Selective Mutation Strategy. Generally, many mutation operators are widely used in the DE, such as rand/1, rand/2, best/1, etc. The solution can be changed after using the mutation operations, and we call the new solution mutant solution, $V_i = \{v_{i1}, v_{i2}, \ldots, v_{ij}, \ldots, v_{i\,D}\}$. Some popular mutation operators can be found as follows:

$$\frac{\text{Rand}}{1}: V_i = X_{r_1} + F \times \left(X_{r_2} - X_{r_3}\right), \tag{13}$$

$$\begin{aligned}\frac{\text{Rand}}{2}: V_i &= X_{r_1} + F \times \left(X_{r_2} - X_{r_3}\right), V_i \\ &= X_{r_1} + F \times \left(X_{r_2} - X_{r_3}\right) + F \times \left(X_{r_4} - X_{r_5}\right),\end{aligned} \tag{14}$$

$$\frac{\text{Best}}{1}: V_i = X_{\text{best}} + F \times \left(X_{r_1} - X_{r_2}\right), \tag{15}$$

$$\frac{\text{Best}}{2}: V_i = X_{\text{best}} + F \times \left(X_{r_1} - X_{r_2}\right) + F \times \left(X_{r_3} - X_{r_4}\right), \tag{16}$$

$$\begin{aligned}\frac{\text{Current} - \text{to} - \text{best}}{1}: V_i &= X_i + F \times \left(X_{\text{best}} - X_i\right) \\ &\quad + F \times \left(X_{r_1} - X_{r_2}\right),\end{aligned} \tag{17}$$

where the integer variables $r_1, r_2, r_3, r_4, r_5, i$ are from [1, NP]. The variable i is determined, and other five variables are randomly generated. Moreover, they satisfy the inequation $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$. X_{best} is the current best solution. The variable F is randomly generated between 0 and 1.

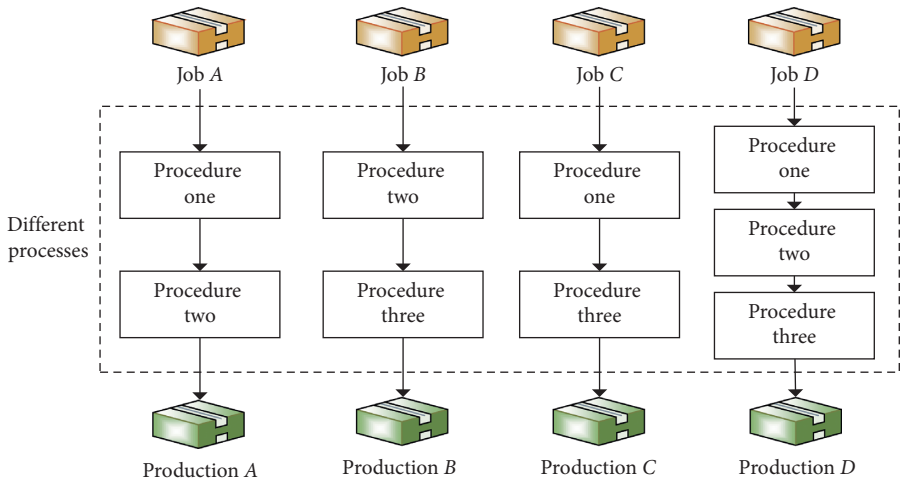
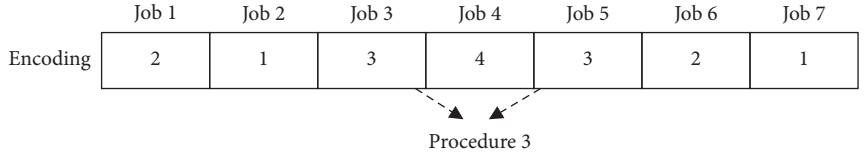


Figure 2: An example of the relationship between productions and processes.



Customers' requirements: production A: 2, production B: 2, production C: 2, and production D: 1

Figure 3: An example of solution presentation.

Customers' requirements: production A: 2, production B: 2, production C: 3, and production D: 1

Figure 4: An example of correction strategy.

In this paper, selective mutation strategy is proposed to update the solutions. For instance, if the best solution is updated in the last iteration, the current-to-best/1 operation will be applied in the mutation process. Otherwise, a random mechanism will be utilized to select the mutation operator. The details of the proposed selective mutation strategy are described as follows:

(1) If rand$(0, 1) < CR$ or $j == j_{\text{rand}}$ then
(2) If X_{best} is updated then
(3) Apply current-to-best/1: $V_i = X_i + F \times (X_{\text{best}} - X_i) + F \times (X_{r_1} - X_{r_2})$
(4) Else s = rand$(0, 1)$
(5) If $s < S_1$ then
(6) Apply rand/1: $V_i = X_{r_1} + F \times (X_{r_2} - X_{r_3})$
(7) Else
(8) Apply best/1: $V_i = X_{\text{best}} + F \times (X_{r_1} - X_{r_2})$
(9) End if
(10) End if
(11) End if

where the parameters CR and S_1 are random numbers between 0 and 1. The integer parameter j_{rand} is randomly generated between 1 and D.

