sustainability

MDPI

Article

A Multiobjective Optimization Approach for Multiobjective Hybrid Flowshop Green Scheduling with Consistent Sublots

Weiwei Wang, Biao Zhang * and Baoxian Jia

School of Computer Science, Liaocheng University, Liaocheng 252000, China
* Correspondence: zhangbiao1218@gmail.com

Abstract: Hybrid flowshop scheduling problems are encountered in many real-world manufacturing scenarios. With increasingly fierce market competition, the production mode of multiple varieties and small batches has gradually been accepted by enterprises, where the technology of lot streaming is widely used. Meanwhile, green criteria, such as energy consumption and carbon emissions, have attracted increasing attention to improving protection awareness. With these motivations, this paper studies a multiobjective hybrid flowshop green scheduling problem with consistent sublots (MOHFGSP_CS), aiming to minimize two objectives, i.e., makespan and total energy consumption, simultaneously. To solve this complex problem, we first formulate a novel multiobjective optimization model. However, due to the NP-hard nature of the problem, the model is computationally prohibitive as the problem scale increases. Thus, a multiobjective discrete artificial bee colony algorithm (MDABC) based on decomposition is proposed. There are three phases in this algorithm: the VND-based employed bee phase, the adjustment weight onlooker bee phase, and the population interaction scout bee phase. In the experimental study, various small-scale and large-scale instances are collected to verify the effectiveness of the multiobjective optimization model and the MDABC. Comprehensive computational comparisons and statistical analysis show that the developed strategies and MDABC show superior performance.

Keywords: hybrid flowshop scheduling problem; green scheduling; consistent sublots; multiobjective evolutionary algorithm

Citation: Wang, W.; Zhang, B.; Jia, B. A Multiobjective Optimization Approach for Multiobjective Hybrid Flowshop Green Scheduling with Consistent Sublots. *Sustainability* **2023**, *15*, 2622. https://doi.org/10.3390/su15032622

Academic Editors: Maxim A. Dulebenets and Mirco Peron

Received: 5 December 2022
Revised: 24 January 2023
Accepted: 26 January 2023
Published: 1 February 2023

1. Introduction

As a classical production scheduling problem, the hybrid flow shop scheduling problem (HFSP) appears in various flexible manufacturing systems, such as electronics, paper, textile, and petrochemical [1]. In a real-world hybrid flow shop, there is a set of jobs that can select exactly one machine from several identical parallel machines at a given stage. The HFSP aims to determine the job sequence and machine assignment at each stage, and it has been proven to be strongly NP-hard in the strong sense [2]. Given its important theoretical and practical value, the HFSP has aroused wide concern in recent decades. Various solution methods have been developed for solving them in the literature [3–5]. It is worth noting that most research on solving the HFSP treats each job as a whole part. Because of the intense market competition, more and more enterprises are steadily adopting the production method of multiple varieties and small batches. The concept of lot streaming was formally introduced by Reiter [6] and is widely used in the flowshop context. Lot streaming has emerged as an attractive method for reducing makespan, cycle time, work-in-process inventory, etc. Using the division methodology, a batch of jobs (called a lot) is divided into several sublots that can be processed in an overlapping fashion at successive stages in a multistage manufacturing system. The division methodology can be classified into three categories: consistent sublots, equal sublots, and variable sublots [7]. Consistent sublots indicate that the sizes of different sublots may not be the same but consistent among different stages. For equal sublots, the sizes of different sublots of a lot are equal at different

stages. With the variable sublots, the sizes of different sublots are different and remain variable among different stages. Equal sublots are a special kind of consistent sublots. For the variable sublots, the complex scenario of variable sublots can cause difficulties in the transformation and management of the system. Thus, this study introduces the consistent sublots into the HFSP, resulting in a scheduling problem, namely, the HFSP with consistent sublots (HFSP_CS). Previous studies on the HFSP_CS only consider production efficiency and do not pay attention to environmental factors [8].

With the improvement of environmental protection awareness, green manufacturing has become a research hotspot in the field of advanced manufacturing. According to statistics, the manufacturing industry consumes approximately 31% of primary energy consumption and 36% of carbon emissions [9,10]. In practice, machines in any given factory can vary in terms of power consumption. Furthermore, any given machine consumes power at a different rate depending on the speed at which it is run. In general, the faster the speed, the faster the lots are completed; but the faster the speed, the more power consumption. Therefore, the purpose of hybrid green flowshop scheduling is to reduce energy consumption, reduce environmental pressure, and achieve sustainable development.

Given the above, considering the variable machine processing speeds, this paper studies a multiobjective hybrid flowshop green scheduling problem with consistent sublots (MOHFGSP_CS) with two minimizing objectives, i.e., makespan and total energy consumption, simultaneously. To solve this problem, we first develop a multiobjective mixed integer programming model (MILP) [11,12] by considering real-world processing constraints. The model is solved by the mathematical programming software Gurobi 9.5.0 by assigning a set of uniform weights to the two objectives. It is found that the two objectives have a trade-off relationship. Meanwhile, the optimal solutions cannot be obtained within a reasonable amount of time as the problem scale increases. Due to their trade-off nature, there does not exist a single solution with two minimum objective values. Therefore, we try to find a set of Pareto solutions for the MOHFGSP_CS.

The multiobjective evolutionary algorithm (MOEA) based on decomposition is suggested to obtain Pareto solutions of their superior performance in solving the multiobjective problems. In this paper, considering the problem-specific characteristics, an artificial bee colony algorithm (MDABC) based on decomposition is proposed. In the VND-based employed bee phase, the variable neighborhood descent (VND) with eight designed neighborhoods is employed for a search. In the adjustment weight onlooker bee phase, a weight adjustment strategy (WAS) is introduced to maintain the diversity of solutions. In the population interaction scout bee phase, the solution interaction strategy (SIS) is developed to enhance the exploration of the algorithm.

The major contributions of this paper can be summarized as follows: (1) regarding the problem modelling, a mathematical model for the MOHFGSP_CS is formulated; in this model, two objectives, including makespan and total energy consumption, are taken into account; (2) regarding the optimization algorithm, we adopted the Tchebycheff decomposition approach to obtain different Pareto optimal solutions; (3) regarding the experiment, we verify the excellent performance of the MDABC via a set of computational experiments. The novelties of the proposed MDABC are as follows: (1) a three-level encoding is specially designed; (2) eight neighborhood structures and the VND are introduced to enable that each subproblem can be optimized independently; (3) the WAS is introduced to maintain the diversity of solutions; (4) the SIS is developed to enhance the exploration of the solution thoroughly along its different direction.

The remainder of this paper is organized as follows: Section 2 provides the literature review; Section 3 gives the problem description and formulates a mathematical model for this problem; Section 4 describes the proposed MDABC for solving the MOHFGSP_CS; Section 5 presents the comprehensive comparison of MDABC; Section 6 summaries this study and offers future research directions.

2. Literature Review

In this section, we first review the studies on HFSPs. Next, the HFSP with lot streaming regarding equal and consistent sublots are reviewed. Finally, green scheduling research on HFSP environments is reviewed.

2.1. The Classical Hybrid Flowshop Scheduling Problem

HFSP is an extension of classical flowshop scheduling that contains multiple operations, and each operation has one or more parallel machines. In most cases, they are defined as NP-hard problems [2]. In the two-stage HFSP problem, Gupta showed that minimizing the makespan with identical parallel machines at each stage is strongly NP-hard [13]. Likewise, Yang demonstrated that a minimization makespan scheduling problem for a two-stage hybrid flow shop with dedicated machines at the second stage is NP-complete [14]. Guirchon et al. [15] proposed a novel polynomial time algorithm to solve a special zero wait two-stage HFSP with different parallel machines in the second stage.

2.2. The Hybrid Flowshop Scheduling Problem with Lot Streaming

In a recent study, the lot streaming splitting is adopted where parallel processing of the same lot in different stages is possible. Table 1 shows the major research concerning minimizing a single objective. In the case of the equal sublots, Cheng et al. [16] studied a special two-stage HFSP containing only one machine in the first stage and two machines in the second stage. First, they determine the optimal sublot size with the number of sublots known; second, the upper limit of sublot quantity is determined according to the size of sublot, and a heuristic method for determining sublot sequence is proposed. Zhang et al. [17] analyzed k-stage HFSP and proposed a heuristic method for determining the size of and sequence of the sublot. In case of the consistent sublots, Kim et al. [18] studied the two-stage HFSP problem under hypothetical conditions for lot streaming problems. Zhang et al. [8] studied k-stage HFSP and proposed a collaborative variable descent algorithm to solve the lot sequence, machine assignment, and lot split, simultaneously. To determine the lot sequence, they developed a MILP model and a local search algorithm. Nejati et al. [19] explored a k-stage HFSP with various special production restrictions, such as work-in-process inventory, working shift, and sequence-dependent setup, taking the case of intermingling into consideration. In their study of a k-stage HFSP, Lalitha et al. [20] only took into account one machine at each stage except the final one. They created a MILP model and employed LINGO software to optimally resolve small-scale issues. In the intermingling case, the number of sublots is assumed to be known, Nejati et al. [21] studied a two-stage assembly HFSP and they proposed the genetic algorithm to solve the problem.

Table 1. Comparison of this study with previous studies considering lot steaming.

Studies	Type of HFSP	Objective	Problem Characteristic
Cheng et al. [16]	Two-stage	Makespan	With equal sublot
Zhang et al. [17]	K-stage	Production lead time	With equal sublot
Kim et al. [18]	Two-stage	Makespan	With consistent sublot
Zhang et al. [8]	K-stage	Makespan	With consistent sublot
This study	HFSP	Makespan, Total energy consumption	With consistent sublot

2.3. The Hybrid Flowshop Green Scheduling Problem

Regarding green scheduling considering energy consumption, Dai et al. [22] proposed an improved genetically simulated annealing algorithm for an energy-efficient flexible flow shop scheduling problem. Fernandez-Viagas et al. [23] used different meta-heuristics or more complex local search procedures to reduce total carbon consumption for the permutation flowshop scheduling problem. Gu et al. [24] proposed a hybrid cuckoo search algorithm to solve MOPFSP with the objective of minimizing total carbon emissions. To solve this multiobjective mixed-integer linear programming (MILP), Lu et al. [25] proposed

the collaborative multi-objective optimization algorithm. Meng et al. [26] developed an MILP model with the objective of minimising energy consumption. For this kind of multiobjective problem, there is usually not a single best solution but a set of solutions that are superior to others when considering all objectives. This set is called the Pareto set of nondominated solutions. Much multiobjective research on ABCs has been proposed. Baiand and Liu proposed a multiobjective artificial bee colony (MOABC) algorithm based on decomposition by the penalty-based boundary intersection (PBI) method [27]. Additionally, considering the relative importance of objectives, Li, Lei, and Cai [28] addressed an energy-efficient HFSP with total tardiness, makespan, and total energy consumption. In this study, the machine speed selection for each job at each stage needs to be determined optimally. Yan et al. [29] divided an energy-efficient HFSP into a machine tool level and a shop floor level that were solved hierarchically. Zhang et al. [30] proposed a decomposition algorithm for HFSP. Experimental results show that the proposed multilevel optimization approach can reduce the makespan and total energy consumption.

Through the above review, as shown in Table 2, we do not find any study on the energy consumption of the hybrid flowshop scheduling problem with a consistent sublot. Therefore, the research in this paper has a very important role. The multiobjective hybrid flowshop green scheduling with consistent sublots studied in this paper draws on the previous lot splitting strategy and the encoding and decoding of green scheduling. MOHFGSP_CS aims to determine the lot sequence, lot split, machine assignment, and speed selection to minimize the makespan and total energy consumption simultaneously.

Table 2. Comparison of this study with previous studies about green scheduling.

Studies	Type	Objective	Problem Characteristic
Fernandez-Viagas et al. [23]	FPSP	Makespan, Total energy consumption	With various machine speeds
Gu et al. [24]	FFSP	Makespan, Total energy consumption	With the relative objectives
Lu et al. [25]	DPFSP	Total energy consumption	With limited buffers
Li et al. [28]	DPFSP	Total flowtime	With the relative algorithm
Zhang et al. [30]	HFSP	Makespan, Total energy consumption	With lot steaming
This study	HFSP	Makespan, Total energy consumption	With lot steaming and various machine speeds

3. Problem Statement

3.1. Mathematical Model

The MOHFGSP_CS is described as follows. Several lots are to be examined consecutively through a series of stages, each of which has several sizes. At each stage, there exist several identical and parallel machines. With consistent sublots, many will be split into a number of sublots, which is limited to a maximum value. Each sublot contains a different number of sublot sizes. The sublot quantity and sublot size are consistent at different stages. The processing time of the sublot is the product unit time and sublot size. The setup time is only expected before the processing time of the first sublot. In addition, machines can be assigned variable speed levels. It is often assumed that energy consumption increases and the processing time decrease when a lot is processed on a machine at a higher speed level [31]. In this study, we take into account two conflicting objectives simultaneously: the makespan and total energy consumption. The problem needs to determine the lot sequence, lot split, machine assignment, and machine speed selection. The makespan refers to the completion time of the last sublots processed in the system. The total energy consumption includes three parts: machine processing power consumption, setup time power consumption, and idle time power consumption. Several assumptions are given as follows [30,32]:

- All machines and lots are available at time zero;
- Each unit can only be processed on one machine at a time;
- Preemption is not allowed, and buffer size is not limited;
- Each machine cannot process more than one unit at a time;

- The machines are turned on when the first lot assigned to them is going to start and turn off once all lots assigned to them are finished;
- A sublot at a stage can be processed after it has been completed at the previous stage and transported to this stage;
- Machine speed cannot be changed while processing a sublot.

To present a mathematical model for the problem described in the previous section, we define several notions as follows.

(1) Notations:

K	The total number of stages;
k	Index of stages, $k \in \{1, 2, 3, \ldots, K\}$;
J	Total number of lots;
j	Index of lots, $j \in \{1, 2, 3, \ldots, J\}$;
M_k	Number of parallel machines at stage k;
i	Index of machines at stage k,$i \in \{1, 2, 3, \ldots, M_k\}$;
T_j	Total number of lots j;
L	The maximum number of sublots of each lot;
e	Index for the sublots, $e \in \{1, 2, 3, \ldots, L\}$;
$LS_{j,e}$	Number units of sublot e of lot j;
v	The speed level index;
V_k	The speed level of machines at stage k, and $V_k \in \{1, 2, 3, \ldots, v, \ldots, \lvert V_k \rvert\}$;
$P_{k,j,v}$	Unit processing time of sublot e of lot j by speed v;
$BT_{k,j,e}$	Beginning processing time of sublot e of lot j at stage;
$ET_{k,j,e}$	Ending processing time of sublot e of lot j at stage k;
$S_{k,j}$	Set up the time of lot j at stage k;
$T_{k,j}$	Transportation time of lot j at stage k;
$PW_{k,i,v}$	The power of consumption of machine i in stage k by speed j during processing status;
$SW_{k,i,v}$	The power of consumption of the machine i in stage k by speed j during setup status;
$PI_{k,i}$	The power of consumption of machine i in stage k during idle status.

