TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 06/15 pp646–663
DOI: 10.26599/TST.2021.9010007
Volume 26, Number 5, October 2021

Decomposition-Based Multi-Objective Optimization for Energy-Aware Distributed Hybrid Flow Shop Scheduling with Multiprocessor Tasks

Enda Jiang, Ling Wang*, and Jingjing Wang

Abstract: This paper addresses the Energy-Aware Distributed Hybrid Flow Shop Scheduling Problem with Multiprocessor Tasks (EADHFSPMT) by considering two objectives simultaneously, i.e., makespan and total energy consumption. It consists of three sub-problems, i.e., job assignment between factories, job sequence in each factory, and machine allocation for each job. We present a mixed inter linear programming model and propose a Novel Multi-Objective Evolutionary Algorithm based on Decomposition (NMOEA/D). We specially design a decoding scheme according to the characteristics of the EADHFSPMT. To initialize a population with certain diversity, four different rules are utilized. Moreover, a cooperative search is designed to produce new solutions based on different types of relationship between any solution and its neighbors. To enhance the quality of solutions, two local intensification operators are implemented according to the problem characteristics. In addition, a dynamic adjustment strategy for weight vectors is designed to balance the diversity and convergence, which can adaptively modify weight vectors according to the distribution of the non-dominated front. Extensive computational experiments are carried out by using a number of benchmark instances, which demonstrate the effectiveness of the above special designs. The statistical comparisons to the existing algorithms also verify the superior performances of the NMOEA/D.

Key words: distributed hybrid flow shop; multiprocessor tasks; energy-aware scheduling; multi-objective optimization; decomposition; dynamic adjustment strategy

1 Introduction

The Hybrid Flow Shop Scheduling Problem with Multiprocessor Tasks (HFSPMT) is an extension of traditional Hybrid Flow Shop Scheduling Problem (HFSP). The HFSPMT commonly exists in bio-process industry, textile industry, and electronics industry[1], in which each job must be processed by several machines simultaneously in every stage. During the past few years, it has gained much attention. Hidri and Gharbi[2] presented an efficient destructive lower bound to minimize makespan.

For the same objective, i.e., makespan, Hidri[3] presented a new valid lower bound. Kurdi[4] proposed an Ant Colony Optimization (ACO) algorithm with a Non-DaemonActions procedure. Gholami et al.[5] presented an elephant herding optimization algorithm. Recently, Gholami and Rezvan[6] presented a memetic algorithm with a dynamic adjustment structure.

For traditional shop scheduling problems, usually only single factory is taken into consideration. With the development of economy and globalization, the manufacturing systems have transformed from one factory to multiple factories. Distributed production has become a great trend. Unlike traditional single factory scheduling problem, the distributed shop scheduling problem needs to consider both job assignment between different factories and job sequence in each factory. Therefore, it is more significant for reality but more complex to solve. For distributed flow shop scheduling problem, Ruiz et al.[7] proposed an improved iterated

• Enda Jiang, Ling Wang, and Jingjing Wang are with the Department of Automation, Tsinghua University, Beijing 100084, China. E-mail: jed16@mails.tsinghua.edu.cn; wangling@tsinghua.edu.cn; wjj18@mails.tsinghua.edu.cn.
* To whom correspondence should be addressed.
Manuscript received: 2020-12-28; accepted: 2021-01-26

greedy algorithm to minimize makespan. For the distributed permutation flow shop, Pan et al.[8] presented three heuristics and four metaheuristics to minimize total flow time. For the distributed fuzzy blocking flow shop scheduling problem, Shao et al.[9] proposed two constructive heuristics and two metaheuristics based on iterated greedy method to minimize fuzzy makespan. Also for makespan objective, J. J. Wang and L. Wang[10] proposed a bi-population cooperative memetic algorithm for the distributed hybrid flow shop scheduling problem. Chaouch et al.[11, 12] proposed a modified ACO and a hybrid ACO combined with local search for the distributed job shop scheduling problem. Hsu et al.[13] proposed an agent-based fuzzy constraint-directed negotiation mechanism scheduling for the distributed job shop scheduling problem. Meng et al.[14] proposed four different mathematical models and a constraint programming for the distributed flexible job shop scheduling problem. Recently, Ying and Lin[1] extended the HFSPMT to the distributed environment and proposed a self-tuning iterated greedy algorithm to minimize makespan. For the same problem, Cai et al.[15] proposed a Dynamic Shuffled Frog-Leaping Algorithm (DSFLA). However, the literature on the Distributed Hybrid Flow Shop Problem with Multiprocessor Tasks (DHFSPMT) is still rare.

In realistic production, the decision makers need to consider some other objectives simultaneously. However, the above research works are all about economic criteria, like makespan or total tardiness. As the environment protection has raised much attention, both economic benefits and environmental criteria should be taken into account simultaneously. Thus, green scheduling is more practical and has attracted much interest. For the no-wait permutation flow shop scheduling problem, Wu and Che[16] proposed an adaptive multi-objective variable neighborhood search to minimize both makespan and total energy consumption. For the hybrid flow shop scheduling problem, Liu et al.[17] proposed an evolutionary algorithm based on weighted sum approach to minimize both makespan and total energy consumption. For the distributed permutation flow shop scheduling problem, J. J. Wang and L. Wang[18] proposed a knowledge-based cooperative algorithm to minimize both makespan and energy consumption. For flexible job shop scheduling problem with variable processing speed, Li et al.[19] proposed an improved artificial bee colony algorithm to minimize makespan, machine loading, and total carbon emission. For flexible job shop scheduling problem under time-of-use electricity prices, Jiang and Wang[20] proposed a Modified multi-Objective Evolutionary Algorithm (MOEA/D) to minimize both makespan and total electricity cost. For the distributed job shop scheduling problem, Jiang et al.[21] proposed a collaborative MOEA/D to minimize both makespan and total energy consumption. Recently, for the distributed parallel machines scheduling problem, Pan et al.[22] proposed a knowledge-based two-population optimization algorithm to minimize both total energy consumption and total tardiness.

This paper is the first to address the Energy-Aware Distributed Hybrid Flow Shop Scheduling Problem with Multiprocessor Tasks (EADHFSPMT) to minimize both makespan and total energy consumption. Since the HFSPMT with single objective has proved to be NP-hard[1], the EADHFSPMT is obviously a more complex NP-hard problem. Thus, those methods based on mathematical programming are not practical to solve the large-scale problems. Evolutionary algorithms have been proved to be effective in solving a variety of large-scale optimization problems in different fields[23–26]. For the multi-objective optimization problems, Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D)[27] is an effective technique with many successful applications[20, 21]. In this paper, we will propose a Novel MOEA/D (NMOEA/D) with some special designs according to characteristics of the EADHFSPMT. To be specific, the decoding scheme is designed based on the problem-specific features. Four initialization heuristics are utilized and two local intensification operators are performed to improve the quality of solutions. Moreover, a dynamic adjustment strategy is proposed to adaptively adjust the weight vectors according to the distribution of solutions in each generation. Extensive numerical tests demonstrate the effectiveness of these modifications, and it also shows that the proposed algorithm is able to obtain better solutions than the existing algorithms.

The rest of the paper is organized as follows. Section 2 presents the description and the formulation of the EADHFSPMT. Section 3 introduces the detail design of the NMOEA/D. The numerical results and comparisons of different algorithms are given in Section 4. Finally, we end the paper with conclusions and future work in Section 5.

2 Problem Description and Formulation

The parameters and variables used for problem formulation are listed as follows:

n: number of jobs.

i: index of jobs, $i = 0, 1, 2, \ldots, n$, where job 0 is the dummy initial job.

f: number of identical factories.

k: index of factories, $k = 1, 2, \ldots, f$.

s: number of stages in each factory.

j: index of stages, $j = 1, 2, \ldots, s$.

m_j: number of machines of each stage.

g: index of machines, $g = 1, 2, \ldots, m_j$.

$M_{j,g}$: machine g at stage j.

$r_{i,j}$: number of machines required to process job i at stage j.

$p_{i,j}$: processing time of job i at stage j.

$e_{j,g,k}$: energy consumption factor of the machine g at stage j in factory k when processing.

$\text{ie}_{j,g,k}$: energy consumption factor of the machine g at stage j in factory k when at idle time, where $\text{ie}_{j,g,k} = e_{j,g,k}/4$.

t: index of time slots, $t = 0, \ldots, L$.

L: time horizon which is large enough for makespan.

$S_{i,j}$: starting time of job i in stage j.

$C_{i,j}$: completion time of job i in stage j.

$x_{i,k}$: binary variable that equals to 1 if job i is assigned in factory k, otherwise it equals to 0.

$y_{i,j,t}$: binary variable that equals to 1 if job i is processed in stage j at time slot t, otherwise it equals to 0.

$z_{i,j,g,k}$: binary variable that equals to 1 if job i occupies machine g at stage j in factory k, otherwise it equals to 0.

T_j (id, ih): required time to schedule job id and job ih in stage j.

$C_{\max}$: maximum completion time (makespan) of all factories.

TEC: total energy consumption.

The EADHFSPMT can be described as follows. There are n jobs to be processed in f identical factories. Each factory is a hybrid flow shop with s stages where stage j consists of m_j machines. Each job follows the same route through s stages in the assigned factory and switching jobs between factories is forbidden. Each job must be processed by several machines simultaneously in every stage. For job i at stage j, $r_{i,j}$ denotes the number of machines required by this job, and $p_{i,j}$ denotes its processing time. Each machine can process at most one job at one time and the processing cannot be interrupted. The energy consumption factors of machines are different, which means processing one job on different machines in the same stage may consume different energy. The energy consumption consists of two parts: processing energy and idle energy. The objectives are to assign each job to the factories, determine their suitable sequence in the factory, and allocate machines for each job, so as to minimize both the maximum completion time of all jobs (makespan) and the TEC.