3.3.2. Crossover. The crossover operation in the DE can generate a new solution vector, which is always called trial

```
Initialization: the solution x, the number of jobs J, the lower-bound of the solution Lower, the upper-bound of the solution Upper,
the list of customers' requirements CR, procedures count list count.
(1) While j ≤ J then
(2)   if x[j] < Lower or x[j] > Upper then
(3)     x[j] = random(Lower, Upper)
(4)   End if
(5)   x[j] = round(x[j])
(6)   j = j + 1
(7) End while
(8) While j ≤ J then
(9)   count[x[j]] = count[x[j]] + 1
(10)    j = j + 1
(11) End while
(12) *The following processes is used to correct the limitation of customers' requirements
(13) While i ≤ length(count) then
(14)   if count[i] < CR[i] then
(15)     For position tims = 1: (CR[i] − count[i])
(16)       index = False
(17)       While k ≤ length(count) and in de x = False then
(18)         if count[k] > CR[k] then
(19)           For position j = 1: J
(20)             if x[j] == k then
(21)               x[j] = i
(22)               count[i] = count[i] + 1
(23)               count[k] = count[k] − 1
(24)               index = True
(25)               break
(26)             End if
(27)           End for
(28)         Else k = k + 1
(29)       End while
(30)     End for
(31)   End if
(32)   i = i + 1
(33) End while
```

Algorithm 1: The pseudocode of the correction strategy.

```
Initialization: the population size NP, the dimension of the solution D, the lower-bound of the solution X^L = {x_1^L, x_2^L, ..., x_D^L}, the
upper-bound of the solution X^U = {x_1^U, x_2^U, ..., x_D^U}, the empty initial solution X = {X_1, X_2, ..., X_i, ..., X_NP}, and
X_i = {x_i1, x_i2, ..., x_ij, ..., x_i D}.
(1) For position i = 1: NP
(2)   For position j = 1: D
(3)     x_ij = x_j^L + rand(0, 1) × (x_j^U − x_j^L)
(4)     x_ij = round(x_ij)
(5)   End for
(6) End for
(7) Using the processes of correcting the limitation of customers' requirements which can be found in the Algorithm 1.
```

Algorithm 2: The processes of initialization.

vector. Each element in trial vector is selected from the corresponding element which may from solution before mutation or mutated solution. In this paper, binomial crossover operation is applied, and the details can be represented as follows:

$$u_{ij} = \begin{cases} v_{ij}, & \text{if rand}(0,1) \le CR \text{ or } j == j_{\text{rand}}, \\ x_{ij}, & \text{otherwise}, \end{cases} \tag{18}$$

where CR is a random number between 0 and 1, and the integer number j_{rand} is selected from 1 to D. The index $i \in \{1, 2, \ldots, NP\}$ means the individual of the solution. The equation indicates that when the random number is less than CR and $j == j_{\text{rand}}$, the element of mutated solution is selected for the trial solution. Otherwise, the element of solution before being mutated is selected for trial solution.

3.3.3. Selection. The selection is the last process of DE in an iteration. The selection mechanism in the paper is the same as that in most studies, which remains the better solution after the solution changing. The selection regulation can be found as follows:

$$X_i = \begin{cases} U_i, & \text{if } \text{obj}(U_i) \le \text{obj}(X_i), \\ X_i, & \text{otherwise}, \end{cases} \tag{19}$$

where $\text{obj}(U_i)$ and $\text{obj}(X_i)$ are the objective value of the solution U_i and X_i, respectively. The equation means that the solution with lower objective value can survive to the next generation.

3.3.4. Local Search. Local search (LS) has been widely applied to solve complex optimization problems, which can get the optimal solution in an available time. Moreover, local search is to find neighborhoods of the selected solution by using specific neighborhood structure operators, such as, 2-opt, exchange, swap, etc. In order to improve the efficiency of the proposed approach, local search has been used in this paper. The proposed local search is utilized after the selection process, which search the neighborhoods by the improved solutions. In addition, two neighborhood structure operations have been applied in the paper, i.e., 2-opt and swap. These two operations are very popular and simple. In Figure 5, the changing mechanisms of 2-opt and swap operations have been illustrated. For example, two jobs need to selected before the operations. In the example, job 2 and job 5 have been selected for the 2-opt operation, and job 1 and job 7 have been selected for the swap operation. Then, the 2-opt operation is to convert all jobs between the selected two jobs as seen in the figure. The swap operation is to exchange the value of the selected two jobs. We provide the pseudocode of the proposed local search, which can be found in Algorithm 3.

3.3.5. The Proposed DE-LS. In this section, the proposed DE-LS approach is described in Algorithm 4, which involves the basic processes of DE, i.e., mutation, crossover, and selection, and a proposed local search is also embedded.