(2) Decision variables:

$W_{j,e}$	Binary variable that takes the value of 1 when the sublot of lot j is larger than 0 and 0 otherwise;
$D_{k,j,i}$	Binary variable that takes the value of 1 when a lot j is assigned to machine i at stage k and 0 otherwise;
$Y_{k,j,j\prime,i}$	Binary variable that takes the value of 1 when l j is scheduled before lot j' on machine i at stage k and 0 otherwise;
$X_{k,i,j,v}$	Binary variable that takes the value of 1 when lot j is processed on machine i at stage k by speed v and 0 otherwise.

(3) Objectives:

$$Minimize(C_{\max}) \tag{1}$$

$$MinimizeTEC = E1 + E2 + E3 \tag{2}$$

$$E1 = \sum_{j\in J}\sum_{e\in E}\sum_{i\in Mk}\sum_{v\in Vk} PW_{k,j,v} * P_{k,j,v} * LS_{j,e} * D_{k,j,i} * X_{k,j,v} \tag{3}$$

$$E2 = \sum_{j\in J}\sum_{e\in E}\sum_{i\in Mk}\sum_{v\in Vk} D_{k,j,i} * S_{k,j} * SW_{k,i,v} * LS_{j,e} \tag{4}$$

$$E3 = \sum_{j\in J}\sum_{e\in E}\sum_{i\in Mk}\sum_{v\in Vk}(BT_{k,j,e} + 1 - ET_{k,j,e} - \sum_{j\in J}\sum_{v\in Vk} X_{k,j,i,v} (P_{k,j,v} + \sum_{j\in J\prime} Y_{k,j\prime,j,i} * S_{k,j})) * PI_{k,i,v} * LS_{j,e} \tag{5}$$

(4) Constraints:

$$\sum_{i=1}^{Mk} Dk, j, i = 1 \quad \forall i \in M_k, j \in J, k \in K \tag{6}$$

$$\sum_{e=1}^{L} LS_{j,e}, e = T_j \quad \forall j \in J, e \in L \tag{7}$$

$$\frac{LS_{j,e}}{G} \leq W_{j,e} \leq LS_{j,e} \quad \forall j \in J, e \in L \tag{8}$$

$$W_{j,e} \leq W_{j,e+1} \quad \forall j \in J, e \in L \tag{9}$$

$$ET_{k,j,e} - BT_{k,j,e} = P_{k,j,v} \times LS_{j,e} \quad \forall k \in K, j \in J, e \in L \tag{10}$$

$$BT_{k+1,j,e} - ET_{k,j,e} \geq 0 \quad \forall k \in K, j \in J, e \in L \tag{11}$$

$$BT_{k,j,1} - ET_{k,j\prime,L} - S_{k,j} + G(3 - Y_{k,j\prime j,i} - D_{k,j,i} - D_{k,j\prime,i}) \geq 0 \forall k \in K, j \in J, e \in L \tag{12}$$

$$BT_{k+1,j,e} - ET_{k,j,e} \geq W_{j,e} \times T_{k,j} \quad \forall k \in \{1, 2, 3, \ldots, K-1\}, j \in J, e \in L \tag{13}$$

$$Y_{k,j,j\prime,i} + Y_{k,j\prime,j,i} \leq 1 \forall k \in K, j, j\prime \in J, i \in M_k \tag{14}$$

$$Y_{k,j,j\prime i} + Y_{k,j\prime,j,i} \leq D_{k,j,i} + D_{k,j\prime,i} \quad \forall k \in K, j, j\prime \in J, i \in M_k \tag{15}$$

$$Y_{k,j,j\prime i} + Y_{k,j\prime,j,i} \geq D_{k,j,i} + D_{k,j\prime,i} - 1 \quad \forall k \in K, j, j\prime \in J, i \in M_k \tag{16}$$

$$Y_{k,j,j\prime i} \in \{0, 1\} \quad \forall k \in K, j, j\prime \in J, i \in M_k \tag{17}$$

$$X_{k,i,j,v} \in \{0, 1\} \quad \forall k \in K, j, j\prime \in J, i \in M_k, v \in V \tag{18}$$

Objective (1) minimizes the maximum completion time $C_{\max}$. Objective (2) determines the total energy consumption, where $E1$ represents the energy consumption when the machines are in the processing state, $E2$ represents the energy consumption when the machines are in the setup state, and $E3$ represents the energy consumption when the machines stay idle. Equation (6) specifies the limits of the lot arrangement; that is, each lot must go through all stages and can only be processed by exactly one machine at each stage. Equation (7) defines the lot split constraint, which requires that for a given lot, the sum of its sublot sizes must be equal to its lot size. Equation (8) establishes the relationship between two decision variables $LS_{j,e}$ and $W_{j,e}$. Equation (9) determines the priorities of the sublots to accommodate the units, and the priority of the previous sublot is higher than that of the following sublot. Equation (10) requires that the processing of each sublot cannot be interrupted. For each sublot, its processing time is equal to the unit processing time times the number of units unit number. In Equation (11), for sublots, their start time at the next stage must be greater than or equal to their completion time at the previous stage. Equation (12) expresses that the first sublot of a lot can be started only after the completion of the last sublot of the previous lot if it exists and the setup operation. Equation (13) requires that at a given stage, a sublot can be started only after it has been completed at the previous stage and transferred to this stage. For a sublot, the transfer is only required when its sublot size is larger than 0. Equation (14) guarantees that $Y_{k,j,j\prime,i}$ does not take the value of 1 at the same time. If j and $j\prime$ are processed on the same machine, they must be processed in sequence. Equations (15) and (16) together establish the relationship between the two variables and $D_{k,j,i}$. Equations (17) and (18) define the possible values of three decision variables $Y_{k,j,j\prime,i}$, $D_{k,j,i}$ and $X_{k,i,j,v}$.

3.2. Trade-Off Relationship of the Objectives

To illustrate the makespan and TEC, consider an example with five lots and two stages. The maximum sublot quantity is set to five, and at each stage, there exist two identical and parallel machines that have three speed levels. The relevant data are described as follows.

$$L_{j,e} = \begin{bmatrix} 1,2,1 \\ 3,3,0 \\ 2,0,0 \\ 1,1,0 \\ 2,2,0 \end{bmatrix}$$

$$T_{k,j} = [1,1,1,1,1]$$

$$S_{k,j} = \begin{bmatrix} 3,4,3,2,1 \\ 1,3,1,3,2 \end{bmatrix}$$

$$V_k = \begin{bmatrix} 1,2,1,3,1 \\ 1,1,2,3,2 \end{bmatrix}$$

$$P_{k,j,v} = \begin{bmatrix} 2,3,2,1,2 \\ 2,2,3,1,3 \end{bmatrix}$$

$$PW_{k,i,v} = \begin{bmatrix} 4,16,4,36,4 \\ 4,4,16,36,16 \end{bmatrix}$$

$$SW_{k,i,v} = [2,2]$$

$$PI_{k,i} = [1,1]$$

To solve the aforementioned example by running the model on Gurobi 9.5.0, we assign 20 uniform weights in the range of [0, 1] to the two objectives, resulting in 20 subproblems. These subproblems can be solved to optimality and output 20 optimal solutions. By collecting these solutions, Figure 1 illustrates the plot of the obtained makespan versus the TEC. It shows that the two objectives cannot be optimized simultaneously. In addition, Figures 2 and 3 show Gantt charts of the example for the optimal makespan and optimal TEC, respectively. They are very different. These figures show that these two objectives conflict with each other. In detail, when makespan increases, TEC decreases, and when makespan decreases, TEC increases. Thus, we can ensure that there exists a trade-off relationship between the two objectives.

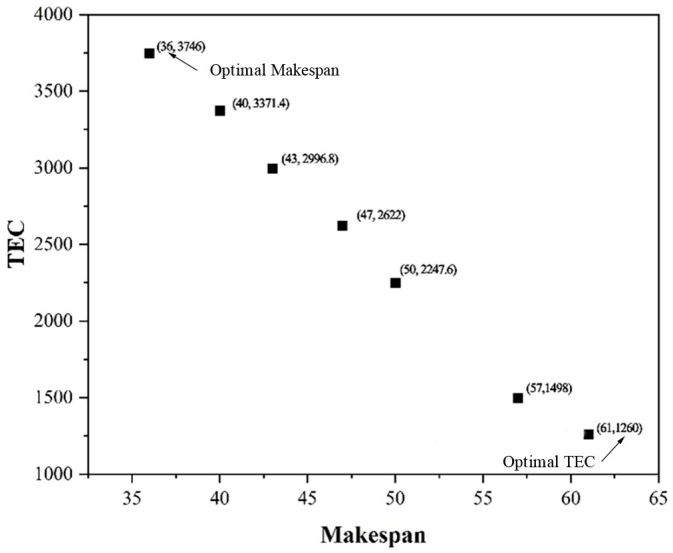


Figure 1. Plots of makespan versus TEC.

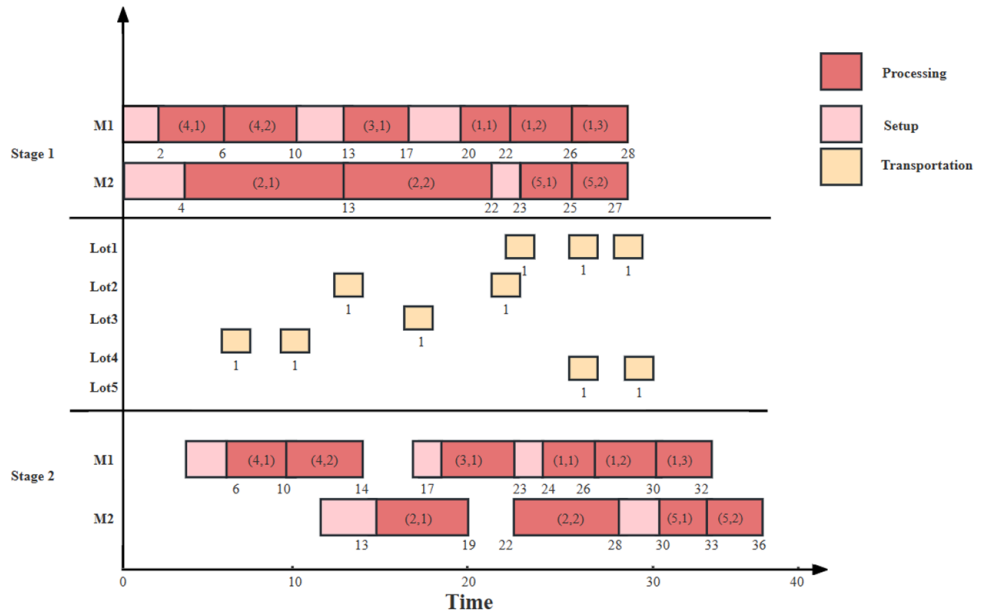


Figure 2. Gantt chart of the schedule for the optimal Cmax.

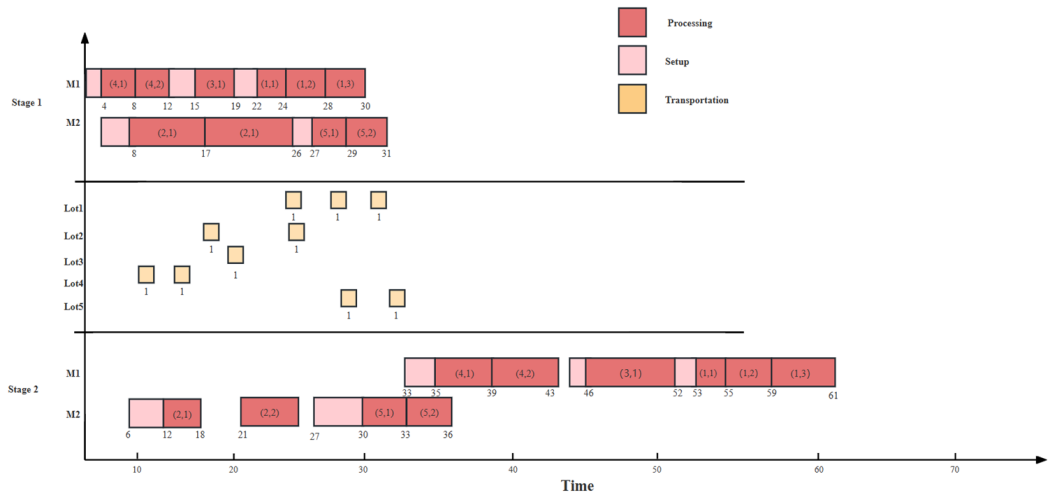


Figure 3. Gantt chart of the schedule for the optimal TEC.

4. Proposed Algorithm

Using the decomposition strategy, the proposed MDABC decomposes the MOHFGSP_CS into a set of subproblems and optimizes them simultaneously. The framework of the MDBC is displayed in Figure 4, which mainly contains four phases. In the following, considering the problem-specific characteristics, the solution encoding and decoding rules are first designed. Then, the approaches of decomposition and objective normalization are detailed. Next, the employed VND-based bee phase, the WAS-based onlooker bee phase, and the SIS-based scout bee phase are detailed.