According to the Mixed Inter Linear Programming (MILP) model[1] for the DHFSPMT, we present the following MILP model for the EADHFSPMT.

$$\min(C_{\max}, \text{TEC}) \tag{1}$$

Subject to

$$C_{i,j} = S_{i,j} + p_{i,j}, \forall i, j \tag{2}$$

$$C_{i-1,j} \leqslant C_{i,j} - p_{i,j}, \forall i > 1, j \tag{3}$$

$$\sum_{k=1}^{f} x_{i,k} = 1, \forall i \tag{4}$$

$$\sum_{i=1}^{N} x_{i,k} y_{i,j,t} r_{i,j} \leqslant m_j, \forall j, k, t \tag{5}$$

$$\sum_{t=0}^{L} y_{i,j,t} = p_{i,j}, \forall i, j \tag{6}$$

$$S_{i,j} \leqslant t + L(1 - y_{i,j,t}), \forall i, j, t \tag{7}$$

$$C_{i,j} \geqslant t y_{i,j,t}, \forall i, j, t \tag{8}$$

$$C_{\max} \geqslant C_{i,s}, \forall i \tag{9}$$

$$\sum_{k=1}^{f} \sum_{g=1}^{m_j} z_{i,j,g,k} x_{i,k} = r_{i,j}, \forall i, j \tag{10}$$

$$C^{j,k} \geqslant C_{i,j} x_{i,k}, \forall i, j, k \tag{11}$$

$$S^{j,k} \leqslant S_{i,j} x_{i,k}, \forall i, j, k \tag{12}$$

$$\text{TEC} = \sum_{k=1}^{f} \sum_{j=1}^{s} \sum_{g=1}^{m_j} \left[\text{ie}_{j,g,k} (C^{j,k} - S^{j,k}) + \sum_{i=1}^{N} z_{i,j,g,k} p_{i,j} (e_{j,g,k} - \text{ie}_{j,g,k}) \right] \tag{13}$$

$$x_{i,k} \in \{0, 1\} \tag{14}$$

$$y_{i,j,t} \in \{0, 1\} \tag{15}$$

$$z_{i,j,g,k} \in \{0, 1\} \tag{16}$$

where Formula (1) indicates the objectives, i.e., makespan and total energy consumption; Constraint (2) indicates that the processing of one job cannot be interrupted; Constraint (3) ensures that no job can be processed before the prior job is completed; Constraint (4) indicates that every job can be assigned to only one factory; Constraint (5) ensures that the number of machines occupied at any time cannot be more than the total number of machines of this stage; Constraint (6) determines the time slots of every job in each stage; Constraint (7) determines the starting time of each job in every stage; Constraint (8) determines the completion time of each job in every stage; Constraint (9) determines the makespan of all factories; Constraint (10) determines the machines occupied by each job in each stage; Constraint (11) indicates that the completion time of one stage must be after all the jobs are completed in this stage; Constraint (12) indicates that the starting time of one stage must be before all the jobs are started in this stage; Eq. (13) calculates the energy consumption; and Formulas (14)–(16) indicate the type of decision variables.

3 NMOEA/D for EADHFSPMT

3.1 Description of basic MOEA/D

A Multi-objective Optimization Problem (MOP) can be described as follows:

$$F(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_p(\boldsymbol{x}))^{\mathrm{T}}, x \in \boldsymbol{\Omega} \qquad (17)$$

where p is the number of objectives and $\boldsymbol{\Omega}$ is the search space of solution $\boldsymbol{x}$.

For two solutions x_a and x_b of a MOP, if $f_l(x_a) \leqslant f_l(x_b)$, $\forall l$, and $f_{l'}(x_a) < f_{l'}(x_b)$, $\exists l' \in \{1, 2, \ldots, p\}$, x_a is said to dominate x_b (denoted as $x_a \succ x_b$). If no solution can dominate x_b, x_b is called a Pareto-optimal solution.

To solve MOPs, MOEA/D is an effective approach[20, 21, 28]. It decomposes an MOP into a number of single-objective sub-problems using decomposition function, and then optimizes these sub-problems simultaneously. Evolutionary operators like mutation and crossover are used to search promising solutions. When the stopping criterion is satisfied, the solutions of all the sub-problems make up a final set, which is an approximation for the optimal solution set of the original MOP. The description of the basic MOEA/D is referred to Ref. [27].

Recently, the MOEA/D has been applied to solve shop scheduling problems. Jiang and Wang[28] proposed an improved MOEA/D for the permutation flow shop scheduling problem with sequence dependent setup time to minimize makespan and energy consumption. Zhao et al.[29] proposed an improved MOEA/D for the job shop scheduling problem to minimize makespan, total tardiness and total flow time. Jiang and Wang[20] proposed a modified MOEA/D for the flexible job shop scheduling problem under time-of-use electricity prices to minimize the makespan and total electricity cost. Jiang et al.[21] proposed a collaborative MOEA/D for the distributed job shop to minimize makespan and total energy consumption. However, to the best of our knowledge, there is no existing work about the MOEA/D for the EADHFSPMT. Thus, it motivates us to propose an NMOEA/D to solve such a complex problem.

3.2 Procedure of NMOEA/D

The whole procedure of NMOEA/D is given in Algorithm 1.

For the above NMOEA/D, firstly N solutions are initialized by four heuristics, and the reference point is determined. The weight vectors are initialized uniformly and the neighbors of each solution are settled. Then, in every generation, the cooperative search, local intensification, and dynamic adjustment for weight vectors are performed in turn. In cooperative search, different operators are performed according to the different relationships between the solution and its neighbors. In local intensification, search operators inside factory and between factories are applied to optimize different objectives. In dynamic adjustment strategy, the weight vectors are adaptively adjusted according to the distribution of solutions. Finally, the neighbors of each solution are updated. The above search process is repeated generation by generation until a stopping criterion is satisfied.

To solve the EADHFSPMT reasonably, each component of the NMOEA/D should be designed specially.

3.3 Encoding and decoding

There are three sub-problems, i.e., job assignment

Algorithm 1 Procedure of NMOEA/D

```
Initialize N solutions and weight vectors and determine the reference point.
Determine the neighbors of each vector and the non-dominated set.
If the stopping criterion is not satisfied
    For Solution 1 to N
        Carry out cooperative search
        Carry out local intensification
        Update the reference point
    End For
    Update the neighbors of each solution
    Update the weight vectors and the non-dominated set
End If
Output the non-dominated set.
```

between factories, job sequence in each factory, and machine allocation for each job, which need to be considered. Therefore, we design an encoding scheme to represent a solution and a decoding scheme to obtain a feasible schedule from the representation.

For the encoding scheme, a solution is represented as f permutations. Each permutation corresponds to a factory and consists of the jobs assigned to the factory. Each job occurs only once and the sequence represents the order where jobs are processed in the first stage. For the simplicity of representation, the machine allocation is not considered in the permutation and will be determined in decoding scheme.

The allocation of machines for each job in every stage is a key factor to the completion time and energy consumption of jobs. Therefore, decoding scheme is very significant for the EADHFSPMT. To balance the makespan and energy consumption, we design an efficient and effective decoding scheme. The details of decoding scheme in each factory are described as follows.

Step 1: In the first stage, the processing order follows the sequence of this factory. In the later stages, the processing order depends on the completion time of each job in the previous stage and follows first-come-first-serve principle. That means, the earlier the job has been completed in the previous stage, the earlier it may be processed in the current stage.

Step 2: For job i in stage j, it needs $r_{i,j}$ machines. Choose $r_{i,j}$ machines in stage j with the earliest available time. The maximum one among these $r_{i,j}$ time determines the earliest processing time of job i in this stage. Let $\Psi = \{1, 2, \ldots, r_{i,j}\}$ be the set of $r_{i,j}$ machines with the earliest available time. Their available time is $\{t_1, t_2, \ldots, t_{r_{i,j}}\}$ and $t_1 \leqslant t_2 \leqslant \cdots \leqslant t_{r_{i,j}}$, which means the earliest processing time of job i is $t_{r_{i,j}}$.

Step 3: If there exists one machine in this stage out of Ψ with its available time as $t_{r_{i,j}}$, then replace the machine with the smallest available time in Ψ with this one. If there exist more than one machines out of Ψ with available time as $t_{r_{i,j}}$, then choose the one with the smallest energy consumption coefficient.

Step 4: Repeat Step 3 until no machine in Ψ can be replaced.

Step 5: If two jobs id and ih can start processing at the same time in stage j, then there exist two possible situations. (1) If T_j (id, ih) $= T_j$ (ih, id), then arrange the job with larger required number of machines in front. (2) If T_j (id, ih) $\neq T_j$ (ih, id), then arrange the job with smaller required number of machines in front.

Step 6: Repeat Steps 1–5 until all the jobs in this factory have been completed through all stages.

After all factories have been arranged by the above steps, one solution is decoded into a feasible schedule. Figure 1 shows an example of 2 factories (F1 and F2), 3 stages, and 7 jobs. There are 3 machines, 2 machines, and 3 machines in Stage 1, Stage 2, and Stage 3, respectively. The processing times and the required machines for each job in every stage are listed in Table 1.

To simplify calculation, set $e_{1,g,k} = \{4, 4, 8\}$ kW, $e_{2,g,k} = \{4, 8\}$ kW, and $e_{3,g,k} = \{4, 8, 8\}$ kW for each factory. From the encoded permutations, Jobs 3, 1, 5, and 7 are assigned to Factory 1, while Jobs 6, 2, and 4 are assigned to Factory 2. For Job 3 in Stage 1 in Factory 1, according to the decoding scheme, all machines can be available at time 0. Meanwhile, $e_{1,1,1} = e_{1,2,1} = 4 < e_{1,3,1} = 8$ and $r_{3,1} = 2$, so we choose Machines 1 and 2 to process Job 3 in Stage 1 and the energy consumption is 4×2×12=96 kWh. After arranging all the jobs in the first permutation, we obtain the feasible schedule for Factory 1. The schedule for Factory 2 can be obtained by the similar way. The whole schedule of this example is illustrated in Fig. 2. The completion time of Factory 1 is 64 h and the completion time of Factory 2 is 65 h. Thus, makespan of this solution is 65 h. The total energy consumption can be calculated through Eq. (13), which is equal to 2599 kWh for this example.

3.4 Initialization procedure

A well distributed initial population can save the computational resource and be helpful to obtain good solutions. As mentioned in Section 3.3, there are three sub-problems in the EADHFSPMT, where machine

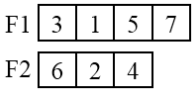

Fig. 1 An encoded solution.

Table 1 Processing time and the number of required machine.