4. Computational Experiments

In this section, the performance of the proposed algorithm will be evaluated by 18 randomly generated instances. The computational experiments are set with the number of jobs $n \in \{20, 30, 40, 50, 60, 70, 80, 90, 100\}$. We classify the number of jobs into two scales, i.e., the small scale and large scale. The small scale has number of jobs more than 50; otherwise, when the number of jobs is more than 50, it is the large scale. Moreover, there are two scales for the number of machines for each number of jobs, and we define each instance according to the number of jobs and machine. For example, the first instance is defined as J30-M3, which means the problem setting for this instance is 30 jobs and 3 machines. The processing time of production with each type in each working process is generated from a discrete uniform distribution. All experiments are implemented in Python and run on a PC with 8 GB of RAM memory, 64-bit operating system, and Intel® Core™ i7-6600U CPU @ 2.60 GHz 2.81 GHz.

The study proposes a DE-LS to tackle the concerned hybrid flow-shop scheduling problem, which combined the metaheuristic differential evolution algorithm with local search in order to improve the quality of the found solution and solve the study problem efficiently. Moreover, A selective mutation strategy of differential evolution has been provided, such as mutation strategy. Therefore, four algorithms have been compared with the proposed DE-LS, i.e., without LS, best/1, current-to-best/1, and rand/1, which are the variants of the proposed DE-LS. In other words, the without LS algorithm is the proposed DE-LS while without using local search. The other three algorithms, best/1, current-to-best/1, and rand/1, are different in their mutation processes and do not have the selected mechanism; for example, the best/1 algorithm only uses the best/1 mutation operation in the mutation process. In addition, each instance has been executed 10 times for avoiding the random results. The mutant factor F is randomly generated from $(F', 1)$, and the parameter F' is set as 0.4. The crossover control parameter CR is randomly generated from $(CR', 1)$, and the parameter CR' is set as 0.6. The population size NP is set as 10, and the dimensions of solution D is equal to the number of jobs. The local search selection is limited by the parameter S', which is set as 0.5.

The maximum number of iterations for each instance is 200, which is the terminal condition for each algorithm. The objective (denoted by Obj) of each algorithm in each instance is the average objective value of 10 runtimes. The minimum value (denoted by Min) is the minimum objective that the algorithm could find in 10 runtimes. The relative percentage deviation (denoted by RPD) is used as the performance measure in the paper which is widely used to evaluate the performance of algorithms [31, 33]. The RPD can be calculated by the given formulation as follows:

$$\text{RPD}_i = \frac{Z(\text{Objective}_i) - Z(\text{Minimum})}{Z(\text{Minimum})} * 100, \tag{20}$$

where $Z(\text{Objective}_i)$ is the average objective value of the algorithm i. $Z(\text{Minimum})$ denotes the minimum objective has been found in all the compared algorithms for the same instance. RPD_i is the RPD value of the specific algorithm i. Moreover, the standard deviation (denoted by SD) is also applied to measure the robustness of the compared algorithm, which is calculated as follows:

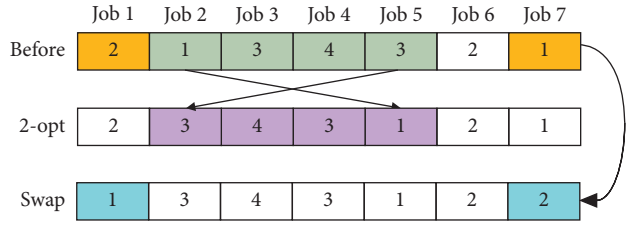


Figure 5: An example of 2-opt and swap operations.

Input: solution $X_i = \{x_1, x_2, \ldots, x_j, \ldots, x_D\}$, the objective value $\text{obj}(X_i)$, maximum number of iterations T, pre-defined index index $\in [0, 1]$.

(1) While $t < T$ then
(2) opt = rand (0, 1)
(3) j_1 = randint $(1, D-1)$
(4) j_2 = randint (r_1, D)
(5) If opt < index then
(6) Get the new solution X_i' by using 2-opt operation, and the selected jobs are j_1 and j_2
(7) Else
(8) Get the new solution X_i' by using swap operation, and the selected jobs are j_1 and j_2
(9) End if
(10) Calculate the objective value $\text{obj}(X_i')$
(11) If $\text{obj}(X_i) > \text{obj}(X_i')$ then
(12) Update $X_i = X_i'$
(13) End if
(14) $t = t + 1$
(15) End while

Algorithm 3: The proposed local search.

Initialization: the population size NP, the dimension of the solution D, the lower-bound of the solution $X^L = \{x_1^L, x_2^L, \ldots, x_D^L\}$, the upper-bound of the solution $X^U = \{x_1^U, x_2^U, \ldots, x_D^U\}$, the empty initial solution $X = \{X_1, X_2, \ldots, X_i, \ldots, X_{NP}\}$, and $X_i = \{x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{iD}\}$, maximum number of iterations $\text{Iter}^{\max}$, F', CR', and S' are the pre-defined parameters.