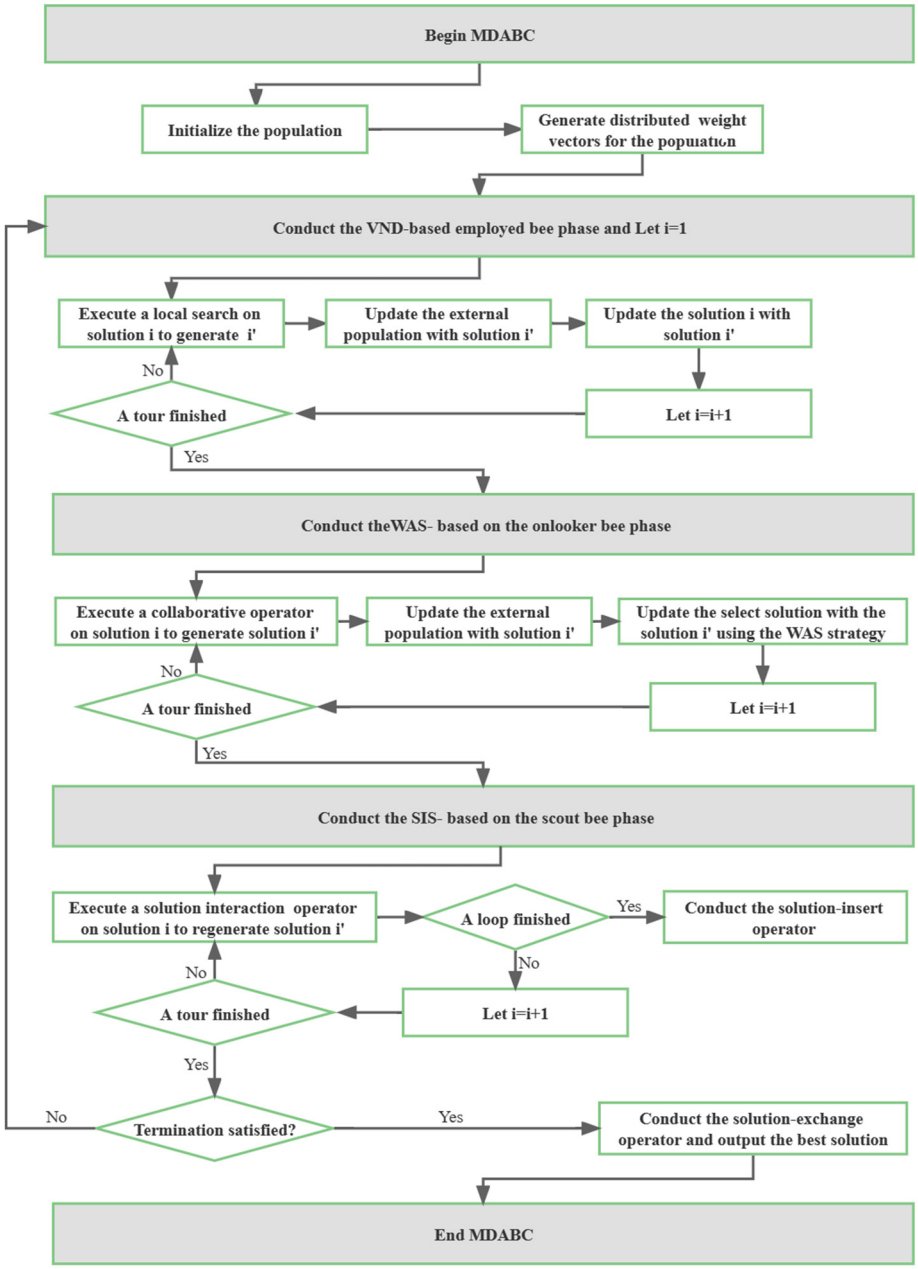


Figure 4. The proposed MDABC framework.

4.1. Solution Encoding and Population Initialization

To solve the MOHFGSP_CS, four tasks need to be addressed simultaneously, i.e., lot sequence, lot split, machine assignment, and machine speed selection. In this paper, machine assignment is addressed in solution decoding using the strategy of an available machine strategy, which is widely used in the HFSP literature and has outstanding performance [33]. The other tasks can be reflected directly in the solution representation. The solution is encoded into three parts. The first part is an n-dimensional lot permutation $\pi_n = \{\pi(1), \ldots, \pi(j), \ldots, \pi(n)\}$, where $\pi(j)$ denotes a lot and n represents the number of lots. This permutation only represents the lot sequence at the first state.

The second part is lot split matrix $\Delta_{n\times L}$ as shown below, which denotes the size of the eth sublot of the jth lot.

$$\Delta_{n\times L}=\begin{bmatrix} \mu_{1,1} & \mu_{1,2} & \cdots, & \mu_{1,L} \\ \mu_{2,1} & \mu_{2,2} & \cdots, & \mu_{2,L} \\ \vdots & \vdots & \mu_{j,e} & \vdots \\ \mu_{2,1} & \mu_{n,2} & \cdots, & \mu_{2,L} \end{bmatrix}$$

The third part is a machine speed selection matrix $v_{m\times n}$, which is constructed as follows. $v_{j,e}$ denotes the machine speed level for sublot e at stage j.

$$v_{m\times n}=\begin{bmatrix} v_{1,1} & v_{1,2} & \cdots, & v_{1,n} \\ v_{2,1} & v_{2,2} & \cdots, & v_{2,n} \\ \vdots & \vdots & v_{j,e} & \vdots \\ v_{m,1} & v_{m,2} & \cdots, & v_{m,n} \end{bmatrix}$$

There are N randomly generated solutions in our algorithm. By using the decomposition strategy, each solution should be assigned a weight vector, resulting in N weight vectors. Uniformly distributed weights are essential to produce uniformly distributed Pareto solutions. Thus, we generated a set of uniformly distributed weight vectors using the method proposed by Zhang et al. [30].

Here, the elements in the weight vector will be chosen from the weight vectors poor W, which take values from $\{ {}^{0}/_{\mathrm{H}}, {}^{1}/_{\mathrm{H}}, \ldots, {}^{\mathrm{H}}/_{\mathrm{H}} \}$. H is a positive integer that controls the size of W. Afterward, we generate $N = C_{H+m-1}^{m-1}$ different weight vectors, where m is the number of objectives. To solve the multiobjective problem, we set m to 2 and H to $N - 1$.

4.2. Solution Decoding

Regarding solution encoding, the heuristic rules for the lot sequence and machine assignment are taken into consideration in order to decode the solution into a feasible schedule.

For the lot sequence, in the HFSP literature, the following rule is commonly used. At the first stage, the lot permutation in the solution representation can directly reflect the lot sequence. At the following stages, the job that has been completed earlier at the previous stage receives the priority to be scheduled at the following stage.

Considering the characteristics of the problem lot split, two heuristics are designed. One is called the Sublot Priority (SP). The sooner the first sublot of a lot is completed in the previous stage, the sooner it will be processed. Another one is called Lot Priority (LP). The sooner the last sublot of a lot is completed in the previous stage, the sooner it will be processed.

For the machine assignment, two common heuristics are employed. One is the First Available (FA). In this rule, for the given lot, the machine that has the earliest available time will be assigned to the lot. Another is the First Completion (FC). In this rule, for the given lot, the machine that can complete the lot earliest will be assigned earliest. According to the above description, the whole decoding process for a solution is summarized as follows.

At the first stage, i.e., $i = 1$, take ${\pi'}_j$ from ${\pi'}_n$ one by one and then conduct the following steps:

(1) Find the earliest available machine using the FA or FC, and then assign π_j to the machine;

(2) The completion time of the sublots $(e = 1, \ldots, L)$ are calculated as the unit processing time times at the v speed level of the number of units unit number. The unit processing time is known, the number of units is known from $\Delta_{n\times L}$, v is known from $v_{m\times n}$, and the available time of the selected machine can also be updated.

At the following stages, i.e., $i = \{2, \ldots, m\}$, obtain a new lot sequence using the SP or LP. We take ${\pi'}_j$ from ${\pi'}_n$ one by one and conduct the following steps.

(1) Find the assigned machine according to FA or FC and assign it π'_j to the machine;
(2) The completion time of sublots is calculated using the above method. Update the machines available time.

4.3. Decomposition Approaches and Objective Normalization

The weighted sum approach and the Tchebycheff approach are two related decomposition methods for solving combinatorial problems [34]. The two approaches are introduced in detail below.

Assume a MILP is considered with m minimization objectives. Consider $\lambda = (\lambda_1, \ldots, \lambda_i, \ldots, \lambda_m)$ being the weight vector assigned to the scalar optimization problem, where λ_i represents the weight value of the ith objective. The weighted sum approach can be defined is:

$$\min imizeg(X|\lambda, Z^*) = \sum_{i=1}^{m} \lambda_i f_i(X), \text{ subject to } X \in \Omega \tag{19}$$

where X is the variable that needs to be optimized, $f_i(X)$ is the objective value of the ith objective, and Ω represents the feasible solution space. While the Tchebycheff approach can be defined is:

$$\min imizeg(X|\lambda, Z^*) = \max_{1 \le i \le m} \{\lambda_i |f_i(x)| - Z_i^*\} \tag{20}$$

where Z_i^* is the optimal value of the ith objective. In this article, low values for the two objectives can be used as reference points.

As discussed in Section 2, two objectives have disparately scaled values in this paper. Objective normalization can improve the performance of the algorithm because the search might favor the objective with a larger scale. For this reason, the max–min method is used to normalize the objectives between 0 and 1 [32], which is described as follows:

$$\widetilde{f_i} = \frac{f_i - \min(f_i)}{\max(f_i) - \min(f_i)} \tag{21}$$

where $\widetilde{f_i}$ is the normalized value of the objective f_i and $Maxf_i$ and $Minf_i$ represent the upper and lower bounds of the objective f_i, respectively. However, it is difficult to find the upper and lower fronts of the two objectives in the MOHFGSP_CS. Two definitions for some extreme cases are given below.

Definition 1. *Consider the case in which a lot is split into several sublots, and one sublot accommodates only one unit; that is, as a unit from a lot is finished at a specific stage, it can be transported to the next stage. Let* $C_i^{\min}(C_i^{\max})$ *denote the completion time of the last sublot* T_j *of lot* j *that is processed at the fastest (/slowest) speed at each stage.*

Definition 2. *Consider the case in which a lot is split into several sublots, and one sublot accommodates only one unit; that is, as a unit from a lot is finished at a specific stage, it can be transported to the next stage. Let* $TEC_{i,j}^{low}$ *denote the total energy consumption of the machine at stage j processing sublot of lot j at the lowest speed.*

According to the above definitions, we can obtain approximations of the upper and lower bounds as follows.

$$\max(C_{\max}) = \sum_{i \in I} C_{j,1}^{\max} \tag{22}$$

$$\min(C_{\max}) = \max_{i \in I} \left\{ C_{j,T_j}^{\min} \right\} \tag{23}$$

$$\max(TEC) = \sum_{i \in I} \sum_{e \in E} \sum_{j \in J} TEC_{i,e,j}^{up} \tag{24}$$

$$\min(TEC) = \sum_{i \in I} \sum_{e \in E} \sum_{j \in J} TEC_{i,e,j}^{low} \tag{25}$$

4.4. VND-Based Employed Bee Phase

In the classical ABC, the role of the employed bees is to perform a local search for each solution. Therefore, the neighborhood structure plays an important role in enhancing the performance of the local search. The neighborhood structure contains many parts, and it is difficult to find an optimal solution using only a single neighborhood structure. Therefore, based on solution encoding, eight neighborhood structures are specially designed. For the lot sequence vector, two widely used neighborhood structures, namely, Lot insertion and Lot swap, are employed. For the lot split matrix, the neighborhood structure, namely, Lot split mutation, is designed. For the speed selection matrix, the neighborhood structure, namely, machine speed mutation, is designed. Moreover, four combined structures are also developed, which combine Lot insertion and Lot swap with Lot split mutation and Lot insertion, and Lot swap with machine speed mutation. They are detailed as follows:

(1) Lot insertion: from the lot permutation, many are randomly selected and inserted into a different randomly picked position; (2) Lot swap: from the lot sequence vector, two lots are randomly selected, and their positions are exchanged; (3) Lot split mutation: from the lot split matrix, a stage is selected randomly in which a lot with two or more sublots is randomly selected, and then two sublots are randomly selected. The random number in the distribution is $U[1, 5]$, reduced from the size of one sublot and added to another sublot size; (4) Machine speed mutation: from the machine speed selection matrix, a stage is selected randomly in which many sublots are randomly selected and the processing speed level of each sublot is changed into a different one; (5) Combined structure 1: first, Lot insertion is conducted, and then Lot split mutation is conducted; (6) Combined structure 2: first, Lot swap is conducted, and then Lot split mutation is conducted; (7) Combined structure 3: first, Lot insertion is conducted, and then Machine speed mutation is conducted; (8) Combined structure 4: first, Lot swap is conducted, and then Machine speed mutation is conducted.

To improve the utilization efficiency of the eight neighborhood structures, the variable neighborhood descent [35] (VND) strategy is applied to each subproblem. The essence of this approach is that a variety of neighborhood structures between the switch strategy are used to explore the solution space. First, a trial solution X_i is chosen from the population, where X_i denotes the ith solution. Then, a neighborhood solution X'_i is generated by using the kth neighborhood structure $N_k(k \in [1, \ldots, 8])$ of the solution S_i, where S_i denotes the ith neighborhood solution. The obtained solution X'_i will replace the solution X_i when the solution X'_i dominates the current solution X_i. Otherwise, the neighborhood structure is switched to the next one and a local search is performed. The procedure of the employed bee phase is displayed in Algorithm 1.

Algorithm 1. VND-based employed bee phase

1: **For** $i = 1$ to N **do**
2: Generate $X\prime_i$ by using a neighborhood S_i
3: Update the external population use $X\prime_i$
4: **If** $g\left(X\prime_i \middle| \lambda_i^{k_i}\right) < g\left(X_i \middle| \lambda_i^{k_i}\right)$ **then**
5: $X_i \leftarrow X\prime_i$; $S_i \leftarrow 1$;
6: **Else**
7: $X_i \leftarrow X_i$; $S_i \leftarrow S_i + 1$;
8: **End if**
9: **If** $N_k > 8$ **then**
10: $k \leftarrow 1$;
11: **End if**
12: **End for**

4.5. WAS-Based Onlooker Bee Phase

In the classical ABC, the role of the onlooker bees is selecting a good solution with a winning probability. To select good solutions, we employed the technique for order preference by similarity to an ideal solution (TOPSIS) and the binary tournament selection rule in this phase. First, two solutions are picked randomly, and the similarity to the ideal solution is measured based on TOPSIS. Then, a better solution is chosen.

The crossover operators aim to share the information with other solutions. According to the characteristics of the solution representation of MOHFGSP_CS. Position-based crossover (PBX) is employed in this part because it has a greater chance of exploring the unknown regions of the search space. This crossover operator for the lot sequence vector is depicted in Figure 5. For the machine speed selection and lot split matrix, because of the fixed lot size constraint, the PBX operator may result in an infeasible solution. Therefore, we proposed lot split matrix and machine speed selection matrix, as shown in Figures 6 and 7. Specifically, the split and select information are selected from the potential solutions with a specific probability P, and a large P value can make the offspring inherit more information from the potential solutions and is more likely to perform well.