Job	Stage 1		Stage 2		Stage 3	
	$r_{i,1}$	$p_{i,1}$ (h)	$r_{i,2}$	$p_{i,2}$ (h)	$r_{i,3}$	$p_{i,3}$ (h)
1	1	9	2	8	2	15
2	3	8	1	16	2	12
3	2	12	1	8	3	13
4	1	14	1	12	3	12
5	2	8	2	12	2	19
6	3	10	2	16	1	20
7	2	8	1	10	1	15

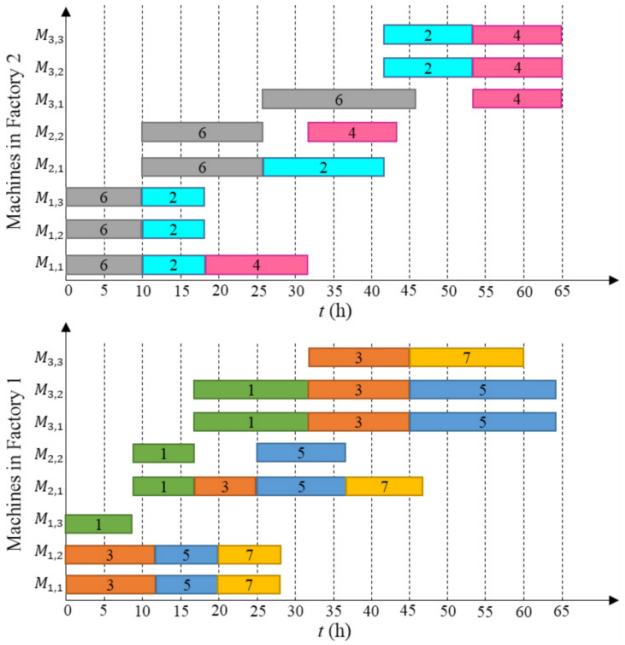

Fig. 2 Schedule of the example.

allocation can be solved by decoding scheme. Therefore, two elements are required to be initialized, i.e., job assignment between factories and job sequence in each factory. According to the above analysis, we present two heuristics for each of the two sub-problems.

For the job assignment between factories, two heuristics are designed as follows.

Smallest Processing Time (SPT) heuristic. Firstly, sort the jobs by total processing time in a descending order and assign the job to the factory with the smallest processing time from the first to the last. Then, select the factory with the least number of jobs if there exist more than one factories with the same processing time.

Smallest Energy consumption Factor (SEF) heuristic. Calculate the sum of energy consumption factors of machines of each factory. Then assign the job to the factory according to the probabilities based on the sum of energy consumption factors.

For example, suppose there are f factories and se_k represents the sum of energy consumption factors of machines of factory k. Sort se_k in an ascending order. Suppose $\mathrm{se}_1 \leqslant \mathrm{se}_2 \leqslant \cdots \leqslant \mathrm{se}_f$ in this example. Set μ_k as the probability for each job assigned to factory k and $\mu_k = \mathrm{se}_{f-k+1} / \sum_{k=1}^{f} \mathrm{se}_k$. By this way, the job has the largest probability to be assigned to the factory with the smallest energy consumption factor.

For the above two heuristics, SPT aims at balancing the processing time between factories to optimize makespan, while SEF aims at balancing the energy consumption factors between factories to optimize TEC.

For the job sequence in each factory, two heuristics are designed below.

Most Machine number First (MMF) heuristic. Sort the jobs of this factory by total number of required machines at each stage in a descending order. The larger its machine number is, the former this job will be.

Largest Processing time First (LPF) heuristic. Sort the jobs of this factory by total processing time in a descending order. The larger its processing time is, the former the job will be.

For the above two heuristics, MMF aims at completing the jobs with more required machines first, while LPF aims at completing the jobs with larger processing time first.

Since there are different heuristics designed in different purpose, four combinations of the above heuristics can be used to improve the quality of the initial population. Such a procedure is illustrated in Algorithm 2.

The decomposition function is crucial in the framework of the MOEA/D. The Tchebycheff approach has been proved to be effective in dealing with shop scheduling problems[20, 21, 28]. Thus, we employ this approach as the decomposition function in the NMOEA/D. For the solution x_p corresponding to the p-th sub-problem, according to the Tchebycheff approach, the objective function of the p-th sub-problem is $g\left(x_p|\boldsymbol{\lambda}_p, z^*\right) = \max_{l=1,2}\left\{\lambda_p^l \times \left|f_l\left(x_p\right) - z_l^*\right|\right\}$, where $\boldsymbol{\lambda}_p = (\lambda_p^1, \lambda_p^2)^{\mathrm{T}}$ is the weight vector corresponding to this sub-problem. The weight vector is initialized uniformly in the range of [0, 1], and $\sum_{l=1}^{r} \lambda_a^l = 1$. In original MOEA/D, the weight vectors remain unchanged. However, in the NMOEA/D, the weight vectors will be adjusted according to the distribution of the current population. The details will be introduced in Section 3.8. The reference point $z^* = (z_1^*, z_2^*)$ is determined by $z_1^* = \min\{f_1(x_1), \ldots, f_1(x_N)\}$ and $z_2^* = \min\{f_2(x_1), \ldots, f_2(x_N)\}$. The reference point

Algorithm 2 Procedure of initializing population

```
For initial solution 1 to N
    rd = rand( ) % 4
    Switch rd
        Case 0: Perform SPT and MMF
        Case 1: Perform SPT and LPF
        Case 2: Perform SEF and MMF
        Case 3: Perform SEF and LPF
    End switch
End For
```

will be updated after each generation updated. The details to determine the neighbors of each sub-problem and those of its corresponding solutions are referred to Ref. [21].

3.5 Cooperative search

In the original MOEA/D, the optimal solution of one sub-problem is regarded to be similar to the optimal solutions of the neighbors of this sub-problem[27]. The information from neighbors may be important to optimize each sub-problem. Thus, we design a cooperative search to produce new solutions.

As shown in Fig. 3, there are several relationships between one solution x_p and its neighbors in the normalized objective space. If the neighbor is located in Region I, this means that it dominates x_p. If the neighbor is located in Regions II or III, this means that it cannot dominate x_p, but it is closer to the reference point than x_p. If the neighbor is located in Region IV, this means that it is farther to the reference point than x_p.

From the above analysis, it can be seen that neighbors with different locations have different helpful information. For the solution x_p, there exist three types of relationship: (1) There is at least one neighbor located in Region I; (2) There is no neighbor located in Region I, but there exists at least one neighbor located in Regions II or II; (3) All neighbors are located in Region IV. Therefore, to adequately use the information from neighbors, different search operators are performed according to the relationship between x_p and its neighbors.

For Relationship 1, it means that one or more neighbors can dominate x_p, which indicates that the neighbors are better than x_p for the p-th sub-problem. Thus, the search operator for Relationship 1 is presented as follows.

Step 1: Select one neighbor x_p^b which dominates x_p.

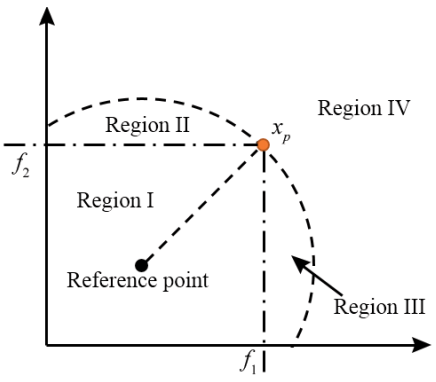

Fig. 3 Relationships between one solution and its neighbors.

If there exist more than one such neighbors, then choose the one closest to the reference point.

Step 2: Make x_p^{new} inherit all permutations from x_p^b. Implement local intensification (see Section 3.6) for x_p^{new}.

For Relationship 2, it means that one or more neighbors cannot dominate x_p, but are closer to the reference point than x_p. Thus, the search operator for Relationship 2 is presented as follows.

Step 1: Select one neighbor x_p^b which is located in Region II or II. If there exist more than one such neighbors, then choose the one closest to the reference point.

Step 2: If x_p^b is located in Region II, x_p^b has lower makespan but higher TEC than x_p. Thus, x_p^{new} inherits the permutation of the factory with minimal completion time from x_p^b. If x_p^b is located in Region III, x_p^b has lower TEC but higher makespan than x_p. Thus, x_p^{new} inherits the permutation of the factory with minimal energy consumption from x_p^b. Let F_b represent the factory x_p^{new} inherited from x_p^b.

Step 3: For other factories except F_b, x_p^{new} inherits the permutation from x_p and removes the jobs which have already been assigned to F_b.

Step 4: Assign the remaining jobs to factories except F_b by SEF or SPT randomly.

Step 5: Sequence the jobs of factories except F_b by MMF or LPF randomly.

Step 6: A new solution x_p^{new} is produced by Steps 1–5. If $x_p^{\text{new}} \succ x_p$ or $x_p \succ x_p^{\text{new}}$, then implement local intensification (see Section 3.6) for x_p^{new} or x_p. If x_p^{new} and x_p cannot dominate each other, then select the one closer to the reference point.

The above search operator generates a feasible new solution by using information from x_p and its neighbor x_p^b. An example of the above steps is illustrated in Fig. 4.

For Relationship 3, it means that x_p can dominate all its neighbors, which indicates that the neighbors have rare helpful information for x_p. Thus, the search operator for Relationship 3 is to make x_p^{new} inherit all permutations from x_p and implement local intensification (see Section 3.6) for x_p^{new}.

From the above description, a cooperative search is presented based on the relationship between each solution x_p and its neighbors. Different search operator are performed according to different types of relationship, which can effectively use the information from neighbors.

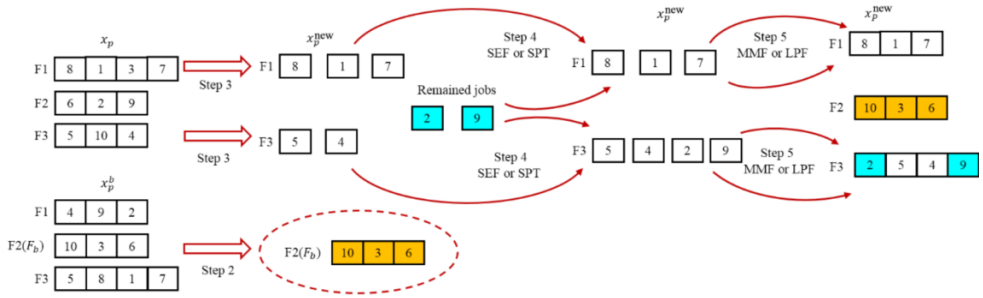

Fig. 4 An example of search operator for Relationship 2.