(1) Apply the initialization process to get the initial solution X and the objective value $\text{obj}(X)$
(2) Find the current best solution X_{best} and $\text{obj}(X_{\text{best}})$
(3) While iter < $\text{Iter}^{\max}$ then
(4) For position $i = 1$: NP then
(5) $F = \text{rand}(F', 1)$
(6) $CR = \text{rand}(CR', 1)$
(7) For position $j = 1$: D then
(8) *The processes of mutation
(9) If rand $(0, 1) < CR$ or $j == j_{\text{rand}}$ then
(10) Generate $r_1 \neq r_2 \neq r_3 \in \{1, 2, \ldots, NP\}$
(11) If X_{best} is updated then
(12) $u_{ij} = x_{ij} + F \times (x_{\text{best},j} - x_{ij}) + F \times (x_{r_1 j} - x_{r_2 j})$
(13) Else $s = \text{rand}(0, 1)$
(14) If $s < S'$ then
(15) $u_{ij} = x_{r_1 j} + F \times (x_{r_2 j} - x_{r_3, j})$
(16) Else
(17) $u_{ij} = x_{\text{best},j} + F \times (x_{r_1 j} - x_{r_2 j})$
(18) End if
(19) End if
(20) Else
(21) $u_{ij} = x_{ij}$
(22) End if
(23) End for
(24) Using correction strategy
(25) Get the objective value $\text{obj}(U_i)$

Algorithm 4: Continued.

```
(26)      If obj(U_i) ≤ obj(X_i) then
(27)         X_i = U_i
(28)      Else
(29)         Apply the proposed local search which can be found in Algorithm 3 to update the solution X_i
(30)      End if
(31)   End for
(32)   Search the new best solution and update the solution X_best and obj(X_best)
(33)   iter = iter + 1
(34) End while
```

Algorithm 4: The proposed DE-LS.

Table 3: Computational results about objective.

Instances	Types	DE-LS		Without LS		Best/1		Current-to-best/1		Rand/1	
		Obj	Min	Obj	Min	Obj	Min	Obj	Min	Obj	Min
1	J20-M3	**104.68**	**104.68**	105.25	**104.68**	105.44	**104.68**	105.15	**104.68**	104.79	**104.68**
2	J20-M4	**79.60**	**79.60**	80.47	**79.60**	81.17	**79.60**	80.11	**79.60**	79.90	**79.60**
3	J30-M3	**134.63**	**134.46**	137.30	134.99	137.90	134.99	136.51	**134.46**	136.14	134.98
4	J30-M4	**120.55**	**120.51**	121.53	120.53	123.46	122.37	121.71	120.56	120.88	**120.51**
5	J40-M3	**196.20**	**195.89**	198.10	196.51	201.96	198.00	199.40	196.59	198.28	196.70
6	J40-M4	**155.23**	**154.96**	157.66	155.62	158.81	156.54	156.12	155.28	155.89	**154.96**
7	J50-M3	**229.65**	**229.08**	230.94	229.05	232.62	230.61	231.07	229.25	230.16	229.21
8	J50-M4	**207.54**	**206.65**	209.93	207.63	213.70	210.03	208.81	207.32	208.56	207.11
9	J60-M3	**283.95**	**282.97**	286.76	284.36	289.78	286.25	285.62	284.11	286.50	283.53
10	J60-M4	**255.86**	**254.68**	257.96	255.77	265.17	261.83	261.88	256.21	260.33	257.66
11	J70-M3	**324.68**	**321.56**	329.79	325.87	334.77	328.82	328.82	325.60	332.79	330.23
12	J70-M4	**337.72**	**336.66**	343.81	340.52	351.49	345.41	346.60	343.06	345.64	342.46
13	J80-M3	**414.80**	**411.70**	423.99	418.48	431.32	424.06	423.16	419.02	429.11	420.77
14	J80-M4	**362.54**	**359.00**	367.54	364.58	370.72	365.28	369.02	361.06	372.58	364.22
15	J90-M3	**399.48**	**398.22**	400.91	399.24	403.39	399.32	400.82	398.44	405.90	402.07
16	J90-M4	**366.55**	**364.21**	373.06	367.38	379.74	374.52	375.61	372.49	384.60	377.17
17	J100-M3	**432.39**	**430.50**	434.83	433.20	439.23	437.19	435.94	430.04	443.80	438.62
18	J100-M4	**463.80**	**460.39**	470.86	465.35	477.93	467.42	471.54	463.94	477.54	471.25
Average		**270.55**	**269.21**	273.93	271.30	277.70	273.72	274.33	271.21	276.30	273.10

$$\mathrm{SD}_i = \sqrt{\frac{\sum_{r=1}^{\mathrm{Runtimes}} \left(Z\left(\mathrm{Objective}_{ir}\right) - Z\left(\mathrm{Minimum}\right)\right)^2}{\mathrm{Runtimes}}}, \tag{21}$$

where $Z(\mathrm{Objective}_r)$ means the objective value of algorithm i in rth runtimes, and then $Z(\mathrm{Minimum})$ denotes the minimum objective has been found in all the compared algorithms for the same instance. The standard deviation value of algorithm i is denoted by SD_i, which measures the deviation between the found optimal solution and the calculated solutions of algorithm i in all runtimes. The smaller the value of SD, the more robust the algorithm.