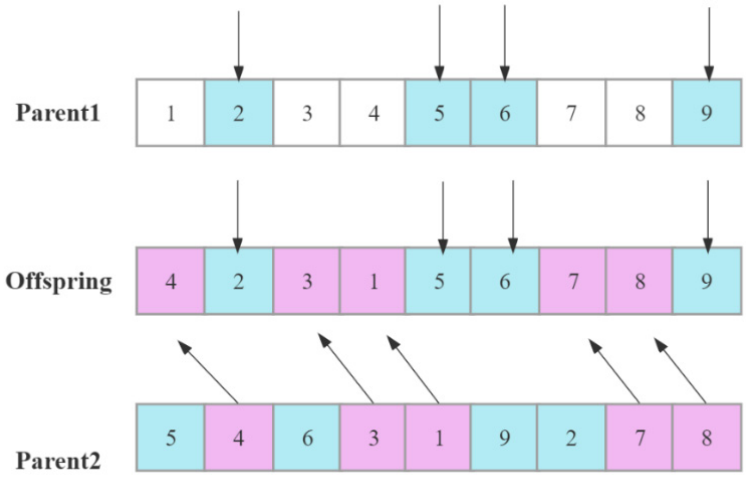


Figure 5. Illustration of PBX for the lot sequence.

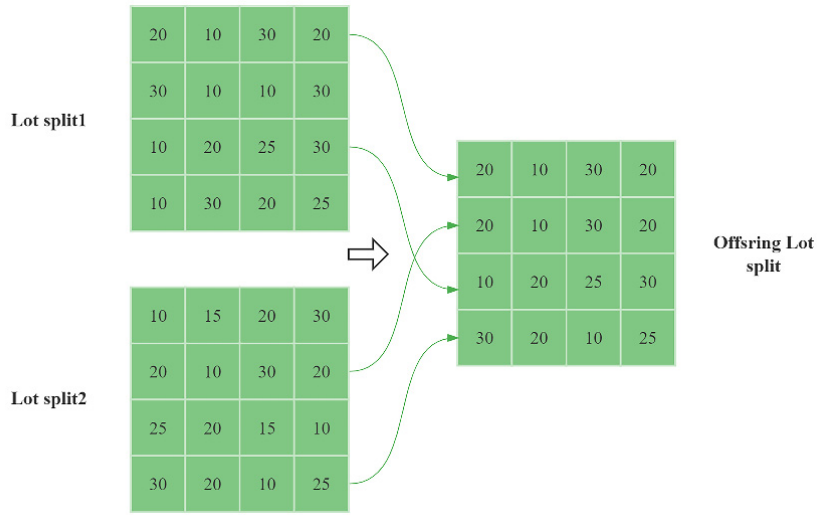


Figure 6. Illustration of lot split matrix.

In most MOEAs based on decomposition, uniformly distributed weight vectors have been designed, yet their performance on complex problems is still not ideal [36]. The intersection of the weight vector and the expected solution is not uniformly distributed [37]. To ensure that every individual moves towards unexplored areas and maintains the diversity of the population, we adopt the weight adjustment strategy (WAS) proposed by Liu [38], which is more suitable to characterize the true distribution of solutions in irregular PF. For the replacement of weight vectors, this strategy considers the solutions that have good convergence to some extent and uses the Euclidean distance to estimate the diversity. W is the pool of the weight vectors, λ is the weight vector of the selected individual, λ_p is the weight vector of individual P, λ_p is the weight vector of the neighbor individual, and k is the index of the weight vector nearest to individual P. λ must satisfy the following two conditions:

(1) $dis\tan ce(\lambda_p, \lambda_k) < dis\tan ce(\lambda, \lambda_k)$: the selected weight vector λ should ensure that the distance increased between λ_p and λ_k;

(2) $dis\tan ce(\lambda, \lambda_k) < dis\tan ce(\lambda, W)$: the selected weight vector λ is not very close to the other weight vectors in the current population.

First, if the selected weight vector satisfies the above two conditions, it will be stored in W, and its corresponding solution will be stored in set A, which is used to store the archive solutions. Then, the weight vector with the largest distance λ_p is selected for replacement. Finally, with each replacement operation completed, the distances of the archive members and the solutions in the population need to be updated separately. Given the above, the procedure of the WAS-based onlooker bee phase is displayed in Algorithm 2.

Algorithm 2. WAS-based onlooker bee phase

For $i = 1$ to N **do**
Select a promising solution X_a based on TOPSIS
Finding a partner solution X_b from X_a
Generate X_{child} by conducting PBX X_a and X_b
Update the external population use X_{child}
$W \leftarrow$ Generate N initial weights
  If $g(X_{child}|\lambda^a) < g(X_a|\lambda^a)$ **then**
    $X_a \leftarrow X_{child}$;
  Else
    $X_a \leftarrow X_a$;
  End if
For j = 1 to T **do**
  If $g\left(X_{child}\middle|\lambda^b\right) < g\left(X_b\middle|\lambda^b\right)$ **then**
    $X_b \leftarrow X_b$;
  End if
$\lambda_P = argmindistance(u, \lambda_P)$;
  If $distance(\lambda_P, \lambda_k) < distance(\lambda, \lambda_k)$ **and** $distance(\lambda, \lambda_k) < distance(\lambda, W)$
    $W \leftarrow \lambda$;
  End if
  $\lambda_P = argmaxdistance(u, \lambda_P)$;
End for
End for

4.6. SIS-Based Scout Bee Phase

In the scout bee phase, individuals are abandoned if they do not have improvements in consecutive generations. In basic ABC algorithms, an abandoned solution is replaced by a randomly generated solution within a predefined search scope. This process will reduce the searching efficiency and could worsen the random regeneration solution. For this reason, we present two solution interaction operators to regenerate a better individual: the solution-exchange operator and the solution-insert operator.

The solution-exchange operator: when a solution to its subproblem is not improved and dominates the neighboring solutions to one of its subproblems, the two subproblems will exchange their solutions. In particular, for the individual solution X_i that has not been improved after successive L cycles, we exchanged it with a neighborhood solution $\overline{X}_i^k$.

The solution-insert operator: when the neighboring solution falls into the local optimum, the insertion operator can retain a new individual solution. For the neighborhood solution, find two different positions and insert them. If the first position is pt1 smaller than the second position pt2, the neighborhood solution $\overline{X}_i^k$ is inserted after the first position; otherwise, it is inserted in front of the first position *pt1*.

The above two operations not only improve the search efficiency, but also reduce the computational complexity.

The pseudo steps of the scout bee phase are shown in Algorithm 3.

Algorithm 3. *SIS-based scout bee phase*

For $i = 1$ to N **do**
  If $L(X_i) > L$ **then**
    k$\leftarrow 1$;
    $isExchanged \leftarrow false$
  while $isExchanged \leftarrow false$ **then**
  If $X_i > \overline{X}_i^k$ **then**
    $X_i \longleftrightarrow \overline{X}_i^k$;
      $isExchanged \leftarrow true$
  Else
    k++;
  End if
  If $k > T$ **then**
    $pt_1 \leftarrow random[1, T]$; $pt_2 \leftarrow random[1, T]$;
    $\overline{X}_{pt_1+1}^k \leftarrow \overline{X}_{pt_1}^k$; $\overline{X}_{pt_1}^k \leftarrow \overline{X}_{pt_2}^k$;
      $insert \leftarrow true$;
  End if
  End while
  End if
End for

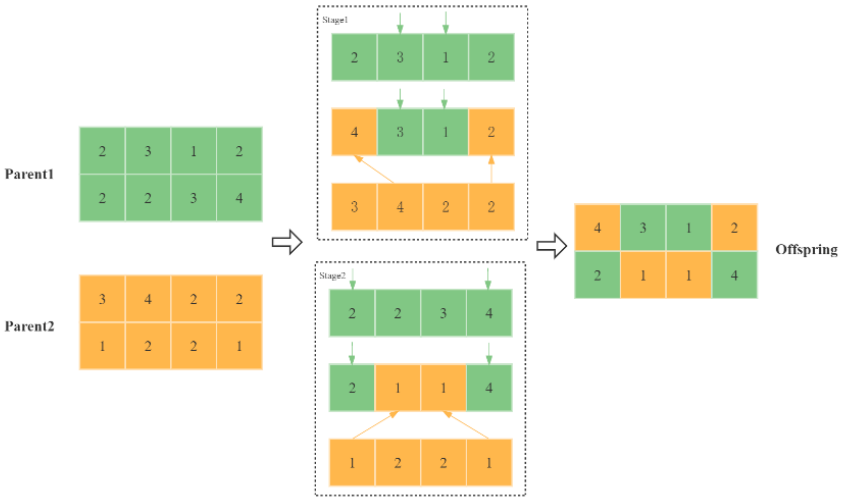


Figure 7. Illustration of the machine speed selection matrix.

5. Experiments and Results

In this section, we present computational experiments to demonstrate the effectiveness of the proposed multiobjective optimization model and the MDABC. Small-scale instances

are solved on Gurobi 9.5.0 by assigning 20 uniform weights in the range of [0, 1] to the two objectives. Thus, the solutions are obtained through 20 runs with the time limit set as 1800 seconds. For all the compared MOEAs, the maximum CPU elapsed time of $n \times \mathrm{m} \times t$ milliseconds is used for the termination criterion, where n is the number of lots, m is the number of stages, and t is a fixed value. This termination allows for more computational time for instances with a larger-scale. Here, we set t = 100 in our comprehensive experiments. All the algorithms are coded in C++ and run on a 3.10 GHz Intel Pentium processor.

5.1. Test Data

To systematically assess the behavior of the MDABC, two sets of instances with different scales are collected, including small-scale instances and large-scale instances [39], which are labeled with $n \times \mathrm{m}$. n represents the number of lots, and m represents the number of stages.

For small-scale instances, we generate the instances with $n \in \{2, 4, 6, 8, 10\}$ and $m \in \{2, 3, 4\}$, and this will lead to 15 different combinations by combining n and m. For each $n \times \mathrm{m}$, we have only one machine layout, resulting in 15 instances. Fifteen small instances are to be solved by the Gurobi and the MOEAs. For large-scale instances, we generate the instances with $n \in \{20, 40, 60, 80, 100\}$ and $m \in \{3, 5, 8, 10\}$. This will lead to 20 different combinations by combining n and m. For each $n \times \mathrm{m}$, we have four different machine layouts [7], resulting in 80 combinations. For each combination, five instances are generated. Thus, 400 instances are generated in total and to be solved by the MOEAs. The processing data are generated below. The number of units for each lot is obtained using a uniform distribution $U[50, 100]$. The unit processing time is randomly given by a uniform distribution $U[1, 10]$. The setup time and transportation time are, respectively, obtained from uniform distributions $U[50, 100]$ and $U[10, 20]$. In addition, the maximum sublot quantity is set as 30.

The above instances are then obtained by adding machine speed and energy consumption information [32]. The machine speed levels V_j at stage j is randomly given from a uniform distribution $U[1, 5]$. The processing time $p_{i,j,v}$ of lot i that is processing a speed level v at stage j can be calculated as $p_{i,j,v} = {}^{p_{i,j}}/_{c_{j,v}}$. The processing power is calculated by $pp_{j,v} = 4 \times c_{j,v}$. In addition, the setup power $sp_j = 2$, and the idle power $ip_j = 1$.

5.2. Performance Metrics

In this paper, four performance metrics are chosen to evaluate the obtained PFs of the algorithms [40]. They are the generational distance (GD), inverse GD (IGD), set cover (C_metric), and the number of nondominated solutions (N_metric), which are detailed as follows:

(1) GD: This metric measures how close between PF_{obtain} and PF_{true}, which can be defined as:

$$GD = \frac{1}{N}\sqrt{\sum_{i=1}^{N} d_i^2}$$

where N is the total number of solutions in PF_{obtain} and d_i denotes the Euclidean distance between the ith solution and its nearest solution in PF_{true}. Usually, a smaller GD value indicates better convergence of the algorithm. Note that a normalization method max–min for each objective is used in this metric;

(2) IGD: This metric is a comprehensive indicator that reflects both the convergence and diversity of the PFs, which can be formulated as:

$$IGD = \frac{1}{N^*}\sqrt{\sum_{i=1}^{N^*} d_i^{*2}}$$

where N^* is the number of solutions in PF_{true} and d_i^* represents the Euclidean distance between the ith solution and its nearest solution in PF_{obtain}. Similar to GD, a smaller IGD is preferred. Note that a normalization method max–min for each objective is used in this metric;

(3) C_metric: This metric directly reflects the quality of the obtained PF. $C(A, B)$ represents the percentage of solutions in B that are dominated by the solutions in A, which can be represented as:

$$C(A, B) = \frac{|\{m \in B | \forall n \in A, n \subseteq m|}{|B|}$$

Obviously, a large value of $C(A, B)$ and a small value of $C(B, A)$ can reflect that A has a better quality than B;

(4) N_metric [41]: This metric represents the number of nondominated solutions in PF_{obtain}. A large value can reflect the diversity of the obtained PF to some extent.

5.3. Parameter Setting

As we all know, the performance of algorithms is sensitive to parameter settings. Therefore, it is important to set the appropriate parameters. All the parameter settings of the MDABC considered in this study are listed below: the size of solutions in the population (N), the number of neighboring solutions in a neighborhood (T), and the number of consecutive generations in which a solution is not improved (R).

To gain empirical insight into the impact of the three parameters, the Taguchi method is applied [42]. For each parameter, four reasonable levels are listed in Table 3, which are determined in our preliminary experiments. For combing the parameter levels, the L_{16} orthogonal matrix is employed. In addition, the average IGD values (AVG) are collected as the response values, which are shown in Table 4. Based on Table 4, the factor level trend is displayed in Figure 8. From Table 4 and Figure 8, it is seen that a large N can facilitate the exploration capability, but it goes against a deep search of the subproblems under a limited termination criterion. A too small value for the parameter T may promote the collaboration of extremely similar solutions, which may hinder global exploration. While a too large value may lead to a lack of convergence. Regarding the parameter R, a small value may cause a solution that is not fully developed to be discarded, while a large value may cause a waste of computing resources. Based on the above description, the best parameter configuration for MDABC is determined, i.e., N = 200, T = 25, and R = 30.

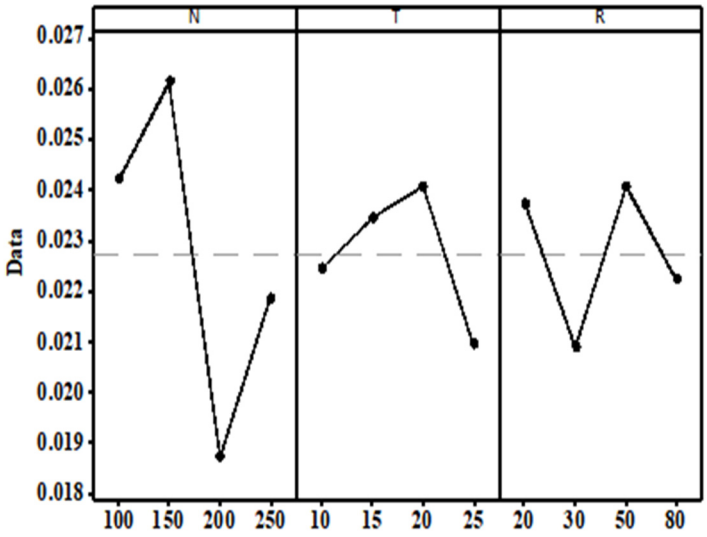


Figure 8. The trend of the factor level.