3.6 Local intensification

After performing cooperative search, we design a local intensification procedure to enhance the capability of exploitation. There are two objectives, i.e., makespan and TEC, which need to be optimized. Thus, two local search operators are designed aiming at optimizing these two objectives.

The first Local Search operator (LS1) for solution x_b is designed as follows.

Step 1: Calculate the energy consumption of all factories and select the factory with maximal energy consumption (denoted as F_e).

Step 2: Randomly choose one job in F_e and insert it in every possible position in this factory. Suppose there are n_F jobs in F_e. This step will generate n_F solutions (including x_b).

Step 3: Choose the solution which can dominate other $n_F - 1$ solutions as the result of LS1. If there is no such solution, then choose the solution closest to the reference point.

The second Local Search operator (LS2) is designed as follows.

Step 1: Calculate the completion time of all factories. Select the factory with maximal completion time (denoted as F_m) and the factory with minimal completion time (denoted as F_n).

Step 2: Calculate the gap between these two completion time (denoted as G_m).

Step 3: Calculate the total processing time of each job in F_m. If these exist one job whose total processing time is less than G_m, then insert it into every possible position in F_n. If there are more than one such jobs, then choose the one with minimal total processing time. If there is no such job, then keep unchanged.

Step 4: Suppose there are n_F jobs in F_n. If there exists such a job, then Step 3 will generate n_F solutions (including x_b). Choose the solution which can dominate other $n_F - 1$ solutions as the result of LS2. If there is no such solution, then choose the solution closest to the reference point.

From the above description, it can be seen that LS1 is performed inside one factory, while LS2 is performed between two factories. Moreover, LS1 aims at minimizing the energy consumption, while LS2 aims at minimizing the makespan. Different solutions have different preference in these two objectives. In the normalized objective space, there are two possibilities. (1) The TEC of this solution is larger than its makespan, which means TEC needs to be minimized more than makespan; (2) The makespan of this solution is larger than its TEC, which means makespan needs to be minimized more than TEC. Therefore, we design a collaborative way to carry out these two search operators to improve the quality of solutions. The details of the local intensification is illustrated in Algorithm 3.

3.7 Updating procedure

The reference point needs to be updated after performing cooperative search and local intensification on each solution. The neighbors of each solution also need to be updated according to the aggregation function of each sub-problem after the current generation updated.

Same as initialization procedure, the reference point $z^* = (z_1^*, z_2^*)$ is updated by the minimal value of objectives of the current population, i.e., $z_1^* =$

Algorithm 3 Local intensification for each x_b

If x_b satisfies the first situation
  Perform LS1(x_b) to produce a new solution x_b^{new}
  Perform LS2(x_b^{new}) to produce a new solution y_b^{new}
  Update $x_b^{new} = y_b^{new}$ if y_b^{new} dominates x_b^{new}
Else
  Perform LS2(x_b) to produce a new solution x_b^{new}
  Perform LS1(x_b^{new}) to produce a new solution y_b^{new}
  Update $x_b^{new} = y_b^{new}$ if y_b^{new} dominates x_b^{new}
End If
Output x_b^{new}

$\min\{f_1(x_1), \ldots, f_1(x_N)\}$ and $z_2^* = \min\{f_2(x_1), \ldots, f_2(x_N)\}$.

To update the neighbors, for each neighbor x_n of solution x_d, calculate $g(x_n|\lambda_n, z^*)$ and $g(x_d|\lambda_n, z^*)$. If $g(x_d|\lambda_n, z^*) \leqslant g(x_n|\lambda_n, z^*)$, replace the neighbor x_n with solution x_d. Otherwise, keep x_n unchanged.

3.8 Dynamic adjustment strategy for weight vectors

Each weight vector in the MOEA/D represents the search direction of each sub-problems. It has been proved that the distribution of the weight vectors can largely influence the distribution of final Pareto solutions[30]. In the original MOEA/D, the weight vectors are initialized uniformly in range of (0, 1) and keep fixed during the following evolutionary search. However, it is very likely that the true Pareto solutions are not uniformly distributed for a complex problem. Moreover, during the evolutionary search, the fixed weight vectors may not be fit to the distribution of the current generation. Thus, we design a dynamic adjustment for weight vectors to overcome the weakness.

According to the above analysis, the adjustment of weight vectors needs to guide the search towards the true distribution of Pareto set to guarantee the diversity and convergence. It needs to decrease search efforts when the current population is overcrowded in some region and increase search efforts when the current population is sparse in some region. Therefore, we reduce weight vectors in the region where the solution density is large and add weight vectors in the region where the solution density is small. An example of such a dynamic adjustment strategy is illustrated in Fig. 5.

The density of solutions in Region r is calculated as $\sigma_r = N_r \times P_r/\alpha_r$, where N_r denotes the number of weight vectors in Region r and P_r denotes the number of solutions in Region r. In addition, α_r is the angle between the axis and the reference point for Region r. Figure 5a shows the normalized objective space is divided into two regions based on the reference point. Figure 5b shows that the MOP is decomposed into eight sub-problems and each corresponds to a solution. At first, the weight vectors are distributed evenly. There are five solutions and four weight vectors in Region I, while three solutions and four weight vectors in Region II. The density of solutions is calculated as $\sigma_1 = 20/\alpha_1$ and $\sigma_2 = 12/\alpha_2$. Suppose $\alpha_1 < \alpha_2$, then $\sigma_1 > \sigma_2$. Thus, we reduce one weight vector in Region I and re-generate uniformly distributed weight vectors in Region I. Then, we add one weight vector in Region II and re-generate uniformly distributed weight vectors in Region II. The details are shown in Fig. 5c. In this way, the weight vectors are evenly distributed inside each region, but unevenly distributed in the whole normalized objective space. Moreover, updating the reference point in each generation will dynamically adjust the division of the normalized objective space. Therefore, the adjustment of weight vectors can be also dynamically achieved in each generation according to the distribution of current population.

4 Computational Experiment

4.1 Experimental settings

A set of DHFSPMT instances have been provided by Ying and Lin[1]. For this set, the scale of each instance is determined by three parameters, i.e., n, s, and f, where $n \in \{10, 20, 50, 100\}$, $s \in \{2, 5, 8\}$, and $f \in \{2, 3, 4, 5, 6\}$. Each combination of n, s, and f has 10 instances and there are 600 instances in total. However, Ying and Lin[1] only investigated single objective, i.e., makespan, without considering energy consumption. To calculate the energy consumption, we generate the $e_{j,g,k}$ uniformly in range of (2, 4) according to Zhang and

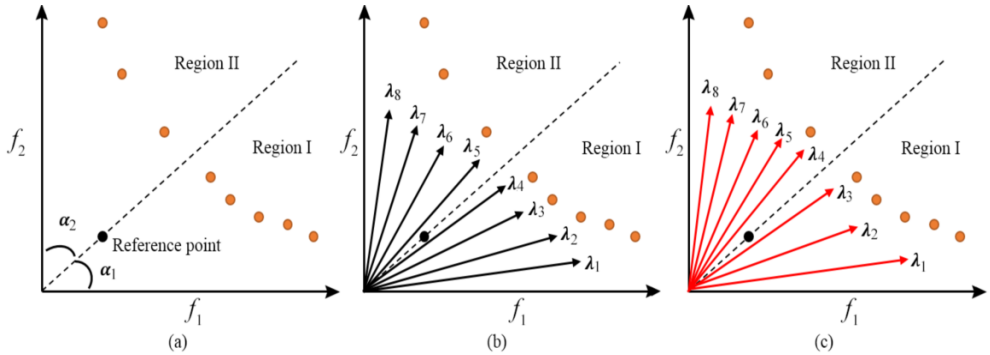

Fig. 5 Illustration of dynamic adjustment for weight vectors. (a) Division of the normalized objective space, (b) distribution of weight vectors, and (c) adjustment of weight vectors.

Chiong[31]. The stopping criterion is set as $0.05 \times n \times s$ (second) for all the experiments. The used computer is with Intel Core i5 CPU/3.20 GHz and 16 GB RAM. All the algorithms are realized in C++ and compiled by Visual Studio 2012.

For MOPs, two metrics are widely used to evaluate the performances of different algorithms, i.e., C-metric and T-metric. The C-metric is to measure the dominance between the non-dominated sets obtained by two algorithms, and the T-metric is to measure the diversity of the non-dominated set. The calculation of these metrics is referred to Ref. [21]. For the C-metric, $C(A, B) > C(B, A)$ means the non-dominated solutions obtained by algorithm A can dominate more solutions obtained by algorithm B. For the T-metric, $T(A) > T(B)$ means the non-dominated solutions obtained by algorithm B distribute more evenly than the solutions obtained by algorithm A.

For the NMOEA/D, two parameters should be set suitably, i.e., the population size (N) and the neighborhood size (T_n). We use the Taguchi method with an instance of the combination ($n = 50, s = 5$, and $f = 3$) to investigate the effect of parameter setting. Consider four levels for each parameter, i.e., $N \in \{50, 100, 200, 300\}$ and $T_n \in \{3, 5, 7, 9\}$. For the 4^2 full-factorial experiments, 16 combinations of levels are investigated. The NMOEA/D is run 10 times independently for each of 16 combinations to aggregate a set of 10 obtained non-dominated sets. Then, all the 16 aggregated sets $E_1 - E_{16}$ are combined to obtain a final non-dominated set ES. The percentage (denoted as SP) of the solutions from each of the 16 aggregated sets $E_1 - E_{16}$ in ES represents the contribution of each parameter combination to the final set. From the description above, it can be concluded that the combination can yield better performance if its SP value is larger.

Table 2 lists the SP value of each parameter combination and the average SP value for 4 levels of each parameter. Figure 6 shows the trends of two parameters. In Table 3, GP denotes the largest gab between two levels of each parameter and represents different significance of these parameters. The larger GP is, the more significant this parameter is. From Table 2 it can be seen that the neighborhood size T_n is slightly more significant than the population N. In the MOEA/D, a large population size N may yield more Pareto solutions, but may cause larger computational complexity. On the contrary, a small N may lead to rapid convergence, but may result in less Pareto solutions. For the neighborhood size T_n, it balances the diversity and convergence of the population at cooperative search and updating procedures. From Table 2 and Fig. 6, it can be seen that the algorithm achieves better performance when setting population size as level 3 and the neighborhood size as level 2. Therefore, we set N as 200 and T_n as 5 for the further computational experiments.