In Table 3, the average objective value and minimum objective value of each compared algorithm for each instance have been provided. We can find that the proposed DE-LS has found the optimal solution in all instances. Moreover, the average objectives of DE-LS in all instances are also better than the other four compared algorithms. In addition, we can find that when the number of jobs is 20 whatever the number of machines is, 3 or 4, the minimum values of all compared algorithms are the same. Moreover, the average objectives value of the proposed DE-LS is equal to the minimum value found, which means that the proposed algorithm can find the minimum value in all runtimes. Although other compared algorithms can find the minimum value, the robustness of other compared algorithms is worser than the proposed DE-LS. When the number of jobs is 30 and 40, and the number of machines is 4, the minimum objective value of Rand/1 algorithm is the same as that of the proposed algorithm. When the number of jobs is 30 and the number of machines is 4, the current-to-best/1 algorithm has found the minimum objective value as same as that of the proposed algorithm. With the increasing number of jobs, the deviation between the proposed DE-LS and the other four compared algorithms becomes obvious.

The RPD values and SD values of each algorithm for all instances have been provided in Table 4. It is easy to find that RPD and SD values of the proposed DE-LS are smaller than those of the other compared algorithms in all instances. In addition, the RPD values of the proposed DE-LS in all instances are no more than 1, and the SD values of the

Table 4: Computational results about RPD and SD.

Instances	Types	DE-LS		Without LS		Best/1		Current-to-best/1		Rand/1	
		RPD	SD	RPD	SD	RPD	SD	RPD	SD	RPD	SD
1	J20-M3	**0.00**	**0.00**	0.54	0.77	0.72	1.17	0.45	1.04	0.10	0.21
2	J20-M4	**0.00**	**0.00**	1.09	1.13	1.97	1.98	0.63	0.66	0.37	0.44
3	J30-M3	**0.12**	**0.25**	2.11	3.27	2.56	4.22	1.52	2.41	1.24	2.31
4	J30-M4	**0.03**	**0.10**	0.84	1.24	2.44	3.12	0.99	1.41	0.31	0.57
5	J40-M3	**0.16**	**0.49**	1.13	2.50	3.10	6.71	1.79	3.95	1.22	2.63
6	J40-M4	**0.17**	**0.46**	1.74	3.07	2.48	4.40	0.75	1.50	0.60	1.23
7	J50-M3	**0.26**	**0.73**	0.82	2.38	1.56	4.04	0.88	2.27	0.49	1.32
8	J50-M4	**0.43**	**1.13**	1.59	3.53	3.41	7.38	1.05	2.52	0.92	2.26
9	J60-M3	**0.35**	**1.10**	1.34	4.11	2.40	7.11	0.93	3.03	1.25	3.94
10	J60-M4	**0.46**	**1.52**	1.29	3.71	4.12	10.96	2.82	8.86	2.22	5.88
11	J70-M3	**0.97**	**3.74**	2.56	8.58	4.11	13.58	2.26	7.55	3.49	11.32
12	J70-M4	**0.31**	**1.46**	2.12	7.37	4.40	15.05	2.95	10.21	2.67	9.24
13	J80-M3	**0.75**	**3.42**	2.99	13.08	4.77	20.11	2.78	12.00	4.23	17.87
14	J80-M4	**0.99**	**4.61**	2.38	8.76	3.27	12.39	2.79	10.89	3.78	14.46
15	J90-M3	**0.31**	**1.43**	0.67	3.15	1.30	5.99	0.65	3.41	1.93	8.10
16	J90-M4	**0.64**	**3.10**	2.43	9.43	4.26	15.82	3.13	11.79	5.60	20.73
17	J100-M3	**0.55**	**2.61**	1.11	5.04	2.14	9.41	1.37	7.00	3.20	14.33
18	J100-M4	**0.74**	**4.20**	2.27	10.89	3.81	18.06	2.42	11.73	3.72	17.64
Average		**0.40**	**1.69**	1.61	5.11	2.93	8.97	1.68	5.68	2.07	7.47