Table 3. Parameters and their levels.

Parameters	Parameter Level			
	1	**2**	**3**	**4**
N	100	150	200	250
T	10	15	20	25
R	20	30	50	80

Table 4. The orthogonal array and AVG values.

Test	Parameters			AVG
	N	**T**	**R**	
1	100	10	20	0.0238
2	100	15	30	0.0230
3	100	20	50	0.0295
4	100	25	80	0.0206
5	150	10	30	0.0237
6	150	15	20	0.0294
7	150	20	80	0.0254
8	150	25	50	0.0260
9	200	10	50	0.0193
10	200	15	80	0.0199
11	200	20	20	0.0201
12	200	25	30	0.0156
13	250	10	80	0.0230
14	250	15	50	0.0215
15	250	20	30	0.0213
16	250	25	20	0.0216

5.4. Evaluation of the Proposed Strategies

This subsection is intended to assess the effectiveness of the strategies proposed in this paper, including an adjustment weight strategy (WAS) and a solution interaction strategy (SIS). In this subsection, MDABC represents the whole algorithm, MDABC1 represents the MDABC without WAS strategy, and MDABC2 denotes the MDABC without SIS strategy. For a fair comparison environment, the three algorithms are conducted for each instance 20 times independently, and the parameter settings are the same as in Section 5.3. Table 5 shows the experimental results of MDABC, MDABC1, and MDABC2 in terms of IGD, where the AVG and the standard variance (SD) values are presented.

From Table 5, based on the AVG values, it is clear that for each problem, the average IGD value obtained by MDABC is much lower than that obtained by MDABC1 and MDABC2; that is, MDABC can yield the PFs with the best diversity and convergence for all of the problems. Concerning the SD values, when compared with MDABC1, MDABC can yield the smallest value for 16 out of 20 problems except for 20×5, 60×10, 80×8, and 80×10. When MDABC is compared with MDABC2, MDABC achieves smaller values for 15 problems except for 20×5, 40×3, 80×5, 100×3, and 100×10.

Thus, it can be concluded that MDABC has the best robustness. Overall, it is clear that MDABC outperforms MDABC1 and MDABC2. From the above observations, the validity of these two strategies is demonstrated. The reasons for the good performance of MDABC are as follows. The WAS strategy allows the solutions to be optimized in different directions, thereby improving the global exploration ability of the algorithm. The SIS strategy can improve the local exploitation of the algorithm.

Table 5. IGD mean values of the proposed strategies.

Problem	MDABC	$MDABC_1$	$MDABC_2$
20 × 3	0.0474(0.0256)	0.1853(0.0199)	0.2496(0.0267)
20 × 5	0.0509(0.0261)	0.1959(0.0204)	0.2450(0.0270)
20 × 8	0.0536(0.0229)	0.2085(0.0247)	0.2470(0.0299)
20 × 10	0.0541(0.0233)	0.2259(0.0317)	0.2515(0.0231)
40 × 3	0.0405(0.0210)	0.1840(0.0223)	0.2481(0.0207)
40 × 5	0.0563(0.0256)	0.2092(0.0268)	0.2637(0.0289)
40 × 8	0.0553(0.0229)	0.2399(0.0250)	0.3175(0.0283)
40 × 10	0.0611(0.0223)	0.2149(0.0226)	0.2731(0.0277)
60 × 3	0.0462(0.0236)	0.1942(0.0286)	0.2587(0.0226)
60 × 5	0.0500(0.0215)	0.2353(0.0240)	0.2945(0.0231)
60 × 8	0.0620(0.0262)	0.2466(0.0273)	0.3083(0.0272)
60 × 10	0.0586(0.0244)	0.2321(0.0221)	0.2897(0.0295)
80 × 3	0.0468(0.0225)	0.1921(0.0292)	0.2706(0.0292)
80 × 5	0.0552(0.0248)	0.2591(0.0272)	0.3299(0.0203)
80 × 8	0.0536(0.0231)	0.2487(0.0167)	0.3164(0.0299)
80 × 10	0.0725(0.0261)	0.2834(0.0219)	0.3609(0.0284)
100 × 3	0.0507(0.0227)	0.2259(0.0287)	0.2983(0.0211)
100 × 5	0.0557(0.0189)	0.2536(0.0230)	0.3336(0.0296)
100 × 8	0.0699(0.0236)	0.2736(0.0306)	0.3360(0.0245)
100 × 10	0.0680(0.0235)	0.2768(0.0289)	0.3569(0.0216)
Mean	0.0054(0.0243)	0.2292(0.0245)	0.2925(0.0250)

5.5. Comparison of the Proposed MDABC with Other Algorithms in Small-Scale Instances

To evaluate the performance of the proposed MDABC, we compare it with four state-of-the-art MOEAs in the literature, i.e., MOEA/D [43], NSGA-II [44], TMOA [30], and MOCGWO [45]. To our knowledge, they all have been proven to have excellent performance. MOEA/D and NSGA-II are two well-known algorithm frameworks that use different fitness evaluation methods. TMOA and MOCGWO are specially designed to solve energy-efficient HFSPs, and they all incorporated energy reduction strategies. Thus, these four algorithms can be well adapted to our problem. To create a fair environment, the parameter settings for all compared algorithms are the same as in Section 5.3.

To obtain the computational results for the small-scale instances, each algorithm is run independently 20 times for each instance. Tables 6–9 report AVG and SD values for small-scale instances based on GD, IGD, C-metric, and N-metric, respectively. From Table 6, based on the GD values, it can be seen that the Gurobi can acquire the best AVG values for the four smallest instances. The reason behind this is that the Gurobi can solve these instances to optimality within the limited time, and obviously can have the best convergence. With the increase in the number of lots and stages, it is difficult for Gurobi to obtain the optimal solutions under the time constraint of 1800 s. Regarding the MOEAs, MDABC obtained the smallest AVG values based on GD for these instances. From Table 7, based on the IGD values, the Gurobi performs better for the five instances. This is because IGD reflects not only convergence but also distribution, and the distribution of solutions obtained by means of weight allocation in a limited time is not very good. Regarding the MOEAs, MDABC obtained the smallest IGD values for these instances. Table 8 shows that for these instances, Gurobi does not obtain the best C-metric values. This can be reflected by the N-metric values given in Table 9, where Gurobi obtains the smallest values in all instances. For these two averages, the proposed MDABC once again performs best among the MOEAs. Regarding the SD values, the SD values obtained by the Gurobi are always smaller.

Table 6. AVG(SD) values for small-scale problems based on GD.

Problem	Gurobi	MDABC	MOEA/D	MOCGWO	TMOA	NSGA-II
2 × 2	0.0006(0.0143)	0.0180(0.0120)	0.0573(0.0080)	0.0765(0.0116)	0.0668(0.0092)	0.1306(0.0143)
2 × 3	0.0028(0.0140)	0.0089(0.0052)	0.0468(0.0061)	0.0617(0.0114)	0.0524(0.0082)	0.1078(0.0140)
2 × 4	0.0084(0.0168)	0.0092(0.0051)	0.0529(0.0068)	0.0645(0.0077)	0.0594(0.0080)	0.1184(0.01680
4 × 2	0.0009(0.0272)	0.0137(0.0085)	0.0779(0.0096)	0.0826(0.0122)	0.0828(0.0085)	0.1809(0.0272)
4 × 3	0.1871(0.0310)	0.0106(0.0065)	0.0823(0.0160)	0.0126(0.0126)	0.0862(0.0128)	0.1871(0.0310)
4 × 4	0.1720(0.0270)	0.0074(0.0043)	0.0771(0.0097)	0.0104(0.0104)	0.0816(0.0099)	0.1720(0.0270)
6 × 2	0.1933(0.0237)	0.0103(0.0068)	0.0884(0.0103)	0.0122(0.0122)	0.0934(0.0113)	0.1933(0.0237)
6 × 3	0.0249(0.0331)	0.0101(0.0062)	0.0922(0.0103)	0.0942(0.0131)	0.0985(0.0107)	0.2049(0.0331)
6 × 4	0.0107(0.0353)	0.0078(0.0042)	0.0908(0.0112)	0.0903(0.0121)	0.0982(0.0127)	0.2027(0.0353)
8 × 2	0.2307(0.0291)	0.0097(0.0061)	0.0899(0.0102)	0.0933(0.0108)	0.0964(0.0109)	0.2037(0.0291)
8 × 3	0.2067(0.0355)	0.0085(0.0052)	0.1019(0.0117)	0.1012(0.0132)	0.1106(0.0137)	0.2218(0.0355)
8 × 4	0.2044(0.0323)	0.0083(0.0056)	0.1084(0.0142)	0.1058(0.0144)	0.1167(0.0131)	0.2244(0.0291)
10 × 2	0.3060(0.0343)	0.0086(0.0053)	0.0970(0.0120)	0.1014(0.0169)	0.1067(0.0134)	0.2136(0.0355)
10 × 3	0.3086(0.0342)	0.0071(0.0044)	0.1018(0.0131)	0.0972(0.0162)	0.1064(0.0148)	0.2079(0.0323)
10 × 4	0.2062(0.0350)	0.0064(0.0037)	0.1072(0.0116)	0.1029(0.0187)	0.1142(0.0131)	0.2291(0.0342)
Mean	0.1978(0.0282)	0.0096(0.0059)	0.0848(0.0104)	0.0738(0.0129)	0.0914(0.0113)	0.1866(0.0528)

Table 7. AVG(SD) values for small-scale problems based on IGD.

Problem	Gurobi	MDABC	MOEA/D	MOCGWO	TMOA	NSGA-II
2 × 2	0.2230(0.0291)	0.1819(0.1037)	0.4281(0.0125)	0.4572(0.0115)	0.4454(0.0119)	0.4747(0.0149)
2 × 3	0.1444(0.0332)	0.0873(0.0478)	0.2908(0.0103)	0.3188(0.0121)	0.3036(0.0088)	0.3444(0.0132)
2 × 4	0.1630(0.0407)	0.0905(0.0494)	0.3195(0.0103)	0.3383(0.0108)	0.3275(0.0107)	0.3630(0.0107)
4 × 2	0.0408(0.0321)	0.0975(0.0526)	0.3916(0.0110)	0.3928(0.0161)	0.3987(0.0133)	0.4806(0.0167)
4 × 3	0.0426(0.0233)	0.0881(0.0448)	0.3920(0.0110)	0.3764(0.0145)	0.3928(0.0131)	0.4826(0.0211)
4 × 4	0.0718(0.0356)	0.0708(0.0373)	0.3920(0.0113)	0.3745(0.0151)	0.3953(0.0113)	0.4718(0.0165)
6 × 2	0.0998(0.0513)	0.0793(0.0452)	0.4036(0.0138)	0.3867(0.0209)	0.4044(0.0127)	0.4998(0.0185)
6 × 3	0.1318(0.0410)	0.0802(0.0445)	0.4339(0.0137)	0.4133(0.0203)	0.4387(0.0124)	0.5318(0.0207)
6 × 4	0.1328(0.0411)	0.0692(0.0393)	0.4347(0.0137)	0.4215(0.0180)	0.4409(0.0122)	0.5328(0.0203)
8 × 2	0.0446(0.0310)	0.0708(0.0423)	0.4414(0.0142)	0.4374(0.0178)	0.4499(0.0124)	0.5446(0.0181)
8 × 3	0.1801(0.0131)	0.0689(0.0415)	0.4711(0.0135)	0.4502(0.0201)	0.4790(0.0126)	0.5801(0.0228)
8 × 4	0.0056(0.0342)	0.0704(0.0384)	0.4973(0.0125)	0.4751(0.0197)	0.5021(0.0133)	0.6056(0.0231)
10 × 2	0.0603(0.0323)	0.0715(0.0414)	0.4578(0.0135)	0.4349(0.0213)	0.4633(0.0117)	0.5603(0.0170)
10 × 3	0.0845(0.0141)	0.0655(0.0376)	0.4834(0.0148)	0.4544(0.0194)	0.4858(0.0137)	0.5845(0.0219)
10 × 4	0.0979(0.0310)	0.0531(0.0309)	0.4933(0.0133)	0.4515(0.0182)	0.4988(0.0122)	0.5979(0.0200)
Mean	0.0513(0.0346)	0.0830(0.0464)	0.4220(0.0126)	0.4122(0.0170)	0.4284(0.0122)	0.5103(0.0184)

Table 8. AVG(SD) values for small-scale problems based on C-metric.

Problem	A:MDABC	B:MOEA/D	A:MDABC	B:MOCGWO
	C (A, B)	C (B, A)	C (A, C)	C (C, A)
2 × 2	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
2 × 3	1.000(0.000)	0.000(0.000)	1.000(0.001)	0.000(0.000)
2 × 4	0.998(0.004)	0.000(0.000)	1.000(0.000)	0.000(0.001)
4 × 2	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
4 × 3	0.999(0.001)	0.000(0.000)	1.000(0.000)	0.000(0.000)
4 × 4	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
6 × 2	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
6 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
6 × 4	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
8 × 2	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
8 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
8 × 4	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)

Table 8. *Cont.*

Problem	A:MDABC	B:MOEA/D	A:MDABC	B:MOCGWO		
	C (A, B)	C (B, A)	C (A, C)	C (C, A)		
10 × 2	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)		
10 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)		
10 × 4	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)		
Mean	0.999(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)		
Problem	**A: MDABC**	**B: TMOA**	**A: MDABC**	**B:N B:NSGA-II**	**A: MDABC**	**B:NB:Gurobi**
	C(A, D)	**C(D, A)**	**C(A, E)**	**C(E, A)**	**C(A, F)**	**C(F, A)**
2 × 2	0.998(0.004)	0.001(0.007)	1.000(0.000)	0.000(0.000)	0.970(0.036)	0.017(0.001)
2 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	0.984(0.019)	0.005(0.008)
2 × 4	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	0.981(0.020)	0.005(0.007)
4 × 2	0.999(0.002)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
4 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
4 × 4	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	0.994(0.014)	0.001(0.003)
6 × 2	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
6 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.016)	0.000(0.000)
6 × 4	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
8 × 2	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
8 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
8 × 4	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
10 × 2	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
10 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
10 × 4	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
Mean	0.999(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	0.995(0.006)	0.001(0.002)

Table 9. AVG(SD) values for small-scale problems based on N-metric.