Table 2 Investigation of parameter effect.

Experiment number	Parameter level		SP
	N	T_n	
1	1	1	0.01
2	1	2	0.05
3	1	3	0.06
4	1	4	0.02
5	2	1	0.03
6	2	2	0.08
7	2	3	0.10
8	2	4	0.04
9	3	1	0.07
10	3	2	0.15
11	3	3	0.09
12	3	4	0.05
13	4	1	0.05
14	4	2	0.09
15	4	3	0.08
16	4	4	0.03

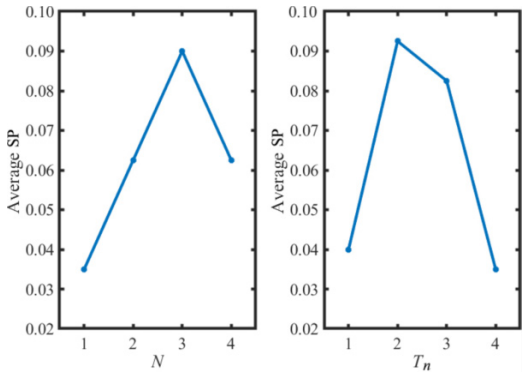

Fig. 6 Trend of two parameters.

Table 3 Average SP value of each parameter.

Parameter level	Average SP	
	N	T_n
1	0.0350	0.0400
2	0.0625	**0.0925**
3	**0.0900**	0.0825
4	0.0625	0.0350
GP	0.0550	0.0575

4.2 Effectiveness of local intensification

In the NMOEA/D, two local search operators and a

collaborative mechanism for these two operators have been designed in the local intensification procedure. To demonstrate the effectiveness of the local intensification, the NMOEA/D is compared to the NMOEA/D with random local search (denoted as NMOEA/D-NL), which inserts a randomly selected job into a random position. Each instance of each (n, s, f) combination is run 20 times independently and the average value of C-metric of 10 instances of each (n, s, f) combination is listed in Table 4. The p values of pairwise comparison with 95% confidence level are provided. The boxplot of the average C-metric is shown in Fig. 7.

From Table 4 and Fig. 7, it can be seen that the average value of C(NMOEA/D, NMOEA/D-NL) is larger than that of C(NMOEA/D-NL, NMOEA/D) on all (n, s, f) combinations. This means the final non-dominated set obtained by the NMOEA/D can dominate more solutions of the NMOEA/D-NL. All the p values are equal to 0, which implies that the NMOEA/D performs significantly better than the NMOEA/D-NL. Therefore, the specially designed local intensification is proved to be effective to improve the performance of the NMOEA/D. Figure 8 illustrates the Pareto fronts obtained by the two algorithms in solving one instance with the combination $n = 20, s = 8$, and $f = 2$.

4.3 Effectiveness of dynamic adjustment strategy

The dynamic adjustment strategy for weight vectors allocates the number and distribution of weight vectors in each generation, which guides the search direction towards the true Pareto fronts. To demonstrate the effectiveness of this strategy, the NMOEA/D is compared to the NMOEA/D without adjustment strategy (denoted as NMOEA/D-NA). Same as Section 4.2, each

Table 4 Average value of C-metric between NMOEA/D and NMOEA/D-NL.

Parameter combination (f, n, s)	NMOEA/D vs. NMOEA/D-NL			Parameter combination (f, n, s)	NMOEA/D vs. NMOEA/D-NL		
	C(NMOEA/D, NMOEA/D-NL)	C(NMOEA/D-NL, NMOEA/D)	p		C(NMOEA/D, NMOEA/D-NL)	C(NMOEA/D-NL, NMOEA/D)	p
(2, 10, 2)	{**0.94**}	0.01	0	(4, 50, 2)	{**0.87**}	0.10	0
(2, 10, 5)	{**0.83**}	0.14	0	(4, 50, 5)	{**0.82**}	0.15	0
(2, 10, 8)	{**0.98**}	0	0	(4, 50, 8)	{**0.79**}	0.19	0
(2, 20, 2)	{**0.87**}	0.10	0	(4, 100, 2)	{**0.79**}	0.18	0
(2, 20, 5)	{**0.87**}	0.10	0	(4, 100, 5)	{**0.78**}	0.11	0
(2, 20, 8)	{**0.75**}	0.13	0	(4, 100, 8)	{**0.76**}	0.14	0
(2, 50, 2)	{**0.95**}	0.02	0	(5, 10, 2)	{**0.61**}	0.23	0
(2, 50, 5)	{**0.82**}	0.16	0	(5, 10, 5)	{**0.61**}	0.19	0
(2, 50, 8)	{**0.95**}	0	0	(5, 10, 8)	{**0.68**}	0.30	0
(2, 100, 2)	{**0.97**}	0	0	(5, 20, 2)	{**0.96**}	0.03	0
(2, 100, 5)	{**0.82**}	0.15	0	(5, 20, 5)	{**0.84**}	0.13	0
(2, 100, 8)	{**0.93**}	0	0	(5, 20, 8)	{**0.95**}	0.03	0
(3, 10, 2)	{**0.84**}	0.15	0	(5, 50, 2)	{**0.97**}	0	0
(3, 10, 5)	{**0.77**}	0.20	0	(5, 50, 5)	{**0.83**}	0.15	0
(3, 10, 8)	{**0.85**}	0.12	0	(5, 50, 8)	{**0.99**}	0	0
(3, 20, 2)	{**0.77**}	0.17	0	(5, 100, 2)	{**0.77**}	0.18	0
(3, 20, 5)	{**0.92**}	0.07	0	(5, 100, 5)	{**0.76**}	0.19	0
(3, 20, 8)	{**0.89**}	0.10	0	(5, 100, 8)	{**0.91**}	0.06	0
(3, 50, 2)	{**0.86**}	0.12	0	(6, 10, 2)	{**0.73**}	0.21	0
(3, 50, 5)	{**0.83**}	0.14	0	(6, 10, 5)	{**0.70**}	0.12	0
(3, 50, 8)	{**0.85**}	0.14	0	(6, 10, 8)	{**0.73**}	0.16	0
(3, 100, 2)	{**0.88**}	0.09	0	(6, 20, 2)	{**0.85**}	0.14	0
(3, 100, 5)	{**0.89**}	0.09	0	(6, 20, 5)	{**0.93**}	0.06	0
(3, 100, 8)	{**0.79**}	0.19	0	(6, 20, 8)	{**0.96**}	0.01	0
(4, 10, 2)	{**0.91**}	0	0	(6, 50, 2)	{**0.85**}	0.11	0
(4, 10, 5)	{**0.81**}	0.16	0	(6, 50, 5)	{**0.89**}	0.10	0
(4, 10, 8)	{**0.78**}	0.19	0	(6, 50, 8)	{**0.87**}	0.09	0
(4, 20, 2)	{**0.77**}	0.19	0	(6, 100, 2)	{**0.97**}	0.02	0
(4, 20, 5)	{**0.84**}	0.13	0	(6, 100, 5)	{**0.95**}	0	0
(4, 20, 8)	{**0.76**}	0.21	0	(6, 100, 8)	{**0.84**}	0.10	0

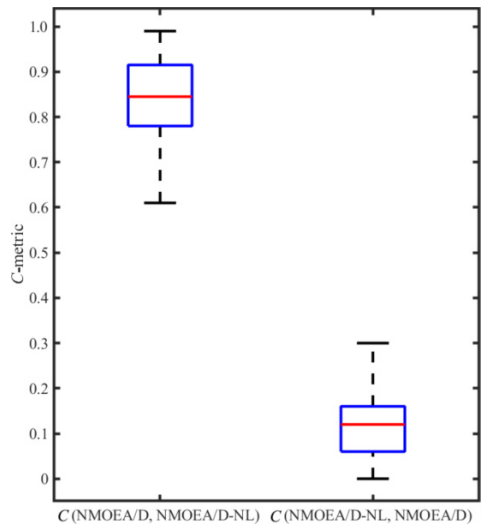

Fig. 7 Boxplot of the average *C*-metrics between NMOEA/D and NMOEA/D-NL.

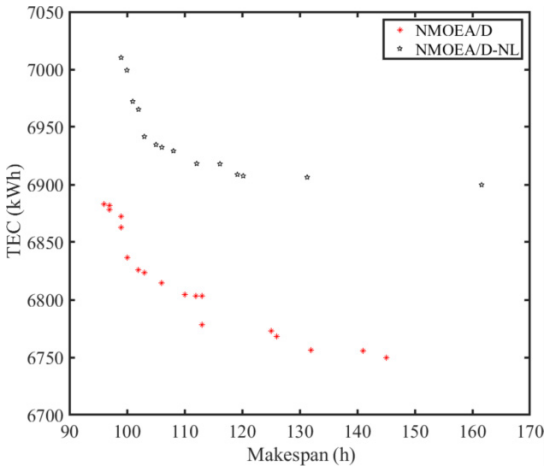

Fig. 8 Pareto fronts by NMOEA/D and NMOEA/D-NL on instance with n=20, s =8, and f =2.

instance of each (n, s, f) combination is run 20 times independently. Since the dynamic adjustment strategy is mainly designed to adjust the diversity and convergence of the final non-dominated solutions, we focus on the distribution of the results more than the relationship of dominance. So, the T-metric is used to compare these two algorithms. The average value of T-metric of the 10 instances of each (n, s, f) combination is listed in Table 5. The boxplot of the average value of T-metrics is shown in Fig. 9.

From Table 5 and Fig. 9, it can be seen that the average value of T-metric of the NMOEA/D is smaller than that of the NMOEA/D-NA on most combinations. This means the non-dominated solutions obtained by the NMOEA/D get better distribution than those obtained by the NMOEA/D-NA on most instances. Therefore, it can be concluded that the dynamic adjustment strategy for weigh vectors is effective to balance the diversity and convergence of the solutions.

4.4 Comparisons to existing algorithms

Next, we compare the NMOEA/D to other two algorithms. Since there is no existing work about the same problem, we adapt the latest DSFLA[15] for the DHFSPMT with makespan criterion and the widely used NSGA-II[32] to solve the EADHFSPMT for comparisons. For fair comparison, the encoding and decoding schemes of the NMOEA/D are employed for the comparative algorithms. The parameter setting of the DSFLA and NSGA-II follows the related Ref. [15] and Ref. [32], respectively.

