IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received September 11, 2018, accepted September 26, 2018, date of publication October 2, 2018, date of current version October 31, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2873401

A Hybrid Pareto-Based Tabu Search for the Distributed Flexible Job Shop Scheduling Problem With E/T Criteria

JUN-QIANG LI[1,2,3], (Member, IEEE), PEIYONG DUAN[1], JINDE CAO[4], (Fellow, IEEE), XIAO-PING LIN[1], AND YU-YAN HAN[5]

[1]School of Information and Engineering, Shandong Normal University, Jinan 250014, China
[2]School of Economics and Management, Nanjing Forestry University, Nanjing 210037, China
[3]Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, Nanjing 211189, China
[4]Research Center for Complex Systems and Network Sciences, School of Mathematics, Southeast University, Nanjing 210096, China
[5]School of Computer, Liaocheng University, Liaocheng 252059, China

Corresponding authors: Peiyong Duan (duanpeiyong@sdnu.edu.cn) and Yu-Yan Han (hanyuyan@lcu-cs.com)

This work was supported in part by the National Science Foundation of China under Grants 61773246, 61773192, and 61803192, in part by the Key Laboratory of Computer Network and Information Integration, Southeast University, under Grant K93-9-2017-02, in part by the State Key Laboratory of Synthetical Automation for Process Industries under Grant PAL-N201602, and in part by the Major Basic Research Projects in Shandong under Grant ZR2018ZB0419.

ABSTRACT During recent years, distributed manufacturing optimization problems have been researched and applied in many fields, such as steelmaking system and textile production process. To solve the multi-objective distributed flexible job shop scheduling problem, a hybrid Pareto-based tabu search algorithm (HPTSA) is investigated to minimize four objectives simultaneously, i.e., the makespan, the maximal workload, the total workload, and the earliness/tardiness (E/T) criteria. In the proposed algorithm, several approaches considering both the problem characteristics and the objective features are used to initialize the group of solutions. Then, five types of neighborhood structures that consider both problem structures are developed to enhance the exploitation and exploration capabilities. In addition, a well-designed backward method is proposed to optimize the E/T criteria. Based on the realistic production data in the steelmaking system, several instances with different problem scales are randomly generated. Four efficient multi-objective optimization algorithms are selected to make detailed comparisons with the proposed HPTSA algorithm. After detailed tests on the realistic instances, the experimental comparison results show that the proposed algorithm shows competitive performance compared with the selected efficient algorithms.

INDEX TERMS Flexible job shop scheduling problem, tabu search, multi-objective optimization, Pareto archive set, distributed scheduling.

I. INTRODUCTION

Task scheduling is critical for many applications, which aims at assigning suitable resources or devices for the given tasks or jobs to minimize several certain objectives [1]–[4]. The flexible job shop scheduling problem (FJSP) is one branch of the classical job shop scheduling problem (JSP), which also can be considered as one type of task scheduling. In FJSP, one of the critical issues is to assign tasks to appropriate or optimal machines for processing with minimization of certain criteria. In addition, in FJSP, all of the jobs have machine selection flexibilities, which increase the problem complexity but improve the processing balance among all of the machines. Therefore, due to its immense value of practical applications, FJSP has gained more and more research focuses. It has been commonly agreed that JSP is one of the NP-hard problem [5]. Moreover, during recent years, distributed manufacturing has gained more and more research focuses. However, there is less literature considering the distributed FJSP problems.

During recent year, with more and more researches, multi-objective optimization algorithms have been developed and applied to solve many industrial problems. In multi-objective optimization problems, two or more conflicting objectives should be minimized or maximized simultaneously. When considering multiple objectives, a unique solution that is the best for all objectives may not exist. A solution with better fitness values in some objectives might have worse fitness values in other objectives. Alternatively, the target of the optimization algorithm is to get enough non-dominated solutions for the problem.

In this paper, we present a hybrid Pareto-based tabu search algorithm (HPTSA) for solving the distributed multi-objective FJSP with the minimization of the makespan, the maximal workload, the total workload, and the earliness/tardiness (E/T) criteria. First, several initialization methods are proposed to consider both the problem characteristics and the objective features. Then, considering the problem features, five types of neighborhood structures are investigated. Next, a well-designed backward method is proposed to optimize the E/T criteria. The rest of this paper is structured as follows: Section 2 briefly reviewed the related meta-heuristics and problems. Section 3 gives the problem description. Section 4 explains the algorithm framework. Section 5 reports the algorithm comparisons and analysis. Finally, Section 6 summarizes and describes the future works.