Problem	Gurobi	MDABC	MOEA/D	MOCGWO	TMOA	NSGA-II
2 × 2	5.9(1.5)	92.3(32.1)	39.4(7.9)	24.8(5.0)	31.5(6.5)	16.5(1.8)
2 × 3	6.1(1.6)	117.8(31.5)	36.7(6.0)	24.8(5.4)	29.9(5.9)	11.2(2.9)
2 × 4	6.5(1.3)	112.4(23.7)	31.7(5.3)	24.6(4.9)	26.5(5.7)	11.5(1.9)
4 × 2	5.9(1.6)	87.2(23.1)	27.0(4.9)	23.2(4.5)	23.6(3.5)	13.7(3.7)
4 × 3	5.8(0.8)	86.3(31.9)	22.0(4.0)	19.1(4.1)	19.9(3.7)	19.7(2.8)
4 × 4	6.1(1.5)	96.4(20.9)	21.3(3.8)	18.6(4.0)	20.1(3.9)	17.2(1.3)
6 × 2	6.4(1.5)	90.2(23.5)	22.3(4.2)	20.5(4.3)	19.8(3.8)	12.3(2.2)
6 × 3	6.1(1.3)	80.8(20.8)	19.8(3.6)	16.1(3.7)	17.9(2.9)	25.2(2.0)
6 × 4	6.2(1.4)	99.4(29.4)	20.3(4.3)	16.8(3.5)	17.8(3.4)	14.7(3.3)
8 × 2	6.3(1.0)	100.5(23.5)	22.9(3.9)	19.1(3.2)	20.5(3.2)	35.7(2.2)
8 × 3	6.4(1.5)	89.2(23.0)	18.4(3.3)	16.3(3.2)	16.6(3.0)	13.2(3.4)
8 × 4	6.1(1.3)	99.4(19.5)	18.0(3.6)	14.8(3.3)	15.8(3.0)	11.8(2.3)
10 × 2	8.1(1.2)	86.2(17.6)	20.5(3.9)	16.8(3.9)	17.5(3.6)	19.3(3.2)
10 × 3	8.1(1.3)	111.6(24.9)	18.1(3.6)	16.2(3.7)	17.1(3.6)	36.5(3.9)
10 × 4	7.5(1.4)	109.2(20.5)	17.5(3.0)	14.3(3.2)	15.6(3.1)	18.0(2.9)
Mean	6.4(1.3)	97.2(24.3)	23.7(4.3)	19.0(4.0)	20.7(3.9)	18.4(2.6)

5.6. Comparison of the Proposed MDABC with Other Algorithms in Large-Scale Problems

For the large-scale instances, each algorithm is run independently 20 times for each instance, and then the results are averaged and grouped into the same scale. Tables 10–13 demonstrate the AVG and SD values for large-scale instances based on GD, IGD, C-metric, and N-metric, respectively. The results in Table 10 show that the GD-metric values of the MDABC are the lowest among all the algorithms. This means that the Pareto solutions obtained by the MDABC algorithm are closest to the Pareto solutions in the true PF. Table 11 shows the IGD-metric values of the MDABC. In terms of average IGD values, MDABC is the winner in each instance of all algorithms. From the IGD-metric of the SD values,

MDABC performs better than NSGA-II, TMOA, and MOCGWO, but worse than MOEA/D on 80 × 10 and 100 × 10. Similar to the GD values, we can further demonstrate the diversity and strong robustness of the solution generated by the MDABC. As presented in Table 12, the proposed MDABC performs outstandingly when testing 20 problems which receive the highest C-metric values. From the above observations, it can be concluded that the Pareto solutions of the other four algorithms are dominated by the Pareto solutions of the MDABC. From Table 13, we can observe that the MDABC is significantly superior to its competitors on almost all problems in terms of the N-metric.

Table 10. AVG(SD) values for large-scale problems based on GD.

Problem	MDABC	MOEA/D	MOCGWO	TMOA	NSGA-II
20 × 3	0.0059(0.0033)	0.0889(0.0011)	0.0716(0.0137)	0.0874(0.0105)	0.1829(0.0275)
20 × 5	0.0066(0.0037)	0.0901(0.0131)	0.0691(0.0112)	0.0863(0.0131)	0.1718(0.0229)
20 × 8	0.0073(0.0032)	0.0954(0.0133)	0.0741(0.0162)	0.0963(0.0144)	0.1883(0.0279)
20 × 10	0.0079(0.0040)	0.0968(0.0116)	0.0766(0.0185)	0.0978(0.0129)	0.2039(0.0329)
40 × 3	0.0047(0.0024)	0.0985(0.0109)	0.0837(0.0151)	0.0960(0.0116)	0.1812(0.0225)
40 × 5	0.0074(0.0040)	0.1088(0.0156)	0.0999(0.0179)	0.1104(0.0141)	0.2113(0.0294)
40 × 8	0.0080(0.0041)	0.1262(0.0194)	0.1187(0.0260)	0.1238(0.0185)	0.2357(0.0357)
40 × 10	0.0111(0.0047)	0.1118(0.0188)	0.1031(0.0190)	0.1078(0.0156)	0.2203(0.0364)
60 × 3	0.0065(0.0033)	0.1074(0.0122)	0.0978(0.0159)	0.1069(0.0131)	0.2045(0.0357)
60 × 5	0.0079(0.0038)	0.1098(0.0131)	0.1010(0.0150)	0.1112(0.0119)	0.2070(0.0315)
60 × 8	0.0108(0.0053)	0.1142(0.0156)	0.1093(0.0201)	0.1145(0.0179)	0.2127(0.0295)
60 × 10	0.0114(0.0051)	0.1074(0.0152)	0.1093(0.0211)	0.1122(0.0156)	0.2059(0.0333)
80 × 3	0.0067(0.0032)	0.1081(0.0134)	0.1018(0.0167)	0.1069(0.0121)	0.2062(0.0328)
80 × 5	0.0086(0.0044)	0.1159(0.0177)	0.1132(0.0224)	0.1140(0.0158)	0.2042(0.0272)
80 × 8	0.0095(0.0049)	0.1078(0.0157)	0.1112(0.0197)	0.1085(0.0130)	0.1989(0.0304)
80 × 10	0.0173(0.0079)	0.1409(0.0200)	0.1489(0.0307)	0.1355(0.0166)	0.2479(0.0425)
100 × 3	0.0068(0.0032)	0.1093(0.0148)	0.1044(0.0161)	0.1076(0.0136)	0.1993(0.0291)
100 × 5	0.0100(0.0045)	0.1165(0.0151)	0.1144(0.0187)	0.1142(0.0141)	0.2092(0.0269)
100 × 8	0.0141(0.0069)	0.1153(0.0178)	0.1249(0.0231)	0.1200(0.0187)	0.2189(0.0349)
100 × 10	0.0161(0.0078)	0.1307(0.0185)	0.1401(0.0241)	0.1281(0.0174)	0.2309(0.0410)
Mean	0.0092(0.0045)	0.1100(0.0151)	0.1037(0.0191)	0.1093(0.0145)	0.2071(0.0315)

Table 11. AVG(SD) values for large-scale problems based on IGD.

Problem	MDABC	MOEA/D	MOCGWO	TMOA	NSGA-II
20 × 3	0.0451(0.0244)	0.4209(0.0139)	0.3880(0.0169)	0.4192(0.0137)	0.5150(0.0153)
20 × 5	0.0484(0.0247)	0.4215(0.0127)	0.4025(0.0174)	0.4221(0.0112)	0.5202(0.0169)
20 × 8	0.0485(0.0215)	0.4100(0.0163)	0.3941(0.0200)	0.4144(0.0162)	0.5189(0.0202)
20 × 10	0.0502(0.0224)	0.4228(0.0165)	0.4066(0.0253)	0.4212(0.0185)	0.5376(0.0125)
40 × 3	0.0396(0.0206)	0.4549(0.0156)	0.4241(0.0183)	0.4498(0.0151)	0.5426(0.0167)
40 × 5	0.0519(0.0246)	0.4607(0.0173)	0.4429(0.0251)	0.4618(0.0179)	0.5641(0.0214)
40 × 8	0.0506(0.0227)	0.4964(0.0200)	0.4610(0.0243)	0.4933(0.0259)	0.6068(0.0229)
40 × 10	0.0563(0.0228)	0.4299(0.0173)	0.4174(0.0229)	0.4329(0.0189)	0.5434(0.0206)
60 × 3	0.0449(0.0231)	0.4750(0.0164)	0.4492(0.0221)	0.4715(0.0158)	0.5698(0.0208)
60 × 5	0.0472(0.0210)	0.4769(0.0179)	0.4640(0.0179)	0.4799(0.0150)	0.5770(0.0194)
60 × 8	0.0578(0.0253)	0.4791(0.0243)	0.4538(0.0274)	0.4714(0.0201)	0.5726(0.0183)
60 × 10	0.0556(0.0235)	0.4478(0.0169)	0.4321(0.0246)	0.4460(0.0211)	0.5489(0.0183)
80 × 3	0.0455(0.0227)	0.4705(0.0172)	0.4471(0.0193)	0.4652(0.0166)	0.5592(0.0171)
80 × 5	0.0540(0.0245)	0.4919(0.0202)	0.4825(0.0219)	0.4893(0.0223)	0.5874(0.0186)
80 × 8	0.0524(0.0228)	0.4578(0.0145)	0.4523(0.0214)	0.4533(0.0196)	0.5522(0.0203)
80 × 10	0.0692(0.0251)	0.5014(0.0188)	0.4858(0.0305)	0.4978(0.0307)	0.6025(0.0256)
100 × 3	0.0488(0.0222)	0.4778(0.0168)	0.4678(0.0221)	0.4793(0.0160)	0.5673(0.0176)
100 × 5	0.0544(0.0226)	0.4776(0.0168)	0.4680(0.0214)	0.4740(0.0186)	0.5677(0.0169)
100 × 8	0.0682(0.0309)	0.4647(0.0165)	0.4633(0.0253)	0.4619(0.0231)	0.5587(0.0198)
100 × 10	0.0641(0.0290)	0.4716(0.0168)	0.4630(0.0269)	0.4675(0.0241)	0.5686(0.0239)
Mean	0.0527(0.0239)	0.4605(0.0172)	0.4433(0.0226)	0.4586(0.0191)	0.5590(0.0196)

Table 12. AVG(SD) values for large-scale problems based on C-metric.

Problem	A: MDABC	B: MOEA/D	A: MDABC	B: MOCGWO	A: MDABC	B: TMOA	A: MDABC	B:N B:NSGA-II
	C(A, B)	C(B, A)	C(A, C)	C(C, A)	C(A, D)	C(D, A)	C(A, E)	C(E, A)
20 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
20 × 5	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
20 × 8	1.000(0.000)	0.000(0.000)	0.997(0.007)	0.001(0.002)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
20 × 10	1.000(0.000)	0.000(0.000)	0.998(0.005)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
40 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
40 × 5	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	0.999(0.003)	0.000(0.003)	1.000(0.000)	0.000(0.000)
40 × 8	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
40 × 10	1.000(0.000)	0.000(0.000)	0.996(0.010)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
60 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
60 × 5	1.000(0.000)	0.000(0.000)	0.999(0.002)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
60 × 8	1.000(0.000)	0.000(0.000)	0.993(0.010)	0.001(0.003)	0.999(0.003)	0.000(0.000)	1.000(0.000)	0.000(0.000)
60 × 10	0.996(0.008)	0.000(0.000)	0.986(0.031)	0.000(0.001)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
80 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
80 × 5	1.000(0.000)	0.000(0.000)	0.997(0.007)	0.000(0.001)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
80 × 8	1.000(0.000)	0.000(0.000)	0.997(0.006)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
80 × 10	0.999(0.002)	0.000(0.000)	0.998(0.006)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
100 × 3	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)
100 × 5	1.000(0.000)	0.000(0.000)	1.000(0.000)	0.000(0.000)	0.999(0.002)	0.000(0.000)	1.000(0.000)	0.000(0.000)
100 × 8	0.999(0.003)	0.000(0.000)	1.000(0.000)	0.000(0.000)	0.999(0.002)	0.000(0.000)	1.000(0.000)	0.000(0.000)
100 × 10	0.998(0.004)	0.000(0.001)	0.995(0.007)	0.003(0.007)	0.994(0.014)	0.002(0.019)	1.000(0.000)	0.000(0.000)
Mean	0.996(0.001)	0.000(0.000)	0.997(0.005)	0.007(0.001)	0.999(0.001)	0.001(0.001)	1.000(0.000)	0.000(0.000)

Table 13. AVG(SD) values for large-scale problems based on N-metric.

Problem	MDABC	MOEA/D	MOCGWO	TMOA	NSGA-II
20 × 3	114.0(12.2)	14.7(3.1)	12.8(2.6)	15.3(2.8)	5.8(1.4)
20 × 5	83.2(13.0)	13.1(3.1)	11.9(2.5)	14.0(3.3)	5.7(1.2)
20 × 8	62.7(9.9)	11.5(2.6)	9.8(2.2)	11.6(2.7)	5.2(1.2)
20 × 10	58.1(8.9)	11.7(2.3)	9.2(2.2)	11.4(2.3)	4.8(1.3)
40 × 3	90.2(12.6)	14.3(2.5)	11.2(2.2)	15.0(2.8)	6.0(1.1)
40 × 5	64.3(9.8)	12.4(2.9)	9.5(2.3)	12.1(2.7)	5.0(1.0)
40 × 8	47.0(7.8)	11.1(2.8)	7.9(2.2)	11.1(2.8)	4.8(1.2)
40 × 10	41.4(6.6)	10.9(2.6)	8.1(1.6)	11.3(2.5)	4.8(1.1)
60 × 3	78.7(13.7)	13.8(2.5)	9.9(2.4)	14.1(2.9)	5.6(1.6)
60 × 5	54.1(8.1)	12.6(2.4)	9.7(2.3)	12.4(2.3)	5.3(1.1)
60 × 8	40.3(6.2)	11.2(2.7)	8.4(2.0)	11.3(2.9)	5.2(0.8)
60 × 10	35.8(5.0)	11.4(2.2)	7.9(1.9)	10.0(2.2)	5.0(1.2)
80 × 3	65.2(9.0)	13.2(2.6)	10.3(2.7)	13.2(2.6)	5.3(1.4)
80 × 5	47.5(7.8)	12.5(3.0)	9.1(2.1)	12.7(2.5)	5.8(1.1)
80 × 8	37.8(6.3)	11.4(2.4)	7.9(2.0)	10.8(1.9)	5.1(1.4)
80 × 10	28.1(4.2)	9.6(1.9)	7.4(2.4)	10.2(1.9)	4.9(1.7)
100 × 3	60.0(8.9)	13.8(2.6)	9.8(2.1)	13.7(2.8)	5.8(1.4)
100 × 5	40.2(6.3)	11.6(2.5)	9.0(2.0)	11.8(2.3)	5.1(1.1)
100 × 8	31.4(4.6)	11.2(2.5)	7.8(1.7)	10.4(2.7)	5.0(1.4)
100 × 10	26.8(4.6)	10.1(2.7)	7.3(1.9)	9.8(2.2)	5.2(1.7)
Mean	55.3(7.8)	12.1(2.6)	9.2(2.2)	12.1(2.6)	5.3(1.3)

Figures 9–11 display the variance values with Tukey's HSD (honesty significant difference) at the 95% confidence level in large-scale instances. Figures 9–11 show that MDABC obtains the obviously lowest GD and IGD values and the largest C-metric and N-metric values among the five algorithms. Figures 12 and 13 present the convergence curve of the GD and IGD values for the increase of stages and jobs. Based on the GD values, we can see that as the number of stages increases, MDABC, MOEA/D, MOCGWO, and TMOA exhibit slight improvement; however, NSGA_II increases significantly, as shown in Figure 12a. In Figure 12b, MDABC remains unchanged with the increase in lots, and the other algorithms have an obvious corresponding change. Based on the IGD values, Figure 13a reveals that all algorithms remain relatively stable. Figure 13b shows that the performances of MOEA/D,

MOCGWO, TMOA, and NSGA_II have remarkable improvement with increasing lots. Meanwhile, MDABC remains stable with the growth of lots. From the above results, we can conclude that the proposed MDABC is highly competitive in terms of convergence, distribution, and stability.