Each instance of each (n, s, f) combination is run 20 times independently and the average value of C-metric of the 10 instances of each (n, s, f) combination is listed in Table 6, where the p values of pairwise comparison with 95% confidence level are also provided. The boxplot of the average value of C-metric is shown in Fig. 10. Moreover, the average value of T-metric of the 10 instances of each (n, s, f) combination is listed in Table 7. The boxplot of the average value of T-metric is shown in Fig. 11.

From Table 6 and Fig. 10, it can be seen that the average C(NMOEA/D, DSFLA) and C(NMOEA/D, NSGA-II) are larger than C(DSFLA, NMOEA/D) and C(NSGA-II, NMOEA/D). This means the non-dominated solutions obtained by the NMOEA/D can dominate more solutions of the DSFLA and NSGA-II on all instances. Meanwhile, the C(NMOEA/D, DSFLA) and C(NMOEA/D, NSGA-II) are much close to 1 while the C(DSFLA, NMOEA/D) and C(NSGA-II, NMOEA/D) is almost equal to 0. This means the solutions obtained by the NMOEA/D can dominate most solutions obtained by the DSFLA and NSGA-II, but the solutions obtained by the DSFLA and NSGA-II can rarely dominate solutions obtained by the NMOEA/D. Therefore, it can be concluded that the NMOEA/D performs better than the DSFLA and NSGA-II in terms of the dominance relationship.

From Table 7 and Fig. 11, it can be seen that the average value of T-metric of the NMOEA/D is smaller than that of the DSFLA and NSGA-II for all combinations. This means the non-dominated solutions obtained by the NMOEA/D get better distribution than those obtained by the DSFAL and NSGA-II on

Table 5 Average value of *T*-metric of NMOEA/D and NOEA/D-NA.

Parameter combination (f, n, s)	NMOEA/D vs. NMOEA/D-NA		Parameter combination (f, n, s)	NMOEA/D vs. NMOEA/D-NA	
	T(NMOEA/D)	T(NMOEA/D-NA)		T(NMOEA/D)	T(NMOEA/D-NA)
(2, 10, 2)	**0.71**	0.86	(4, 50, 2)	**0.72**	0.88
(2, 10, 5)	**0.66**	0.92	(4, 50, 5)	**0.80**	0.86
(2, 10, 8)	**0.82**	0.88	(4, 50, 8)	**0.69**	0.80
(2, 20, 2)	**0.67**	0.94	(4, 100, 2)	**0.79**	0.85
(2, 20, 5)	0.84	**0.82**	(4, 100, 5)	0.85	**0.80**
(2, 20, 8)	**0.71**	0.79	(4, 100, 8)	**0.80**	0.83
(2, 50, 2)	**0.80**	0.85	(5, 10, 2)	**0.78**	0.90
(2, 50, 5)	**0.73**	0.86	(5, 10, 5)	**0.66**	0.85
(2, 50, 8)	**0.66**	0.80	(5, 10, 8)	**0.83**	0.87
(2, 100, 2)	**0.72**	0.97	(5, 20, 2)	0.84	**0.81**
(2, 100, 5)	**0.70**	0.84	(5, 20, 5)	0.85	**0.82**
(2, 100, 8)	**0.80**	0.87	(5, 20, 8)	**0.77**	0.89
(3, 10, 2)	**0.77**	0.84	(5, 50, 2)	**0.74**	0.87
(3, 10, 5)	**0.67**	0.87	(5, 50, 5)	**0.69**	0.93
(3, 10, 8)	**0.79**	0.96	(5, 50, 8)	**0.82**	0.84
(3, 20, 2)	**0.69**	0.88	(5, 100, 2)	**0.79**	0.85
(3, 20, 5)	**0.83**	0.96	(5, 100, 5)	**0.66**	0.87
(3, 20, 8)	**0.70**	0.85	(5, 100, 8)	**0.68**	0.96
(3, 50, 2)	**0.76**	0.81	(6, 10, 2)	**0.72**	0.84
(3, 50, 5)	**0.79**	0.88	(6, 10, 5)	**0.69**	0.80
(3, 50, 8)	**0.68**	0.79	(6, 10, 8)	0.85	**0.81**
(3, 100, 2)	**0.82**	0.90	(6, 20, 2)	**0.82**	0.83
(3, 100, 5)	**0.67**	0.81	(6, 20, 5)	**0.73**	0.81
(3, 100, 8)	**0.73**	0.85	(6, 20, 8)	**0.83**	0.85
(4, 10, 2)	**0.66**	0.96	(6, 50, 2)	**0.77**	0.92
(4, 10, 5)	0.83	**0.79**	(6, 50, 5)	**0.82**	0.93
(4, 10, 8)	**0.75**	0.78	(6, 50, 8)	**0.85**	0.90
(4, 20, 2)	**0.76**	0.88	(6, 100, 2)	**0.84**	0.97
(4, 20, 5)	**0.73**	0.81	(6, 100, 5)	**0.67**	0.93
(4, 20, 8)	**0.80**	0.91	(6, 100, 8)	**0.77**	0.96

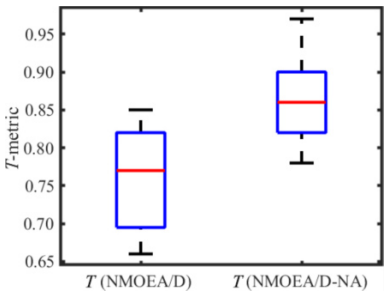

Fig. 9 Boxplot of the average *T*-metrics between NMOEA/D and NMOEA/D-NA.

all instances. Therefore, it can be concluded that the NMOEA/D has better performance than the DSFLA and NSGA-II in terms of the diversity and convergence.

Thus, it is demonstrated that the NMOEA/D significantly performs better than the existing DSFLA and NSGA-II in solving the EADHFSPMT. Figure 12 illustrated the Pareto fronts obtained by the three algorithms in solving some different instances, which clearly show the superiority of the NMOEA/D.

5 Conclusion and Future Work

This is the first research work using decomposition-based multi-objective evolutionary algorithm to solve the EADHFSPMT. We specially design decoding scheme, search operators, and adjusting strategy to improve the performance of the MOEA/D in solving the complex scheduling problem. The comparative results demonstrate the effectiveness of local intensification and adjusting strategy. Statistical comparisons also show the superior performance of the proposed algorithm to the existing optimization algorithms. This work provides an effective optimization technique for solving the EADHFSPMT, and it also provides some idea in

Table 6 Average value of C-metric between NMOEA/D, DSFLA, and NSGA-II.

Parameter combination (f, n, s)	NMOEA/D vs. DSFLA			NMOEA/D vs. NSGA-II		
	C(NMOEA/D, DSFLA)	C(DSFLA, NMOEA/D)	p	C(NMOEA/D, NSGA-II)	C(NSGA-II, NMOEA/D)	p
(2, 10, 2)	**0.78**	0.11	0	**0.93**	0.03	0
(2, 10, 5)	**0.83**	0.02	0	**0.94**	0	0
(2, 10, 8)	**0.79**	0.09	0	**0.94**	0.01	0
(2, 20, 2)	**0.76**	0.10	0	**0.87**	0.02	0
(2, 20, 5)	**0.84**	0.02	0	**0.92**	0.04	0
(2, 20, 8)	**0.75**	0.11	0	**0.91**	0.03	0
(2, 50, 2)	**0.81**	0.08	0	**0.94**	0.02	0
(2, 50, 5)	**0.78**	0.10	0	**0.91**	0.06	0
(2, 50, 8)	**0.73**	0.16	0	**0.90**	0.06	0
(2, 100, 2)	**0.88**	0.01	0	**0.89**	0.04	0
(2, 100, 5)	**0.74**	0.11	0	**0.92**	0.05	0
(2, 100, 8)	**0.90**	0.04	0	**0.92**	0.01	0
(3, 10, 2)	**0.83**	0.04	0	**0.80**	0.11	0
(3, 10, 5)	**0.88**	0.05	0	**0.82**	0.07	0
(3, 10, 8)	**0.83**	0.02	0	**0.90**	0.03	0
(3, 20, 2)	**0.74**	0.12	0	**0.89**	0.05	0
(3, 20, 5)	**0.75**	0.14	0	**0.85**	0.08	0
(3, 20, 8)	**0.77**	0.10	0	**0.93**	0.01	0
(3, 50, 2)	**0.88**	0.06	0	**0.87**	0.06	0
(3, 50, 5)	**0.90**	0.01	0	**0.91**	0.06	0
(3, 50, 8)	**0.78**	0.07	0	**0.91**	0.06	0
(3, 100, 2)	**0.76**	0.14	0	**0.87**	0.07	0
(3, 100, 5)	**0.83**	0.06	0	**0.80**	0.12	0
(3, 100, 8)	**0.87**	0.01	0	**0.82**	0.14	0
(4, 10, 2)	**0.81**	0.06	0	**0.85**	0.10	0
(4, 10, 5)	**0.80**	0.09	0	**0.81**	0.14	0
(4, 10, 8)	**0.71**	0.18	0	**0.82**	0.08	0
(4, 20, 2)	**0.82**	0.05	0	**0.89**	0.04	0
(4, 20, 5)	**0.72**	0.18	0	**0.86**	0.08	0
(4, 20, 8)	**0.70**	0.16	0	**0.92**	0.04	0
(4, 50, 2)	**0.89**	0.02	0	**0.83**	0.12	0
(4, 50, 5)	**0.76**	0.14	0	**0.88**	0.06	0
(4, 50, 8)	**0.77**	0.10	0	**0.88**	0.09	0
(4, 100, 2)	**0.76**	0.14	0	**0.82**	0.13	0
(4, 100, 5)	**0.71**	0.14	0	**0.88**	0.09	0
(4, 100, 8)	**0.75**	0.10	0	**0.94**	0	0
(5, 10, 2)	**0.77**	0.08	0	**0.93**	0.03	0
(5, 10, 5)	**0.75**	0.14	0	**0.88**	0.08	0
(5, 10, 8)	**0.73**	0.12	0	**0.92**	0.05	0
(5, 20, 2)	**0.77**	0.13	0	**0.95**	0	0
(5, 20, 5)	**0.89**	0.01	0	**0.92**	0.02	0
(5, 20, 8)	**0.72**	0.18	0	**0.88**	0.09	0
(5, 50, 2)	**0.73**	0.16	0	**0.94**	0	0
(5, 50, 5)	**0.86**	0.04	0	**0.91**	0.02	0
(5, 50, 8)	**0.88**	0.01	0	**0.88**	0.06	0
(5, 100, 2)	**0.85**	0.05	0	**0.88**	0.09	0
(5, 100, 5)	**0.78**	0.11	0	**0.93**	0	0
(5, 100, 8)	**0.84**	0.02	0	**0.84**	0.12	0

(To be continued)

Table 6 Average value of C-metric between NMOEA/D, DSFLA, and NSGA-II.