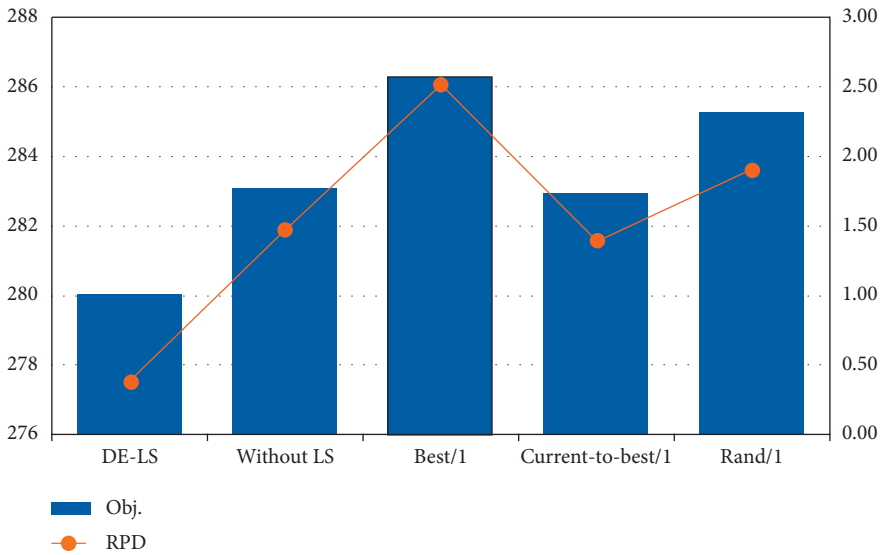


Figure 6: Two-dimensional diagram when the number of machines equals 3.

proposed DE-LS are smaller than 5, which indicate that the proposed DE-LS has stronger robustness than the other compared algorithms. It is worth mentioning that when the number of jobs is 20, the RPD and SD values of the proposed DE-LS are equal to zero, which means that DE-LS has found the minimum value in all runtimes in these two instances. When the number of jobs is no more than 60, the RPD values of DE-LS are no more than 0.5 and the SD values of DE-LS are no more than 1.6, which indicates the proposed DE-LS has shown more superiority in the small-scale problem than in the large-scale problem.

Moreover, two-dimensional comparison plots have been provided in Figures 6 and 7 in order to make a deeper analysis of the experimental results. We select the average objective value and RPD as the variables of the combination-comparison plots. Furthermore, according to the number of machines, we classify all instances into two groups, i.e., the number of machines equal to 3 and the number of machines equal to 4. There is a two-dimensional comparison plot for each group, respectively. The objective value in the diagram for each algorithm is the further average objective, which is calculated by the average objective of the algorithm for each instance in the specific group. Similarly, the RPD value in the diagram for each algorithm is the average RPD, which is calculated by the RPD value of the algorithm for each instance in the specific group. These two combination plots show that

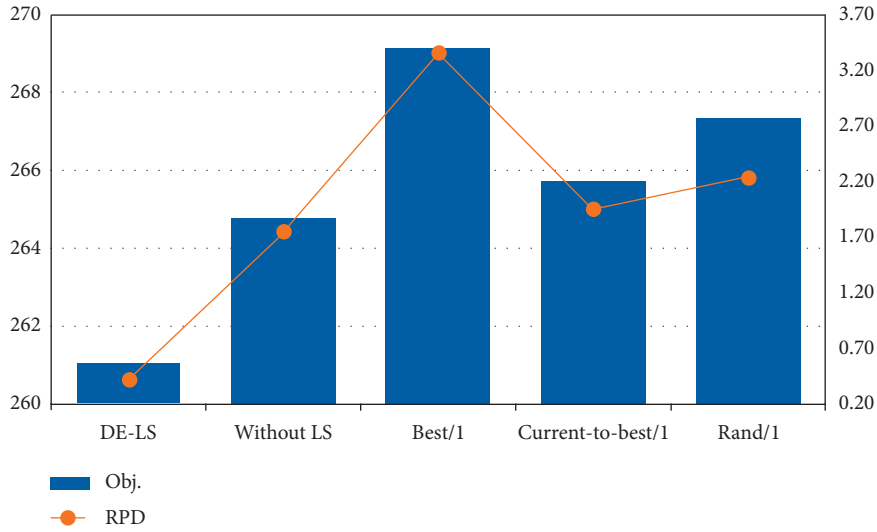


Figure 7: Two-dimensional diagram when the number of machines equals 4.

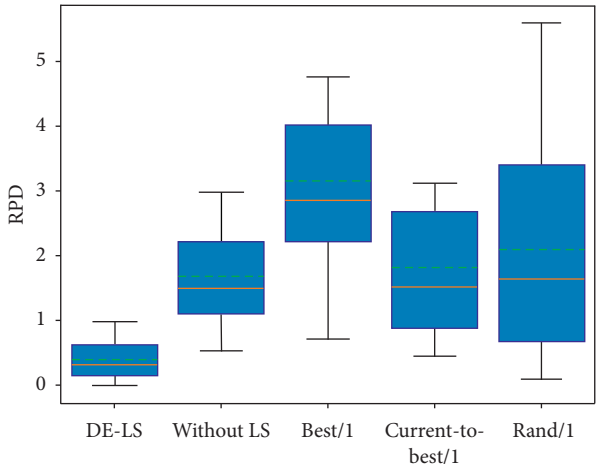


Figure 8: Box-plot of RPD for five compared algorithms.

the proposed DE-LS has outperformed other compared algorithms whatever in objective values or RPD values.