To visualize the performance of MDABC and its peers, Figure 14 displays the approximation of Pareto fronts found by those algorithms for four representative problems. The horizontal coordinate represents the maximum makespan time, and the vertical coordinate represents the total energy consumption (TEC). As expected, the nondominated solutions obtained by the MDABC have better quantity and distribution, and the corresponding PF is lower than that of the other algorithms.

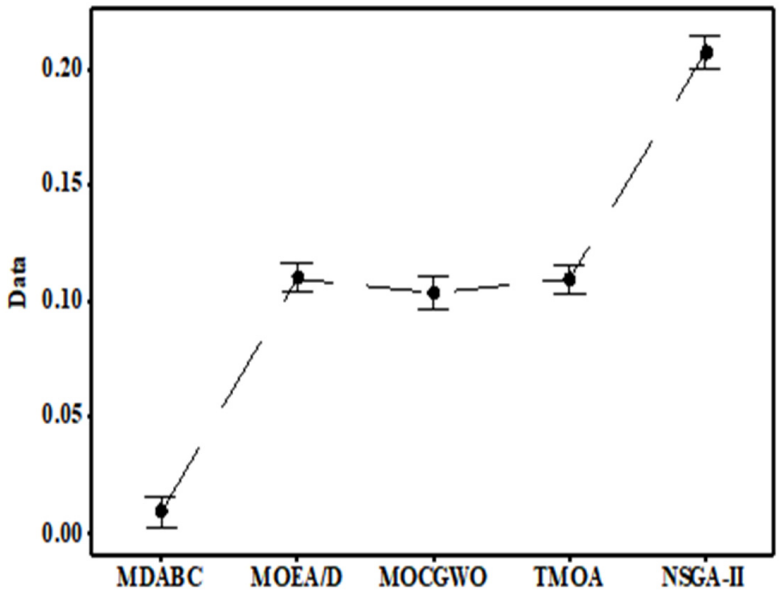


Figure 9. The mean plot and 95% confidence intervals on GD for large-scale problems.

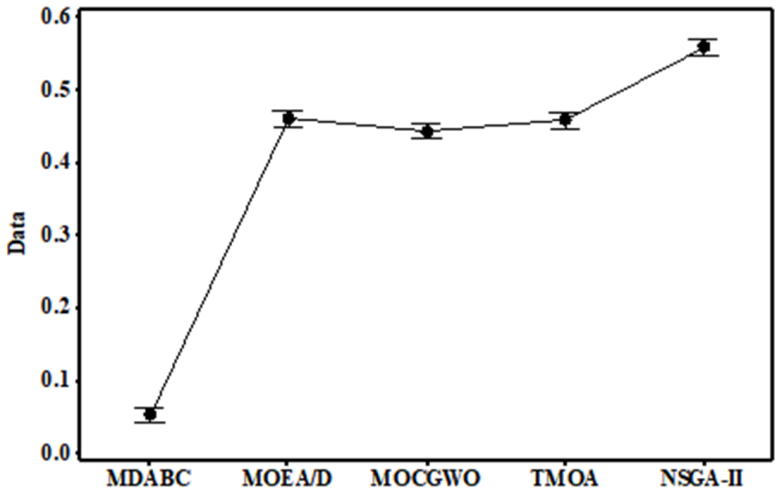


Figure 10. The mean plot and 95% confidence intervals on IGD for large-scale problems.

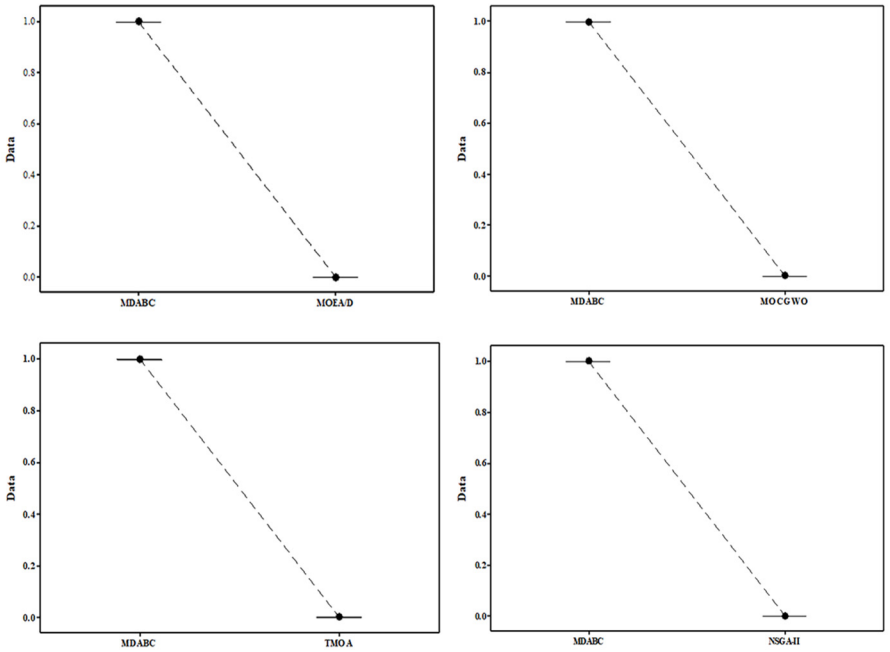
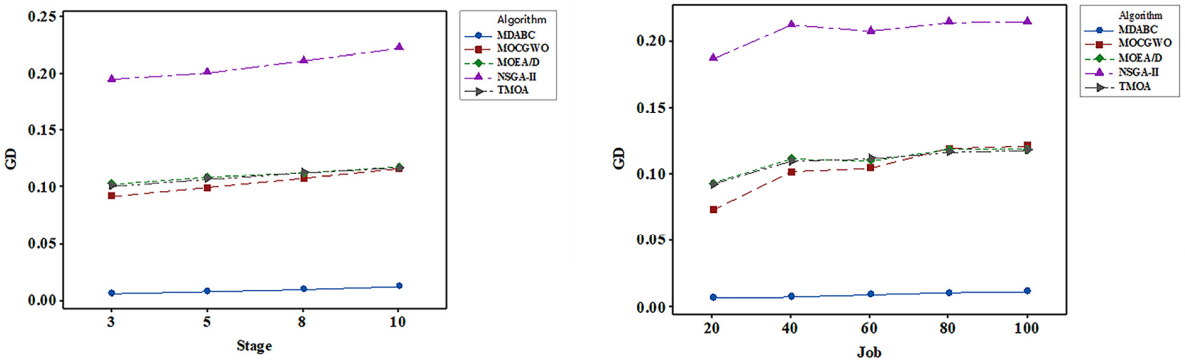
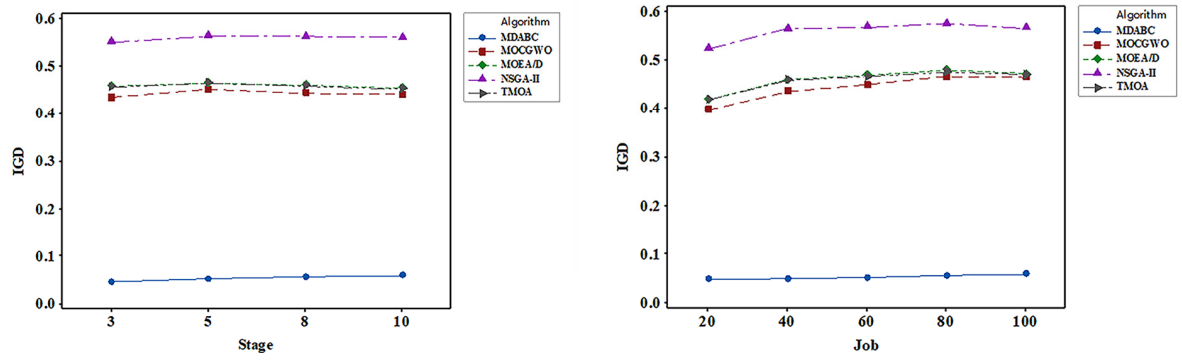


Figure 11. The mean plot and 95% confidence intervals on the C-metric for large-scale problems.



(**a**) (**b**)

Figure 12. Effect of the problem variables on the algorithm regarding GD for large-scale problems. (**a**) Effect of the Stage on MDABC regarding GD; (**b**) effect of the Job on MDABC regarding GD.



(**a**) (**b**)

Figure 13. Effect of the problem variables on the algorithm regarding IGD for large-scale problems. (**a**) Effect of the Stage on MDABC regarding IGD; (**b**) effect of the Job on MDABC regarding IGD.

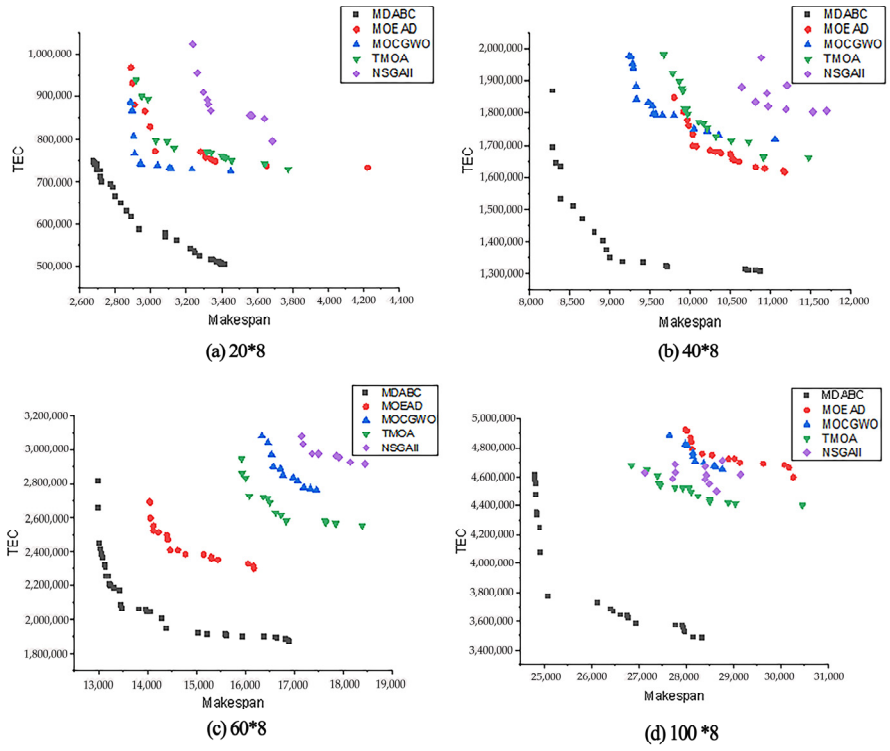


Figure 14. Pareto fronts obtained by the compared algorithms. (**a**) Pareto fronts in 20*8; (**b**) Pareto fronts in 40*8; (**c**) Pareto fronts in 60*8; (**d**) Pareto fronts in 100*8.

5.7. Experimental Analysis

This subsection aims to summarize the performance of Gurobi and MDABC based on four types of evaluation indicators on both small-scale and large-scale sets.

The results from Tables 6–9 indicate that the optimal solutions cannot be obtained on Gurobi with 1800 s under some weights. Thus, as the size of the problem increases, the MILP model cannot be optimally resolved in an acceptable time. From the above experimental results, it can be seen that the MOEAs are suitable for solving the MOHFGSP_CS, and MDABC performs well in various indicators among the compared algorithms.

From Tables 10 and 11, we can see that the AVG values based on GD and IGD are consistent on large-scale sets. It shows the best overall convergence performance and diversity performance on these instances. The reason for the good performance of MDABC can be explained as follows: first, the initialization and VND strategy can contribute to enhancing the exploration; second, the WAS strategy and the crossover operators help the algorithm maintain a compromising diversity: third, the SIS strategy prevented the algorithm from falling into a local optimum. It should be pointed out that MOEA/D, TMOA, MOCGWO, and NSGA-II seem better than MDABC from the view of IGD values. Since stability is difficult to guarantee when the number of Pareto solutions increases, it seems reasonable to obtain a larger variance. For further illustration, the AVG and SD values based on the N-metric are given in Table 13. Since the AVG values achieved by the MDABC are larger for most of the instances, the SD values are always larger. Figures 12 and 13 present the convergence curve of the GD and IGD values for the increase of lots and stages. Regarding the number of lots and stages, all algorithms obtain worse GD and IGD results with increasing values except MDABC. Overall, according to the above analysis, we can conclude that MDABC is effective and efficient in solving the MOHFGSP_CS.