(Continued)

Parameter combination (f, n, s)	NMOEA/D vs. DSFLA			NMOEA/D vs. NSGA-II		
	C(NMOEA/D, DSFLA)	C(DSFLA, NMOEA/D)	p	C(NMOEA/D, NSGA-II)	C(NSGA-II, NMOEA/D)	p
(6, 10, 2)	**0.87**	0.03	0	**0.95**	0.01	0
(6, 10, 5)	**0.82**	0.07	0	**0.91**	0.02	0
(6, 10, 8)	**0.89**	0.04	0	**0.85**	0.11	0
(6, 20, 2)	**0.72**	0.18	0	**0.93**	0.02	0
(6, 20, 5)	**0.70**	0.18	0	**0.87**	0.09	0
(6, 20, 8)	**0.89**	0.01	0	**0.80**	0.07	0
(6, 50, 2)	**0.82**	0.07	0	**0.93**	0.04	0
(6, 50, 5)	**0.75**	0.12	0	**0.95**	0.02	0
(6, 50, 8)	**0.74**	0.13	0	**0.88**	0.06	0
(6, 100, 2)	**0.84**	0.03	0	**0.82**	0.11	0
(6, 100, 5)	**0.76**	0.11	0	**0.82**	0.11	0
(6, 100, 8)	**0.72**	0.16	0	**0.88**	0.06	0

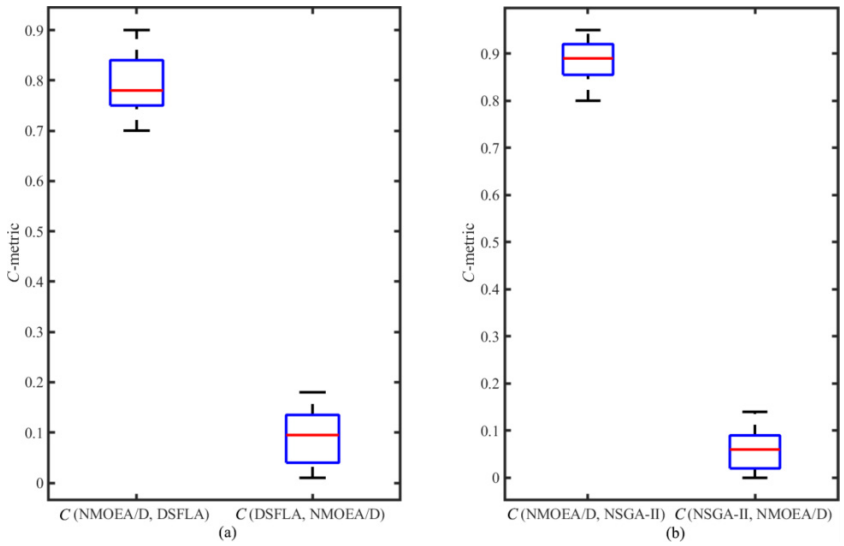

Fig. 10 Boxplots of the average C-metrics between NMOEA/D, DSFLA, and NSGA-II.

improving the MOEA/D for complex problems.

In future research, in problem aspect we will extend the MOEA/D for solving the scheduling problems with other objectives and other manufacturing environments. In algorithm aspect we will further explore the efficient decomposition mechanism and knowledge-guided search operators for the MOEA/D. It is also interesting to study hybrid algorithms by fusing the evolutionary algorithm with artificial intelligence approaches like reinforcement learning.

Acknowledgment

This research was supported by the National Natural Science Fund for Distinguished Young Scholars of China (No. 61525304) and the National Natural Science Foundation of China (No. 61873328).

References

[1] K. C. Ying and S. W. Lin, Minimizing makespan for the distributed hybrid flowshop scheduling problem with multiprocessor tasks, *Expert Syst. Appl.*, vol. 92, pp. 132–141, 2018.

[2] L. Hidri and A. Gharbi, New efficient lower bound for the hybrid flow shop scheduling problem with multiprocessor tasks, *IEEE Access*, vol. 5, pp. 6121–6133, 2017.

[3] L. Hidri, Note on the hybrid flowshop scheduling problem with multiprocessor tasks, *Int. J. Prod. Econ.*, vol. 182, pp. 531–534, 2016.

[4] M. Kurdi, Ant colony system with a novel Non-DaemonActions procedure for multiprocessor task scheduling in multistage hybrid flow shop, *Swarm Evol. Comput.*, vol. 44, pp. 987–1002, 2019.

[5] H. R. Gholami, E. Mehdizadeh, and B. Naderi, Mathematical models and an elephant herding optimization for multiprocessor-task flexible flow shop scheduling

Table 7 Average value of T-metric of NMOEA/D, DSFLA, and NSGA-II.

Parameter combination (f, n, s)	T-metric			Parameter combination (f, n, s)	T-metric		
	T(NMOEA/D)	T(DSFLA)	T(NSGA-II)		T(NMOEA/D)	T(DSFLA)	T(NSGA-II)
(2, 10, 2)	**0.71**	0.92	0.82	(4, 50, 2)	**0.72**	0.98	0.87
(2, 10, 5)	**0.66**	0.95	0.82	(4, 50, 5)	**0.80**	0.91	0.94
(2, 10, 8)	**0.82**	1.03	0.83	(4, 50, 8)	**0.69**	1.01	0.87
(2, 20, 2)	**0.67**	1.05	0.89	(4, 100, 2)	**0.79**	0.99	0.8
(2, 20, 5)	**0.84**	0.91	0.91	(4, 100, 5)	**0.85**	1.00	0.96
(2, 20, 8)	**0.71**	0.95	0.78	(4, 100, 8)	**0.80**	1.03	0.82
(2, 50, 2)	**0.80**	0.98	0.93	(5, 10, 2)	**0.78**	0.90	0.88
(2, 50, 5)	**0.73**	0.94	0.78	(5, 10, 5)	**0.66**	0.92	0.9
(2, 50, 8)	**0.66**	0.92	0.79	(5, 10, 8)	**0.83**	0.94	0.76
(2, 100, 2)	**0.72**	0.97	0.92	(5, 20, 2)	**0.84**	0.91	0.94
(2, 100, 5)	**0.70**	1.00	0.85	(5, 20, 5)	**0.85**	0.99	0.97
(2, 100, 8)	**0.80**	1.02	0.86	(5, 20, 8)	**0.77**	0.97	0.76
(3, 10, 2)	**0.77**	1.01	0.82	(5, 50, 2)	**0.74**	0.91	0.94
(3, 10, 5)	**0.67**	1.03	0.89	(5, 50, 5)	**0.69**	1.02	0.77
(3, 10, 8)	**0.79**	0.99	0.86	(5, 50, 8)	**0.82**	0.99	0.85
(3, 20, 2)	**0.69**	0.90	0.78	(5, 100, 2)	**0.79**	1.00	0.82
(3, 20, 5)	**0.83**	1.05	0.88	(5, 100, 5)	**0.66**	1.02	0.84
(3, 20, 8)	**0.70**	0.90	0.93	(5, 100, 8)	**0.68**	0.92	0.93
(3, 50, 2)	**0.76**	0.90	0.81	(6, 10, 2)	**0.72**	0.90	0.88
(3, 50, 5)	**0.79**	0.93	0.84	(6, 10, 5)	**0.69**	1.04	0.81
(3, 50, 8)	**0.68**	0.90	0.94	(6, 10, 8)	**0.85**	0.97	0.87
(3, 100, 2)	**0.82**	0.90	0.95	(6, 20, 2)	**0.82**	0.95	0.86
(3, 100, 5)	**0.67**	0.93	0.77	(6, 20, 5)	**0.73**	1.00	0.85
(3, 100, 8)	**0.73**	0.92	0.78	(6, 20, 8)	**0.83**	1.03	0.95
(4, 10, 2)	**0.66**	0.92	0.86	(6, 50, 2)	**0.77**	0.99	0.78
(4, 10, 5)	**0.83**	0.99	0.93	(6, 50, 5)	**0.82**	0.95	0.83
(4, 10, 8)	**0.75**	0.94	0.83	(6, 50, 8)	**0.85**	0.99	0.96
(4, 20, 2)	**0.76**	0.99	0.95	(6, 100, 2)	**0.84**	1.02	0.98
(4, 20, 5)	**0.73**	1.00	0.94	(6, 100, 5)	**0.67**	0.96	0.95
(4, 20, 8)	**0.80**	0.93	0.87	(6, 100, 8)	**0.77**	1.02	0.93

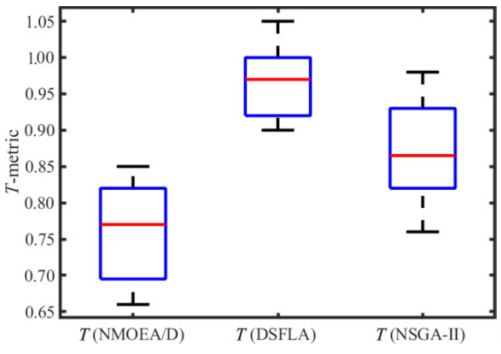

Fig. 11 Boxplots of the average T-metrics NMOEA/D, DSFLA, and NSGA-II.

problems in the manufacturing resource planning (MRPII) system, *Sci. Iran.*, vol. 27, no. 3, pp. 1562–1571, 2020.

[6] H. Gholami and M. T. Rezvan, A memetic algorithm for multistage hybrid flow shop scheduling problem with multiprocessor tasks to minimize makespan, *Int. J. Ind. Eng. Manage. Sci.*, vol. 7, no. 1, pp. 181–200, 2020.