II. LITERATURE REVIEW

A. META-HEURISTICS

Recently, many types of meta-heuristics have been proposed to solve different problems. We can classify these meta-heuristics into two categories, local search meta-heuristics, and global search meta-heuristics. The local search meta-heuristics include many algorithms which focus on increasing the exploiting ability of the algorithm. Tabu search (TS) is one of the local search meta-heuristics which has been used to solve many engineering optimization problems. Some local search methods were also used for the flow shop scheduling problem [6], the lot-streaming flow shop scheduling problem [7], and the FJSPs [8]. However, the local search heuristic has many shortcomings, and many meta-heuristics consider both the local search and global search abilities. Table 1 gives a detailed description of the meta-heuristics for engineering problems.

TABLE 1. Literature about the meta-heuristics.

No.	Meta-heuristics	Problem	Reference
1	particle swarm optimization (PSO)	permutation flow shop scheduling problem (PFSP) polypropylene processes	[9] [10]
2	invasive weed optimization (IWO)	optimal chiller loading problem lot-streaming flow shop reconfigurable linear optimization problems	[11] [12] [13]
3	teaching-learning-based optimization (TLBO)	realistic flow shop rescheduling problems optimal chiller loading problem ethylene cracking furnace system	[14] [15] [16]
4	fruit fly optimization algorithm (FOA)	realistic hybrid flow shop rescheduling problem multidimensional knapsack problem continuous function optimization problems	[17] [18] [19]
5	artificial bee colony (ABC)	multi-objective FJSP with maintenance activities dynamic co-evolution method hierarchical communication model hybrid flexible flow shop problem	[20] [21] [22] [23]
6	chemical-reaction optimization (CRO)	FJSP multi-area environmental/economic dispatch	[24] [25]
7	estimation of distribution algorithm (EDA)	anisotropic adaptive variance scaling domain adaptation and nonparametric estimation	[26] [27]
8	migrating birds optimization (MBO)	permutation flow shop problem	[28]
9	artificial fish swarm algorithm	Optimal chiller loading problem	[29]
10	GA–Elman algorithm	neural network problem	[30]
11	biogeography-based optimizer	cooperative co-evolutionary method	[31]

B. THE MULTI-OBJECTIVE ALGORITHMS

Most of the published multi-objective optimization algorithms can be divided into two categories, i.e., the Pareto-based multi-objective optimization algorithm [32]–[39], and the decomposition-based multi-objective optimization algorithm [40]–[46].

The Pareto-based method has been used by researchers. However, the crowding distance and distribution and diversity of the population are the two major issues of the Pareto-based methods, which make the Pareto-based algorithm efficient for solving two or three objectives but weak in solving optimization problems with more than three objectives. For example, Battiti and Passerini [34], Li *et al.* [35], [37], and Yi *et al.* [36] verify the efficiency of the Pareto-based optimization methods for solving optimization problems with two objectives. Gao *et al.* [38] applied the Pareto-based harmony search algorithm to optimize two objectives in FJSPs, namely the maximum completion time (makespan) and the mean of earliness and tardiness. Li and Huo [39] considered three objectives to reduce delivery delay, minimization idleness of machines and interruption in production.

MOEA/D is one of the most important methods among the decomposition based multi-objective optimization algorithms. Since it has been designed, many researchers have improved the canonical MOEA/D algorithm. The MOEA/D has advantages in solving continuous optimization problems [40]–[46], but fewer literatures have applied it to solve discrete optimization problems, such as the scheduling problem in this study.

Table 2 gives a detailed description of the multi-objective algorithms.

TABLE 2. Literature about the multi-objective algorithms.

No.	Type	Method	Problem	Ref
1	Pareto Based algorithms	multi-objective particle swarm optimization algorithm	feature selection in classification	[32]
2		multiple objective custom genetic algorithm	multi-state reliability optimization design	[33]
3		Reactive Search Optimization (RSO)	interactive optimization problem	[34]
4		local search NSGA-II	reverse logistics network	[35]
5		Multi-objective bacterial foraging optimization	aluminum electrolysis production process	[36]
6		Efficient multi-objective optimization algorithm	hybrid flow shop scheduling problems	[37]
7		Multi-objective Harmony Search	FJSPs	[38]
8		modified genetic algorithm	FJSPs	[39]
9	Decomposition based algorithms	Balancing Convergence and Diversity	Many-Objective Optimization	[40]
10		Two-Archive Algorithm	Many-Objective Optimization	[41]
11		Angle-Based-Selection	Many-Objective Optimization	[42]
12		Surrogate-based MOEA/D	electric motor design	[43]
13		Adaptive replacement strategies	Many-Objective Optimization	[44]
14		decomposition-based algorithm	Constrained problems	[45]
15		Multi-objective artificial bee algorithm	Many-Objective Optimization	[46]