6. Conclusions

This paper investigated an MOHFGSP_CS with a limited sublot quantity and various machine speeds for minimizing makespan and total energy consumption (TEC), which has a strong application background for green manufacturing. A multiobjective optimal

model was developed, and the trade-off between the two objectives was evaluated by using Gurobi. To solve this problem, the decomposition-based multiobjective MDABC algorithm is proposed, which has proven its effectiveness after comprehensive and in-depth calculations and statistical tests. In this MDABC, the variable neighborhood descent (VND) and eight neighborhood structures are designed for intensification, which ensures further exploitation of the unexplored areas. The weight adjustment strategy (WAS) for population diversification has been presented. The solution interaction strategy (SIS) has been produced for skipping the local optima. Furthermore, the MDABC parameters were calibrated by the Taguchi experimental design method. Finally, to demonstrate the validity of the strategy to solve the multiobjective HFSP_CS, we have conducted 400 instances. The experimental results reveal that the proposed MDABC has the ability to obtain PFs with good uniform distribution.

We will consider two directions in our future research. One is exploring more advanced optimization algorithms for MOHFGSP_CS. Specific practices are as follows: (1) combining the traditional mathematical programming method and the modern meta-heuristic algorithm, the accuracy and efficiency of the algorithm in the evolution process are studied; (2) introducing machine learning into a meta-heuristic algorithm to realize self-learning and self-adaptation to enhance the performance of the algorithm. In addition, we will apply advanced optimization algorithms to various fields. There are many different domains where advanced optimization algorithms have been applied as solution approaches, such as online learning and medicine. Online learning is a new field of scheduling problems [46]. The same strategy may behave differently when dealing with problems with various features when optimizing multiobjective problems. Therefore, analyzing problem features aids in the development of superior solutions. In actuality, the problem features are unknown when the optimization process is taking place. It is difficult in this situation to learn how to modify techniques to match the problem features. Large-scale and sudden public health events require hospitals to receive patients with high efficiency, and the problem of rescue vehicle routing is particularly prominent. Masoud Rabbani [47] proposed am MILP model according to the needs and characteristics of different patient groups to minimize the service completion time. Therefore, we will promote the application of MOHFGSP in various fields in future studies and verify the feasibility and effectiveness of the advanced algorithm combined with the needs of real-world MOHFGSP.

Author Contributions: W.W.: conceptualization, methodology, data curation, software, validation, writing—original draft preparation, writing—review and editing; B.Z.: provided methodological guidance, writing review, and funding support; B.J.: provided methodological guidance, writing review and funding support. All authors have read and agreed to the published version of the manuscript.

Funding: This research is partially supported by the Natural Science Foundation of Shandong Province (Grant No. ZR2021QF036, ZR2021QE195), National Science Foundation of China (Grant No. 52175490), Shandong Province Colleges and Universities Youth Innovation Talent Introduction and Education Program, China, the "Guangyue Young Scholar Innovation Team" of Liaocheng University (Grant No. LCUGYTD2022-03), and Discipline with Strong Characteristics of Liaocheng University—Intelligent Science and Technology (Grant No. 319462208).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Low, C.; Hsu, C.-J.; Su, C.-T. A two-stage hybrid flowshop scheduling problem with a function constraint and unrelated alternative machines. *Comput. Oper. Res.* **2008**, *35*, 845–853. [CrossRef]
2. Ruiz, R.; Vázquez-Rodríguez, J.A. The hybrid flow shop scheduling problem. *Eur. J. Oper. Res.* **2010**, *205*, 1–18. [CrossRef]

3. Pan, Q.-K.; Gao, L.; Li, X.-Y.; Gao, K.-Z. Effective metaheuristics for scheduling a hybrid flowshop with sequence-dependent setup times. *Appl. Math. Comput.* **2017**, *303*, 89–112. [CrossRef]
4. Ribas, I.; Leisten, R.; Framiñan, J.M. Review and classification of hybrid flow shop scheduling problems from a production system and a solutions procedure perspective. *Comput. Oper. Res.* **2010**, *37*, 1439–1454. [CrossRef]
5. Qin, W.; Zhang, J.; Song, D. An improved ant colony algorithm for dynamic hybrid flow shop scheduling with uncertain processing time. *J. Intell. Manuf.* **2015**, *29*, 891–904. [CrossRef]
6. Reiter, S. A System for Managing Job-Shop Production. *J. Bus.* **1966**, *39*, 371. [CrossRef]
7. Cheng, M.; Mukherjee, N.J.; Sarin, S.C. A review of lot streaming. *Int. J. Prod. Res.* **2013**, *51*, 7023–7046. [CrossRef]
8. Zhang, B.; Pan, Q.-K.; Meng, L.-L.; Zhang, X.-L.; Ren, Y.-P.; Li, J.-Q.; Jiang, X.-C. A collaborative variable neighborhood descent algorithm for the hybrid flowshop scheduling problem with consistent sublots. *Appl. Soft Comput.* **2021**, *106*, 107305. [CrossRef]
9. Fang, K.; Uhan, N.; Zhao, F.; Sutherland, J.W. A new approach to scheduling in manufacturing for power consumption and carbon footprint reduction. *J. Manuf. Syst.* **2011**, *30*, 234–240. [CrossRef]
10. Liu, Y.; Dong, H.; Lohse, N.; Petrovic, S. Reducing environmental impact of production during a Rolling Blackout policy—A multi-objective schedule optimisation approach. *J. Clean. Prod.* **2015**, *102*, 418–427. [CrossRef]
11. Meng, L.; Zhang, C.; Ren, Y.; Zhang, B.; Lv, C. Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem. *Comput. Ind. Eng.* **2020**, *142*, 106347. [CrossRef]
12. Meng, L.; Gao, K.; Ren, Y.; Zhang, B.; Sang, H.; Chaoyong, Z. Novel MILP and CP models for distributed hybrid flowshop scheduling problem with sequence-dependent setup times. *Swarm Evol. Comput.* **2022**, *71*, 101058. [CrossRef]
13. Gupta, J.N. Two-stage, hybrid flowshop scheduling problem. *J. Oper. Res. Soc.* **1988**, *39*, 359–364. [CrossRef]
14. Yang, J. A new complexity proof for the two-stage hybrid flow shop scheduling problem with dedicated machines. *Int. J. Prod. Res.* **2010**, *48*, 1531–1538. [CrossRef]
15. Guirchoun, S.; Martineau, P.; Billaut, J.-C. Total completion time minimization in a computer system with a server and two parallel processors. *Comput. Oper. Res.* **2005**, *32*, 599–611. [CrossRef]
16. Cheng, M.; Sarin, S.C. Two-stage, Multiple-lot, Lot Streaming Problem for a 1 + 2 Hybrid Flow Shop. *IFAC Proc. Vol.* **2013**, *46*, 448–453. [CrossRef]
17. Zhang, W.; Liu, J.; Linn, R.J. Model and heuristics for lot streaming of one job in M-1 hybrid flowshops. *Int. J. Oper. Quant. Manag.* **2003**, *9*, 49–64.
18. Kim, J.-S.; Kang, S.-H.; Lee, S.M. Transfer batch scheduling for a two-stage flowshop with identical parallel machines at each stage. *Omega* **1997**, *25*, 547–555. [CrossRef]
19. Nejati, M.; Mahdavi, I.; Hassanzadeh, R.; Mahdavi-Amiri, N.; Mojarad, M. Multi-job lot streaming to minimize the weighted completion time in a hybrid flow shop scheduling problem with work shift constraint. *Int. J. Adv. Manuf. Technol.* **2013**, *70*, 501–514. [CrossRef]
20. Lalitha, J.L.; Mohan, N.; Pillai, V.M. Lot streaming in [N − 1](1) + N (m) hybrid flow shop. *J. Manuf. Syst.* **2017**, *44*, 12–21. [CrossRef]
21. Nejati, M.; Mahdavi, I.; Hassanzadeh, R.; Mahdavi-Amiri, N. Lot streaming in a two-stage assembly hybrid flow shop scheduling problem with a work shift constraint. *J. Ind. Prod. Eng.* **2016**, *33*, 459–471. [CrossRef]
22. Dai, M.; Tang, D.; Giret, A.; Salido, M.A.; Li, W.D. Energy-efficient scheduling for a flexible flow shop using an improved genetic-simulated annealing algorithm. *Robot. Comput. Manuf.* **2013**, *29*, 418–429. [CrossRef]
23. Fernandez-Viagas, V.; Prata, B.; Framinan, J.M. A critical-path based iterated local search for the green permutation flowshop problem. *Comput. Ind. Eng.* **2022**, *169*, 108276. [CrossRef]
24. Gu, W.; Li, Z.; Dai, M.; Yuan, M. An energy-efficient multi-objective permutation flow shop scheduling problem using an improved hybrid cuckoo search algorithm. *Adv. Mech. Eng.* **2021**, *13*, 16878140211023603. [CrossRef]
25. Lu, C.; Huang, Y.; Meng, L.; Gao, L.; Zhang, B.; Zhou, J. A Pareto-based collaborative multi-objective optimization algorithm for energy-efficient scheduling of distributed permutation flow-shop with limited buffers. *Robot. Comput. Manuf.* **2021**, *74*, 102277. [CrossRef]
26. Meng, L.; Zhang, B.; Gao, K.; Duan, P. An MILP Model for Energy-Conscious Flexible Job Shop Problem with Transportation and Sequence-Dependent Setup Times. *Sustainability* **2023**, *15*, 776. [CrossRef]
27. Bai, J.; Liu, H. Multi-objective artificial bee algorithm based on decomposition by PBI method. *Appl. Intell.* **2016**, *45*, 976–991. [CrossRef]
28. Pan, Q.-K.; Gao, L.; Wang, L.; Liang, J.; Li, X.-Y. Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem. *Expert Syst. Appl.* **2019**, *124*, 309–324. [CrossRef]
29. Yan, J.; Li, L.; Zhao, F.; Zhang, F.; Zhao, Q. A multi-level optimization approach for energy-efficient flexible flow shop scheduling. *J. Clean. Prod.* **2016**, *137*, 1543–1552. [CrossRef]
30. Zhang, B.; Pan, Q.-K.; Gao, L.; Meng, L.-L.; Li, X.-Y.; Peng, K.-K. A Three-Stage Multiobjective Approach Based on Decomposition for an Energy-Efficient Hybrid Flow Shop Scheduling Problem. *IEEE Trans. Syst. Man Cybern. Syst.* **2019**, *50*, 4984–4999. [CrossRef]
31. Ding, J.-Y.; Song, S.; Wu, C. Carbon-efficient scheduling of flow shops by multi-objective optimization. *Eur. J. Oper. Res.* **2016**, *248*, 758–771. [CrossRef]

32. Zhang, B.; Pan, Q.-K.; Gao, L.; Li, X.-Y.; Meng, L.-L.; Peng, K.-K. A multiobjective evolutionary algorithm based on decomposition for hybrid flowshop green scheduling problem. *Comput. Ind. Eng.* **2019**, *136*, 325–344. [CrossRef]
33. Worasan, K.; Sethanan, K.; Pitakaso, R.; Moonsri, K.; Nitisiri, K. Hybrid particle swarm optimization and neighborhood strategy search for scheduling machines and equipment and routing of tractors in sugarcane field preparation. *Comput. Electron. Agric.* **2020**, *178*, 105733. [CrossRef]
34. Zhang, Q.; Li, H. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Trans. Evol. Comput.* **2007**, *11*, 712–731. [CrossRef]
35. Peng, K.; Pan, Q.-K.; Gao, L.; Li, X.; Das, S.; Zhang, B. A multi-start variable neighbourhood descent algorithm for hybrid flowshop rescheduling. *Swarm Evol. Comput.* **2019**, *45*, 92–112. [CrossRef]
36. Ma, X.; Yu, Y.; Li, X.; Qi, Y.; Zhu, Z. A Survey of Weight Vector Adjustment Methods for Decomposition-Based Multiobjective Evolutionary Algorithms. *IEEE Trans. Evol. Comput.* **2020**, *24*, 634–649. [CrossRef]
37. Zhou, J.; Yao, X.; Chan, F.T.; Gao, L.; Jing, X.; Li, X.; Lin, Y.; Li, Y. A decomposition based evolutionary algorithm with direction vector adaption and selection enhancement. *Inf. Sci.* **2019**, *501*, 248–271. [CrossRef]
38. Liu, Y.; Hu, Y.; Zhu, N.; Li, K.; Zou, J.; Li, M. A decomposition-based multiobjective evolutionary algorithm with weights updated adaptively. *Inf. Sci.* **2021**, *572*, 343–377. [CrossRef]
39. Gharib, Z.; Yazdani, M.; Bozorgi-Amiri, A.; Tavakkoli-Moghaddam, R.; Taghipourian, M.J. Developing an integrated model for planning the delivery of construction materials to post-disaster reconstruction projects. *J. Comput. Des. Eng.* **2022**, *9*, 1135–1156. [CrossRef]
40. Lu, C.; Gao, L.; Li, X.; Pan, Q.; Wang, Q. Energy-efficient permutation flow shop scheduling problem using a hybrid multi-objective backtracking search algorithm. *J. Clean. Prod.* **2017**, *144*, 228–238. [CrossRef]
41. Gharib, Z.; Tavakkoli-Moghaddam, R.; Bozorgi-Amiri, A.; Yazdani, M. Post-Disaster Temporary Shelters Distribution after a Large-Scale Disaster: An Integrated Model. *Buildings* **2022**, *12*, 414. [CrossRef]
42. Zhang, B.; Pan, Q.-K.; Gao, L.; Zhang, X.-L.; Peng, K.-K. A multi-objective migrating birds optimization algorithm for the hybrid flowshop rescheduling problem. *Soft Comput.* **2018**, *23*, 8101–8129. [CrossRef]
43. Zhang, B.; Pan, Q.-K.; Meng, L.-L.; Lu, C.; Mou, J.-H.; Li, J.-Q. An automatic multi-objective evolutionary algorithm for the hybrid flowshop scheduling problem with consistent sublots. *Knowl. Based Syst.* **2022**, *238*, 107819. [CrossRef]
44. Chen, T.-L.; Cheng, C.-Y.; Chou, Y.-H. Multi-objective genetic algorithm for energy-efficient hybrid flow shop scheduling with lot streaming. *Ann. Oper. Res.* **2018**, *290*, 813–836. [CrossRef]
45. Lu, C.; Gao, L.; Pan, Q.; Li, X.; Zheng, J. A multi-objective cellular grey wolf optimizer for hybrid flowshop scheduling problem considering noise pollution. *Appl. Soft Comput.* **2019**, *75*, 728–749. [CrossRef]
46. Zhao, H.Z.; Chang, S. An online-learning-based evolutionary many-objective algorithm. *Inf. Sci.* **2019**, *509*, 1–21. [CrossRef]
47. Rabbani, M.; Oladzad-Abbasabady, N.; Akbarian-Saravi, N. Ambulance routing in disaster response considering variable patient condition: NSGA-II and MOPSO algorithms. *J. Ind. Manag. Optim.* **2022**, *18*, 1035. [CrossRef]