[7] R. Ruiz, Q. K. Pan, and B. Naderi, Iterated greedy methods for the distributed permutation flowshop scheduling problem, *Omega*, vol. 83, pp. 213–222, 2019.

[8] Q. K. Pan, L. Gao, L. Wang, J. Liang, and X. Y. Li, Effective heuristics and metaheuristics to minimize total flowtime for the distributed permutation flowshop problem, *Expert Syst. Appl.*, vol. 124, pp. 309–324, 2019.

[9] Z. Shao, W. Shao, and D. Pi, Effective heuristics and metaheuristics for the distributed fuzzy blocking flow-shop scheduling problem, *Swarm Evol. Comput.*, vol. 59, p. 100747, 2020.

[10] J. J. Wang and L. Wang, A bi-population cooperative memetic algorithm for distributed hybrid flow-shop scheduling, *IEEE Trans. Emerg. Top. Comput. Intell.*, doi: 10.1109/TETCI.2020.3022372.

[11] I. Chaouch, O. B. Driss, and K. Ghedira, A modified ant colony optimization algorithm for the distributed job shop scheduling problem, *Procedia Comput. Sci.*, vol. 112, pp. 296–305, 2017.

[12] I. Chaouch, O. B. Driss, and K. Ghedira, A novel dynamic assignment rule for the distributed job shop scheduling

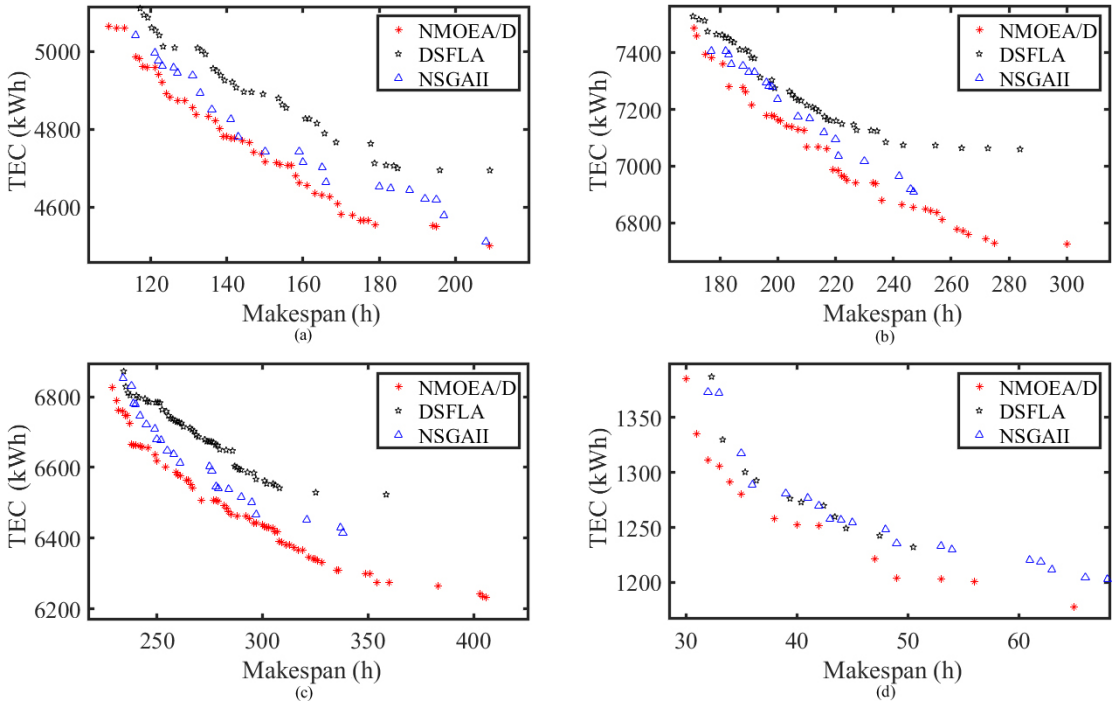

Fig. 12 Pareto fronts in solving different instances, (a) n=50, s=2, and f=2; (b) n=50, s=5, and f=2; (c) n=100, s=2, and f=2; and (d) n=20, s=2, and f=4.

problem using a hybrid ant-based algorithm, *Appl. Intell.*, vol. 49, no. 5, pp. 1903–1924, 2019.

[13] C. Y. Hsu, B. R. Kao, and K. R. Lai, Agent-based fuzzy constraint-directed negotiation mechanism for distributed job shop scheduling, *Eng. Appl. Artif. Intel.*, vol. 53, pp. 140–154, 2016.

[14] L. Meng, C. Zhang, Y. Ren, B. Zhang, and C. Lv, Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem, *Comput. Ind. Eng.*, vol. 142, p. 106347, 2020.

[15] J. Cai, R. Zhou, and D. Lei, Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks, *Eng. Appl. Artif. Intel.*, vol. 90, p. 103540, 2020.

[16] X. Wu and A. Che, Energy-efficient no-wait permutation flow shop scheduling by adaptive multi-objective variable neighborhood search, *Omega*, vol. 94, p. 102117, 2020.

[17] Z. Liu, J. Yan, Q. Cheng, C. Yang, S. Sun, and D. Xue, The mixed production mode considering continuous and intermittent processing for an energy-efficient hybrid flow shop scheduling, *J. Clean. Prod.*, vol. 246, p. 119071, 2020.

[18] J. J. Wang and L. Wang, A knowledge-based cooperative algorithm for energy-efficient scheduling of distributed flow-shop, *IEEE Trans. Syst., Man, Cybern.: Syst.*, vol. 50, no. 5, pp. 1805–1819, 2020.

[19] Y. Li, W. Huang, R. Wu, and K. Guo, An improved artificial bee colony algorithm for solving multi-objective low-carbon flexible job shop scheduling problem, *Appl. Soft Comput.*, vol. 95, p. 106544, 2020.

[20] E. D. Jiang and L. Wang, Multi-objective optimization based on decomposition for flexible job shop scheduling under time-of-use electricity prices, *Knowl.-Based Syst.*, vol. 204, p. 106177, 2020.

[21] E. D. Jiang, L. Wang, and Z. Peng, Solving energy-efficient distributed job shop scheduling via multi-objective evolutionary algorithm with decomposition, *Swarm Evol. Comput.*, vol. 58, p. 100745, 2020.

[22] Z. X. Pan, D. Lei, and L. Wang, A knowledge-based two-population optimization algorithm for distributed energy-efficient parallel machines scheduling, *IEEE Trans Cybern*, doi: 10.1109/TCYB.2020.3026571.

[23] K. Gao, Y. Huang, A. Sadollah, and L. Wang, A review of energy-efficient scheduling in intelligent production systems, *Complex Intell. Syst.*, vol. 6, pp. 237–249, 2020.

[24] H. Zhang, J. Xie, J. Ge, J. Shi, and Z. Zhang, Hybrid particle swarm optimization algorithm based on entropy theory for solving DAR scheduling problem, *Tsinghua Sci. Technol.*, vol. 24, no. 3, pp. 282–290, 2019.

[25] L. Zhang, N. R. Alharbe, G. Luo, Z. Yao, and Y. Li, A hybrid forecasting framework based on support vector regression with a modified genetic algorithm and a random forest for traffic flow prediction, *Tsinghua Sci. Technol.*, vol. 23, no. 4, pp. 479–492, 2018.

[26] Y. Zhang, G. Cui, Y. Wang, X. Guo, and S. Zhao, An optimization algorithm for service composition based on an improved FOA, *Tsinghua Sci. Technol.*, vol. 20, no. 1, pp. 90–99, 2015.

[27] Q. Zhang and H. Li, MOEA/D: A multiobjective evolutionary algorithm based on decomposition, *IEEE Trans Evol. Comput.*, vol. 11, no. 6, pp. 712–731, 2007.

[28] E. D. Jiang and L. Wang, An improved multi-objective evolutionary algorithm based on decomposition for energy-efficient permutation flow shop scheduling problem with

sequence-dependent setup time, *Int. J. Prod. Res.*, vol. 57, no. 6, pp. 1756–1771, 2019.

[29] F. Zhao, Z. Chen, J. Wang, and C. Zhang, An improved MOEA/D for multiobjective job shop scheduling problem. *Int. J. Comput. Integ. M.*, vol. 30, no. 6, pp. 616–640, 2017.

[30] A. Trivedi, D. Srinivasan, K. Sanyal, and A. Ghosh, A survey of multiobjective evolutionary algorithms based on decomposition, *IEEE Trans Evol. Comput.*, vol. 21, no. 3, pp. 440–462, 2016.

[31] R. Zhang and R. Chiong, Solving the energy-efficient job shop scheduling problem: A multi-objective genetic algorithm with enhanced local search for minimizing the total weighted tardiness and total energy consumption, *J. Clean. Prod.*, vol. 112, pp. 3361–3375, 2016.

[32] G. Lebbar, I. El Abbassi, A. El Barkany, A. Jabri, and M. Darcherif, Solving the multi objective flow shop scheduling problems using an improved NSGA-II, *Int. J. Oper. Quant. M.*, vol. 24, no. 3, pp. 211–230, 2018.

Ling Wang received the BEng degree in automation and the PhD degree in control theory and control engineering from Tsinghua University, Beijing, China, in 1995 and 1999, respectively. Since 1999, he has been working at the Department of Automation, Tsinghua University, where he became a full professor in 2008. His current research interests include intelligent optimization and scheduling. He has authored five academic books and more than 300 refereed papers.

He is a recipient of the National Natural Science Fund for Distinguished Young Scholars of China, the National Natural Science Award (Second Place) in 2014, the Science and Technology Award of Beijing City in 2008, the Natural Science Award (First Place in 2003 and Second Place in 2007) nominated by the Ministry of Education of China. He is now the editor-in-chief of *International Journal of Automation and Control*, and the associate editor of *IEEE Transactions on Evolutionary Computation*, *Swarm and Evolutionary Computation*, *Expert Systems with Applications*, etc.

Enda Jiang received the BS degree from Dalian University of Technology, Dalian, China in 2016. He is currently pursuing the PhD degree in Tsinghua University, Beijing, China. His main research interests include the distributed and green scheduling with intelligent optimization.

Jingjing Wang received the BS and MS degrees from Tsinghua University, Beijing, China in 2015 and 2018, respectively. She is currently pursuing the PhD degree in Tsinghua University. Her main research interests include the distributed and green scheduling with intelligent optimization.