In addition, consider the stochastic features of the metaheuristic algorithms; the stability of the proposed algorithm has been further evaluated using the box-plot analysis. The box-plot can describe the discrete distribution of the data in a relatively stable way, which means the plot will not be influenced by the abnormal value. In this part, the box-plot of the RPD value for five algorithms for 16 instances in 10 independent runs is shown in Figure 8. Appling the concept of quartile, there are six elements having been drawn for each box plot, i.e., the up limb, the up quartile, the average value, the median, the low quartile, and the low limb. In the figure, we can find that all elements in the RPD box-plot graphics of the proposed DE-LS are smaller than other compared algorithms, which shows that there is significant difference in RPD values between the proposed DE-LS and other algorithms. Moreover, the gap between the up limb and low limb of the proposed DE-LS is the smallest. On the contrary, the gap between the up limb and low limb of the rand/1 algorithm is the largest.

5. Conclusions

This paper addresses a hybrid flow-shop scheduling problem with considering different requirements of customers, which reflects the procedures of the productions. A mixed-integer programming model has been formulated for the concerned problem. The objective of the studied problem is to minimize the makespan and cost of delay. Different requirements of customers indicate the diverse productions need to be processed in different procedures and have the specific number of the procedures, which increases the complexity of the studied problem. Moreover, a hybrid metaheuristic algorithm which combined differential evolution and local search named DE-LS has been proposed in the paper to solve the studied problem. The differential evolution and local search are efficient metaheuristic algorithms, which can search for the better solution in a short time. Thus, the paper combines these two algorithms in order to find the optimal solution of the concerned problem in an available time. Moreover, the proposed DE-LS has been compared with four variant algorithms, and the experimental results show that the proposed DE-LS outperforms the compared algorithms. In future study, more practical features in flow-shop scheduling problem will be investigated, such as the delivery requirements of customers, the deteriorating effect of the machines, etc.

Data Availability

The raw/processed data required to reproduce these findings cannot be shared at this time as the data forms part of an ongoing study.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] Q.-K. Pan, M. Fatih Tasgetiren, P. N. Suganthan, and T. J. Chua, "A discrete artificial bee colony algorithm for the lot-streaming flow shop scheduling problem," *Information Sciences*, vol. 181, no. 12, pp. 2455–2468, 2011.

[2] J. Deng, L. Wang, S.-Y. Wang, and X.-L. Zheng, "A competitive memetic algorithm for the distributed two-stage assembly flow-shop scheduling problem," *International Journal of Production Research*, vol. 54, no. 12, pp. 3561–3577, 2016.

[3] J. Lin, Z.-J. Wang, and X. Li, "A backtracking search hyper-heuristic for the distributed assembly flow-shop scheduling problem," *Swarm and Evolutionary Computation*, vol. 36, pp. 124–135, 2017.

[4] J. Deng and L. Wang, "A competitive memetic algorithm for multi-objective distributed permutation flow shop scheduling problem," *Swarm and Evolutionary Computation*, vol. 32, pp. 121–131, 2017.

[5] S. M. Johnson, "Optimal two- and three-stage production schedules with setup times included," *Naval Research Logistics Quarterly*, vol. 1, no. 1, pp. 61–68, 1954.

[6] V. Riahi and M. Kazemi, "A new hybrid ant colony algorithm for scheduling of no-wait flowshop," *Operational Research*, vol. 18, no. 1, pp. 55–74, 2018.

[7] M. K. Marichelvam, Ö. Tosun, and M. Geetha, "Hybrid monkey search algorithm for flow shop scheduling problem under makespan and total flow time," *Applied Soft Computing*, vol. 55, pp. 82–92, 2017.

[8] 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.

[9] 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.

[10] W. Qin, J. Zhang, and D. Song, "An improved ant colony algorithm for dynamic hybrid flow shop scheduling with uncertain processing time," *Journal of Intelligent Manufacturing*, vol. 29, no. 4, pp. 891–904, 2018.

[11] D. Lei and Y. Zheng, "Hybrid flow shop scheduling with assembly operations and key objectives: a novel neighborhood search," *Applied Soft Computing*, vol. 61, pp. 122–128, 2017.

[12] B. Zhang, Q.-K. Pan, L. Gao, X.-L. Zhang, H.-Y. Sang, and J.-Q. Li, "An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming," *Applied Soft Computing*, vol. 52, pp. 14–27, 2017.

[13] C. Yu, Q. Semeraro, and A. Matta, "A genetic algorithm for the hybrid flow shop scheduling with unrelated machines and machine eligibility," *Computers & Operations Research*, vol. 100, pp. 211–229, 2018.

[14] Q.-K. Pan, L. Gao, X.-Y. Li, and K.-Z. Gao, "Effective metaheuristics for scheduling a hybrid flowshop with sequence-dependent setup times," *Applied Mathematics and Computation*, vol. 303, pp. 89–112, 2017.