C. TASK SCHEDULING AND OPTIMIZATION PROBLEMS

Recently, task scheduling in cloud systems has been studied. Li *et al.* [47] studied the task scheduling in the resource-constrained steelmaking scheduling problems. Liu *et al.* [48] investigated the task assignment problem in a multi-agent design system. Zhang and Liu [49] further studied the task assignment in cloud systems. Wang *et al.* [50] considered the heterogeneous scheduling problems with energy-efficiency features. Wnag *et al.* [4], [51] also developed an artificial swarm intelligence method and parallel heuristics. The above literature discussed task assignment without considering the flexible capabilities of the cloud system. For optimization problems in different types of systems, many literatures have investigated different types of optimization problems, such as green communications optimization problems [52], [53], support vector machine problem [54], fuzzy clustering and deflection problem [55], recurrent Neural Network optimization problem [56], crowd simulation based on computational intelligence method [57], group recommendation problem [58], Supervised Feature Learning problem [59], face recognition [60], localization prediction problem [61], multi-features fusion optimization [62], linear or nonlinear optimization problems [63]–[68], and multi-agent systems optimization problems [69]–[72].

Considering the flexible task assignment in cloud systems, many literatures have modeled it as a hybrid flow shop scheduling (HFS) problem. Ruiz and Vázquez-Rodríguez [73] reviewed the literature about HFS published before 2010. Very recently, the HFS problems have been considered using meta-heuristics, such as migrating birds optimization [74], [75], self-tuning iterated greedy (SIG) algorithm [76], hybrid algorithm combining ant system and GA [77], variants of iterated greedy [78], hybrid artificial bee colony algorithm [79], variable neighborhood search algorithm [80], and iterated search methods [81].

D. FLEXIBLE JOB SHOP SCHEDULING PROBLEM

Recently, the FJSP problems have been researched with many meta-heuristics, such as discrete virus optimization algorithm [82], game theory based multi-objective algorithm [83], a memetic algorithm considering worker flexibility [84], shuffled frog-leaping algorithm (SFLA) [6], a hybrid fruit fly algorithm to reduce manufacturing carbon footprint [85], the multi-objective harmony search algorithm [38], the modified genetic algorithm [39], and the energy-efficient multi-objective optimization algorithm [86]. From the above literature, we find that there is less literature that considers the multi-objective FJSPs. Therefore, in this study, we consider a novel Pareto-based tabu search algorithm considering four objectives.

For considering the distributed manufacturing problems, Rifai *et al.* [87] developed a novel multi-objective adaptive large neighborhood search (MOALNS) algorithm to simultaneously satisfy three objectives of minimizing makespan, total cost and average tardiness values in consideration of the reentrant characteristic of DPFSP. Lin and Ying [88] solved the no-wait flowshop scheduling problem (DNFSP) by developing a mixed integer programming (MIP) mathematical model and an iterated cocktail greedy (ICG) algorithm. Bargaoui *et al.* [89] investigated an artificial chemical reaction meta-heuristic to minimize the maximum completion time. Ying *et al.* [90] presented an Iterated Reference Greedy (IRG) algorithm for the distributed no-idle permutation flowshop scheduling problem (DNIPFSP) with the objective of minimizing the makespan. Komaki and Malakooti [91] minimized the makespan of the distributed no-wait flow shop scheduling problem utilizing a general variable neighborhood search (GVNS) algorithm. Deng and Wang [92] developed a competitive memetic algorithm (CMA) to solve the multi-objective DPFSP using the makespan and total tardiness criteria. Lin *et al.* [93] addressed the distributed assembly permutation flow-shop scheduling problem (DAPFSP) using a backtracking search hyper-heuristic (BS-HH) algorithm. Zhang *et al.* [94] solved the distributed flowshop scheduling problem with flexible assembly and setup time using a constructive heuristic (TPHS) and two hybrid meta-heuristics.

III. PROBLEM DESCRIPTION

A. PROBLEM DESCRIPTION

Gao and Pan [95] considered the steelmaking continuous-casting process as an extension of the classical FJSPs.

Li and Huo [39] also considered the steel tube production system as a multi-objective FJSPs. In this study, we investigated the distributed features of the steelmaking production system and considered it as a distributed FJSP. In the distributed FJSP problem, there are n jobs, m machines, and s distributed factories. Each job has its own pre-defined and deterministic processing stages, and which constructs a set of operations for it. To schedule each job, the first task is to decide which factory should be assigned to the job. After assigning a certain factory, the job should be processed through pre-defined stages. In each pre-defined stage, the operation has a set of candidate machines for processing, which is the second task for the problem, that is, to select a suitable machine from candidate machines for each job. After assigning the candidate machine for each job, the third task is to schedule all jobs on the assigned machine in each factory. Therefore, the key issues for the distributed FJSP are as follows.

- Assign a suitable factory for each job;
- Assign a suitable machine for each job from a candidate machine set;
- Schedule all jobs on each assigned machine in each factory.