[15] A. Ebrahimi, H. W. Jeon, S. Lee, and C. Wang, "Minimizing total energy cost and tardiness penalty for a scheduling-layout problem in a flexible job shop system: a comparison of four metaheuristic algorithms," *Computers & Industrial Engineering*, vol. 141, Article ID 106295, 2020.

[16] S. Kavitha and P. Venkumar, "A vibrant crossbreed social spider optimization with genetic algorithm tactic for flexible job shop scheduling problem," *Measurement and Control*, vol. 53, no. 1-2, pp. 93–103, 2020.

[17] T. R. Chinnusamy, P. Kammar, F. Praveen et al., "Scheduling of flexible manufacturing system by hybridizing petri net with improved scatter search algorithm," in *Recent Trends in Mechanical Engineering*, G. S. V. L. Narasimham, A. V. Babu, S. S. Reddy, and R. Dhanasekaran, Eds., pp. 305–332, Springer Singapore, Singapore, 2020.

[18] G. Kliewer and S. Tschoke, "A general parallel simulated annealing library and its application in airline industry," in *Proceedings of the 14th International Parallel and Distributed Processing Symposium. IPDPS 2000*, pp. 55–61, Cancun, Mexico, May 2000.

[19] O. Ezzinbi, M. Sarhani, A. El Afia, and Y. Benadada, "A metaheuristic approach for solving the airline maintenance routing with aircraft on ground problem," in *Proceedings of the 2014 International Conference on Logistics Operations Management*, pp. 48–52, Rabat, Morocco, June 2014.

[20] E. Teymourian, A. Sadeghi, and F. Taghipourian, "A dynamic virtual hub location problem in airline networks—formulation and metaheuristic solution approaches," in *Proceedings of the First International Technology Management Conference*, pp. 1061–1068, San Jose, CA, USA, June 2011.

[21] R. Mansi, S. Hanafi, C. Wilbaut, and F. Clautiaux, "Disruptions in the airline industry: math-heuristics for re-assigning aircraft and passengers simultaneously," *European Journal of Industrial Engineering*, vol. 6, no. 6, p. 690, 2012.

[22] N. R. Das, S. C. Rai, S. C. Rai, and A. Nayak, "Intelligent scheduling of demand side energy usage in smart grid using a metaheuristic approach," *International Journal of Intelligent Systems and Applications*, vol. 10, no. 6, pp. 30–39, 2018.

[23] M. A. Djema, M. Boudour, K. Agbossou, A. Cardenas, and M. L. Doumbia, "Adaptive direct power control based on ANN-GWO for grid interactive renewable energy systems with an improved synchronization technique," *International Transactions on Electrical Energy Systems*, vol. 29, Article ID e2766, 2019.

[24] F. Ekinci, T. Demirdelen, I. O. Aksu, K. Aygul, B. Esenboga, and M. Bilgili, "A novel hybrid metaheuristic optimization method to estimate medium-term output power for horizontal axis wind turbine," *Proceedings of the Institution of Mechanical Engineers, Part A: Journal of Power and Energy*, vol. 233, no. 5, pp. 646–658, 2019.

[25] D. Dabhi and K. Pandya, "Enhanced velocity differential evolutionary particle swarm optimization for optimal scheduling of a distributed energy resources with uncertain scenarios," *IEEE Access*, vol. 8, pp. 27001–27017, 2020.

[26] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[27] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[28] S. Das and P. N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[29] G. Zhang and K. Xing, "Differential evolution metaheuristics for distributed limited-buffer flowshop scheduling with makespan criterion," *Computers & Operations Research*, vol. 108, pp. 33–43, 2019.

[30] B.-H. Zhou, L.-M. Hu, and Z.-Y. Zhong, "A hybrid differential evolution algorithm with estimation of distribution algorithm

for reentrant hybrid flow shop scheduling problem," *Neural Computing and Applications*, vol. 30, no. 1, pp. 193–209, 2018.

[31] R. Ruiz and T. Stützle, "A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 2033–2049, 2007.

[32] J. Xu, C.-C. Wu, Y. Yin, and W.-C. Lin, "An iterated local search for the multi-objective permutation flowshop scheduling problem with sequence-dependent setup times," *Applied Soft Computing*, vol. 52, pp. 39–47, 2017.

[33] M. F. Tasgetiren, D. Kizilay, Q.-K. Pan, and P. N. Suganthan, "Iterated greedy algorithms for the blocking flowshop scheduling problem with makespan criterion," *Computers & Operations Research*, vol. 77, pp. 111–126, 2017.

[34] B. Yagmahan and M. M. Yenisey, "A multi-objective ant colony system algorithm for flow shop scheduling problem," *Expert Systems with Applications*, vol. 37, no. 2, pp. 1361–1368, 2010.

[35] J. Bruno and P. Downey, "Complexity of task sequencing with deadlines, set-up times and changeover costs," *SIAM Journal on Computing*, vol. 7, no. 4, pp. 393–404, 1978.

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