The general constraints for the distributed multi-objective FJSP with E/T criteria are given as follows:

- Each job should select exactly one factory;
- Each operation should select exactly one machine in each stage;
- Each machine can process at most one operation at a given time;
- Each operation can be processed at most on one machine at a given time;
- The processing of any operation cannot be interrupt during its processing;
- Each operation can be transported to its following stage after the completion of the current stage;
- On any machine, the overlap of the processing different operations is not permitted;
- The disruption events, such as machine breakdown, job insertion, and job cancelation are not considered; and
- The processing time for any operation on any suitable machine is pre-defined and deterministic.

In this study, we examine the distributed FJSP with four objectives, minimization of the makespan, the maximal workload, the total workload, and the earliness/tardiness (E/T) penalty. The E/T penalty is used to assign a lower fitness value to the solutions which can not release the needed jobs at a given due date. The E/T penalty aims to allow the completion time of all jobs at the given dates to maximize the economic profit. Earliness is not always profitable because the jobs which have been completed before the due date need an additional memory buffer. Tardiness is not beneficial for the economic profit because it cannot satisfy the customer's demand. Therefore, the algorithm should consider both the earliness and the tardiness to maximize the economic profit. The notations used in this study are given in Table 3.

TABLE 3. Notations employed in the paper.

i	Index of jobs, i=1, 2,…,n
n	Total number of jobs
$\mathbb{J}$	The set of jobs $\mathbb{J}=\{J_1, J_2,\ldots,J_n\}$
n_i	The total number of operations of the job J_i
j	Index of stages
k	Index of machines
m	Total number of machines
f	Index of factories
s	Total number of factories
m_k	Total number of operations being processed on the k^{th} machine
$O_{i,j}$	The j^{th} operation of the job J_i
$M(O_{i,j})$	The set of candidate machines for the operation O_{ij}
$p_{i,j,k}$	The processing time of $O_{i,j}$ on the k^{th} machine
W_k	The workload of the k^{th} machine, which is the total processing time of operations that are operated on it
W_T	Total workload in the system, which is the sum of all the processing times.
C_i	The completion time of the last operation of job J_i
C_{max}	The makespan of the system, which is the maximal completion time of all jobs
S_k	The start time point of the due date time window for the k^{th} machine
E_k	The end time point of the due date time window for the k^{th} machine
CM_k	The completion time of the k^{th} machine

In addition, to apply the Pareto-based optimization algorithm, which is efficient for solving problem with two or three objectives, we combine the second and the third objectives into one objective. Based on the above notations, the objectives are used in this study:

1) minimization of maximum completion time (makespan):

$$F_1 = \max\{C_i | i = 1, 2, \ldots, n\} \tag{1}$$

2) minimization of the workloads:

$$F_2 = W_{max} + W_T \tag{2}$$

where the critical machine workload W_{max} is computed as follows: $W_{\max} = \max\{W_k | k = 1, 2, \ldots, m\}$, and the minimization of total workload W_T is computed as follows:

$$W_T = \sum p_{i,j,k}, \quad i = 1, 2, \ldots, n; \quad k = 1, 2, \ldots, m; \ \forall j.$$

3) minimization of E/T penalty:

$$F_3 = \sum_{k=1}^{m} (\max(0, S_k - CM_k) + \max(0, CM_k - E_k)) \tag{3}$$

B. PROBLEM EXAMPLE

Fig. 1 gives an example of the distributed FJSP problem in Gantt chart, where there are two factories, f_1 and f_2. In each factory, there are five machines which construct

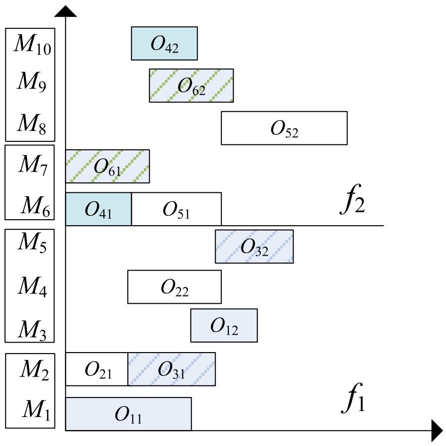


FIGURE 1. An example of Gantt for the distributed FJSP.

TABLE 4. Processing time table.

factory	machine	Job					
		J_1	J_2	J_3	J_4	J_5	J_6
	M_1	30	20	30	15	25	30
	M_2	40	15	25	20	30	25
f_1	M_3	15	30	25	30	40	30
	M_4	20	25	30	25	45	25
	M_5	20	30	20	20	50	30
	M_6	30	20	30	15	25	30
	M_7	40	15	25	20	30	25
f_2	M_8	15	30	25	30	40	30
	M_9	20	25	30	25	45	25
	M_{10}	20	30	20	20	50	30

two stages. In the first stage of each factory, there are two parallel machines and three identical machines in the second stage. Six jobs are to be scheduled, and among them, the first three jobs are assigned to the first factory and the second three jobs are processed in the second factory. In the first factory, each job has two operations which should be assigned to one machine from the parallel machines in each stage. For example, in the first stage of the first factory, the operation O_{11} is assigned to the machine M_1. The machine assignment for the first factory is as follows: $\{\langle O_{11},M_1\rangle, \langle O_{21},M_2\rangle, \langle O_{31}, M_2\rangle, \langle O_{12}, M_3\rangle, \langle O_{22}, M_4\rangle, \langle O_{32}, M_5\rangle\}$. In each assigned machine, all of the waiting jobs should be scheduled, for example, on M_2, the processing sequence of the assigned jobs are $\{O_{21}, O_{31}\}$. The processing time for each job on each machine is given in Table 4.

From the example Gantt chart, we can see that the main difference of the distributed FJSP and the canonical FJSP is that, in the former, all jobs should first be assigned the processing factory. Therefore, the distributed FJSP is harder than the classical FJSP.

IV. THE PROPOSED ALGORITHM

This section presents the components of the proposed algorithm. First, the framework of the proposed algorithm is described. Then, the other components are detailed in the following sub-sections.

A. THE FRAMEWORK OF HPTSA

The detailed steps of the proposed HPTSA algorithm are given in Algorithm 1.

B. CODING

In order to represent the machine assignment and operation scheduling simultaneously, we design a three-component-based vector for the solution representation. For example, given a solution {{1, 1, 1, 2, 2, 2}, {1, 3, 2, 4, 2, 5, 6, 10, 6, 8, 7, 9}, {1, 2, 3, 2, 1, 3, 4, 6, 5, 4, 6, 5}, the detailed description of the solution representation is as follows.

(1) Factory assignment vector (hereafter called A_1): In each factory assignment vector $A_1 = \{A_1(1), A_1(2), \ldots, A_1(n)\}$, where $A_1(i), i = 1, 2, \ldots n$ represents the corresponding assignment device for the i^{th} operation. In Fig. 1, the factory assignment vector can be considered as follows:

$$f_1\text{: }\{J_1, J_2, J_3\} \quad \text{and } f_2 : \{J_4, J_5, J_6\}.$$

(2) Routing vector (hereafter called A_2): In each routing vector $A_2 = \{A_2(1), A_2(2), \ldots, A_2(\kappa)\}$, where $A_2(i)$, $i = 1, 2, \ldots \kappa$ represents the corresponding assignment device for the i^{th} operation, where κ denotes the number of operations. In Fig. 1, the routing vector can be considered as follows:

$$f_1\text{: } \{\langle O_{11}, M_1\rangle, \langle O_{12}, M_3\rangle, \langle O_{21}, M_2\rangle, \langle O_{22}, M_4\rangle, \langle O_{31}, M_2\rangle, \langle O_{32}, M_5\rangle\}$$
$$f_2\text{: } \{\langle O_{41}, M_6\rangle, \langle O_{42}, M_{10}\rangle, \langle O_{51}, M_6\rangle, \langle O_{52}, M_8\rangle, \langle O_{61}, M_7\rangle, \langle O_{62}, M_9\rangle\}$$

(3) Scheduling vector (hereafter called A_3): In the scheduling vector, each operation belonging to the same job is represented with the same integer number, and the occurrence sequence of the integer number represents the operation one by one. For the example solution in Fig. 1, the corresponding scheduling vector can be interpreted as follows:

$$f_1 : \{1, 2, 3, 2, 1, 3\} \rightarrow \{O_{11}, O_{21}, O_{31}, O_{22}, O_{12}, O_{32}\}$$
$$f_2 : \{4, 6, 5, 4, 6, 5\} \rightarrow \{O_{41}, O_{61}, O_{51}, O_{42}, O_{62}, O_{52}\}$$

C. INITIAL SOLUTIONS

The initial solutions are used to initialize the population, from which the initial solution for the TS algorithm was selected, and the initial Pareto archive set was generated. The factory assignment vector is generated in a random way, that is, to assign a random available factory for each operation. To increase the performance of the initial solutions, the initial population was generated according to two types of priority rules: routing initial rules and scheduling initial rules.

1) ROUTING INITIAL RULES

- Random rule is denoted as $\boldsymbol{MS_a}$.
- Local minimum processing time rule is denoted as $\boldsymbol{MS_b}$. For each operation of the same job, select the machine with the minimum processing time and fix the selection.

Algorithm 1 General Framework of HPTSA

input: system parameters
output: the Pareto set

for $i \leftarrow N$ **do**
  Initialize the factory assignment component of each solution in a random way.
  Initiate the routing component of each solution randomly use the random rule, the operation minimum processing time rule (OPT), and the global minimum processing time rule (GPT).
  Initiate the scheduling component of each solution randomly use the most work remaining rule (MWR), the most number of operations remaining (MOR) rule, and the shortest processing time rule (SPT).
end
Evaluate each solution and initialize the Pareto archive set, select the best one as the initial solution for the TS algorithm
If the stopping criterion is satisfied, output the Pareto archive set. Otherwise, perform the following steps
while stop criterion is not satisfied **do**
  Perturbation in the factory assignment component phase.
  for $i \leftarrow N$ **do**
    Produce a neighboring solution by applying the neighborhood structure V to the current solution.
    Evaluate the neighboring solution, and insert it into a solution set.
  end
  Perturbation in the routing component phase.
  for $i \leftarrow N$ **do**
    Produce a neighboring solution by applying the function neighborhood structure I, Π, III to the current solution.
    Insert the neighboring solution into a solution set
  end
  Apply the non-dominated sort algorithm to the current neighboring population.
  Update the tabu list by adding the best neighboring solution and removing the oldest solution.
  Update the Pareto archive set.
  Perturbation in the scheduling component phase
  for $i \leftarrow N$ **do**
    Produce a neighboring solution by applying the critical block neighboring structure to the current solution
    Insert the neighboring solution into a solution set
  end
  Apply the non-dominated sort algorithm to the current neighboring population.
  Update the tabu list by adding the best neighboring solution and removing the oldest solution.
  Update the Pareto archive set.
end

- Global minimum processing time rule is denoted as $\boldsymbol{MS_c}$. From the processing timetable, find the global minimum processing time, fix the assignment, then add the selected processing time to every other entry in the same column.

2) SCHEDULING INITIAL RULES

- Random rule is denoted as $\boldsymbol{OS_a}$.
- Last Processing rule, denoted as $\boldsymbol{OS_b}$. The detailed steps are as follows: (1) calculate the completion time for each machine; (2) record each machine with the completion time equal to the current makespan into a set named M_{lp}; (3) mark all of the critical operations; (4) randomly select a critical operation and assign it to another different candidate machine.
- EDD (Earliest Due Date) rule, denoted as $\boldsymbol{OS_c}$. First, sort all operations according to the due date in non-decreasing order. Then, select the operation with the earliest due date to schedule on the assigned machine.
- Modified due date rule, denoted as $\boldsymbol{OS_d}$. This rule combines the EDD rule and the total processing time of the operations belonging to the same job which have been scheduled before $O_{i,j}$. First, we define a modified due date as follows:

$$d_{i,j} = \begin{cases} d_i - \sum_{h=1}^{j-1} p_{i,h,j}, & if\ j > 1 \\ d_i, & otherwise \end{cases}$$

where d_i is the due date of the job J_{i}, $\sum_{h=1}^{j-1} p_{i,h,j}$ is the total processing time of the operations belonging to the job J_{i} which have been scheduled before $O_{i,j}$. Then, sort the remaining operations according to the modified due date $d_{i,j}$ in non-decreasing order.

D. NEIGHBORHOOD STRUCTURES

For the problem considered, we design five types of neighborhood structures as follows.

1) NEIGHBORHOOD STRUCTURE I

The main procedures of the first neighborhood structure are described as follows:

Step 1. For each machine, compute the total number of critical operations;
Step 2. Sequence all of the machines according to the number of critical operations in descending order;
Step 3. For the first machine in the resulted sequence, denotes it as M_{old}, randomly select a critical operation and denoted it as O_{s};
Step 4. For the selected critical operation O_{s}, from its candidate machine set, select a machine with relative less number of critical operations;
Step 5. Replace the routing component.

2) NEIGHBORHOOD STRUCTURE II

The neighborhood structure II is based on the critical path theory. Fig. 2 shows a Gantt chart of a critical path, where the critical path is the longest path with critical operations.

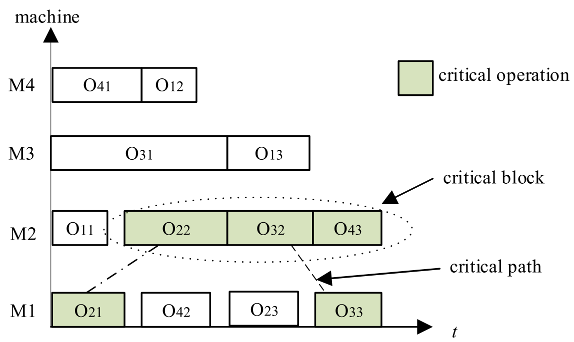


FIGURE 2. Gantt chart for a critical path example.

The critical operation is the operation which cannot be right or left shift given the maximal completion time. For example, in Fig. 2, the operation O_{22} is the critical operation and any shift of it will affect the makespan of the solution. However, the operation O_{12} is not the critical operation, because we can right shit it without any effect of the makespan.

It can be seen in Fig. 2, the example solution contains two critical paths, i.e., $CP_1 = \{O_{21}, O_{22}, O_{32}, O_{43}\}$, and $CP_2 = \{O_{21}, O_{22}, O_{32}, O_{33}\}$. In Fig. 2 the critical path CP_1 is divided into two blocks, $B_{11} = \{O_{21}\}$ and $B_{12} = \{O_{22}, O_{32}, O_{43}\}$, while the critical path CP_2 is divided into three blocks, $B_{21} = \{O_{21}\}$, $B_{22} = \{O_{22}, O_{32}\}$, and $B_{23} = \{O_{33}\}$, where any of the operation in the block is called a critical operation. The two critical paths contain two public adjacent critical operations, i.e., $\{O_{22}, O_{32}\}$, which construct a public critical block. The example solution shown in Fig. 2 contains five critical operations, i.e., $\{O_{21}, O_{22}, O_{32}, O_{33}, O_{43}\}$. The detailed steps of the neighborhood structure Π are shown as follows:

Step 1. Get all critical operations of the current solution;

Step 2. Randomly select a critical operation $O_{i,j}$ (with a position in the routing component of the current solution denoted by $pos_{i,j}$) with at least two candidate devices, denote the current machine as M_k. For each candidate machine other than M_k, denote as $M_k^{'}$. If one of the following criterions is satisfied, replace M_k with $M_k^{'}$ at position $pos_{i,j}$ in the routing component of the current solution. 1) The newly-resulted $p_{i,j,k'}$ is smaller than $p_{i,j,k}$ (to improve the total workload). 2) M_k is the busiest machine and therefore $W_{k'} + p_{i,j,k'} < W_{\max}$ (to improve the critical workload).

3) NEIGHBORHOOD STRUCTURE III

The neighborhood structure III is embedded in the scheduling algorithm to minimize the makespan.

a: PUBLIC CRITICAL BLOCK

In Fig. 2, we give an example of a Gantt chart form for a feasible solution with two critical paths. The adjacent critical operations belonging to at least two critical paths are called a public critical block. For example, in Fig. 2, the three operations $\{O_{21}, O_{22}, O_{32}, O_{43}, O_{33}\}$ constructs a critical block, and which is the only block in the Gantt chart. It should be noted that, in the critical block, the critical operation in the first position is called the block head, while the ending critical operation in the block is called the block tail. The block head and block tail have an important role in the local search.

b: LOCAL SEARCH

To improve the local search abilities, three types of neighborhood structures are developed, which are named Swap, Insert_I, and Insert_Π, respectively. Assume that a public critical block in a feasible solution is denoted by $PB = \{pb_1, pb_2, \ldots pb_c\}$, where pb_1 and pb_c are the first and ending operations, respectively, of the public critical block. $pb_2, \ldots pb_{c-1}$ are the remaining operations of the critical block. First, we define three functions: (1) swapping x with y is denoted by *swap(x, y)*; (2) inserting x just after y is denoted by $insert_\Phi(x, y)$; (3) inserting x just before y is denoted by $insert_\Psi(x, y)$. Next, we give three neighborhood structures as follows:

1) *Swap* neighborhood structure.

$$\Pi_{swap} = \{swap(pb_1, x)|x \in PB - \{pb_1\}\} \cup swap(pb_c, x)|x \in PB - \{pb_c\}\}\} \quad (4)$$

2) *Insert* I neighborhood structure.

$$\Pi_{insert} = \{insert_\Phi(pb_1, x)|x \in PB - \{pb_1\}\} \cup insert_\Psi(pb_c, x)|x \in PB - \{pb_c\}\}\} \quad (5)$$

3) *Insert* II neighborhood structure.

$$\Pi_{insertII} = \{insert_\Phi(x, pb_c)|x \in PB - \{pb_c\}\} \cup insert_\Psi(x, pb_1)|x \in PB - \{pb_1\}\}\} \quad (6)$$

The detailed local search procedure is shown in Algorithm 2.

4) NEIGHBORHOOD STRUCTURE IV

The neighborhood structure IV is embedded in the scheduling algorithm to minimize the E/T criteria. Thus, the ideal schedule is the one which schedules each operation just following its due date. Before the modified backward procedure, we first give the following notations in Table 5.

The modified backward procedure is given in Algorithm 3.

TABLE 5. Notations for the modified backward procedure.

notation	representation	notation	representation
$e_{i,k}$	possible start time of $O_{i,k}$	$p_{i,k}$	the processing time of $O_{i,k}$
$O_{i,\lambda}$	the last operation of job J_i	I_m	idle time of machine M_m
r_i	possible start time of job J_i	$d_{\max}$	the maximum due date
n_i	the number of operations of job J_i	d_i	the due date of job J_i

Algorithm 2 Local_Search

input: a set named M_{pb} which contains all the critical operations.
output: the best neighboring solution of the current solution

1.	Set the global best solution gbest with a large value.
2.	**for** each public critical block *pb* in M_{pb} **do**
3.	if there are two or more operations in pb, then
4.	Apply the *swap*, *InsertI* and *Insert*Π neighborhood structures in Equations (4), (5) and (6) to search the neighboring solutions.
5.	Evaluate each neighboring solution; replace the *gbest* with the best new solution if the latter is better.
6.	**end**
7.	**end**
8.	Output *gbest* as the best neighboring solution.
9.	**end**

5) NEIGHBORHOOD STRUCTURE V

The neighborhood structure V is to generate a different neighboring solution by randomly changing the assigned factory for one or two jobs. The detailed implements of the neighborhood structure V are to perform the following two approaches in a random way.

(1) One job changing approach: This type of approach performs the following steps: first, randomly select one job from the factory with most workload; then, randomly assign a different factory to process the selected job.

(2) Two job changing approach: This type of approach performs the following steps: first, randomly select two jobs with different processing factories; then, swap the two assigned factories for the two selected jobs.

6) EXAMPLE OF THE NEIGHBORING STRUCTURES

Given the routing component of a feasible solution, {2, 2, 1, 1, 2, 1}, the problem is two jobs to be processed on two machines, and each job has three operations. The due date and the coefficient for the earliness and tardiness are given in Table 6. For the scheduling component, we apply the above two neighborhood structure to decide the sequence of each operation on each machine. The *neighborhood structure III* considers the makespan criteria while the *neighborhood structure IV* considers the E/T criteria. Fig. 3(a) gives a schedule applying the first neighborhood structure but without left-shift. Fig. 3(b) gives a schedule applying the first neighborhood structure with left-shift. Fig. 3(c) gives a schedule applying backward procedure while Fig. 3(d) gives a schedule applying the modified backward procedure. The fitness values of the two objectives are as follows:

Algorithm 3 Modified_Backward

input: a feasible solution
output: an modified feasible solution

1.	**Initialization phase**
2.	Let $U = \{O_{i,n_i} \mid i = 1, 2, \ldots.., n\}$
3.	Calculate the possible start time of each job: $r_i = d_{\max} - \max\{d_i, \sum_{k=1}^{n_i} p_{i,k}\}$
4.	Set the idle time of each machine: $I_m = 0$
5.	Set the possible earliest start time of each operation in U: $e_{i,k} = r_i$
6.	**end**
7.	**Scheduling phase**
8.	**While** (U is not empty) **do**
9.	Calculate the possible completion time of each operation: $Q_{i,k} = e_{i,k} + p_{i,k} \forall O_{i,k} \in U$
10.	Find the operations with the minimum possible completion time and the assigned machine m^* for the operation: $Q^* = \min\{Q_{i,k} \mid O_{i,k} \in U\}$
11.	Memory all operations found so far in the set C: $C = \{O_{i,k} \mid O_{i,k} \in U \wedge Q_{i,k} \leq Q^* \wedge m_{i,k} = m^*\}$
12.	Calculate the priority index for all operations in the set C: $S_{i,k} = k \times ((\alpha_i + \beta_i)/p_{i,k})$
13.	Select the operation with the maximum priority index denoted by $O^*_{i,k}$, and then schedule it on the assigned machine m^*
14.	Update the idle time of machine m^*: $I'_{m^*} = \max\{I_{m^*}, e_{i,k^*}\} + p_{i,k^*}$
15.	Delete $O^*_{i,k}$ from U
16.	If $O^*_{i,k}$ is not the first operation of job J_i, insert $O_{i,(k-1)}$ to U
17.	Update the possible start time of the operations $O_{i,(k-1)}$ in U: $e_{i,(k-1)} = \max\{Q_{i,k}, I_{m_{i,(k-1)}}\}$
18.	**end**
19.	**Calculate the fitness value.**
20.	Calculate the $C_{\max}$ of the obtained schedule
21.	Calculate the E/T value of the obtained schedule
22.	**end**
	Right-shift procedure
23.	If $C_{\max} > d_{\max}$, then perform step4.1 to step4.5 to consider right-shift.
24.	Record the idle time interval of each machine in the form of several pairs of <start, end>
25.	Sort all operations according to the processing time in non-decreasing order, the position corresponding to $O_{i,k}$ denoted by $L_{i,k}$
26.	Create a set of flags denoted by F to indicate whether an operation needs a right-left move, whose length equals the total number of operations. The initial flag for each operation is set to *false*
27.	For each machine, from right to left, check whether an operation being operated can be moved to the right. Then, set the flag to *true* corresponding to the selected operation
28.	select the operation $O'_{i,k}$ by following the rule, and then move it to the right position on the machine. $O'_{i,k} = \{O_{i,k} \mid \max(L_{i,k}) \wedge F(O_{i,k}) = true\}$ where $F(O_{i,k})$ represents the flag for a move right of $O_{i,k}$
29.	**end**
30.	**end**

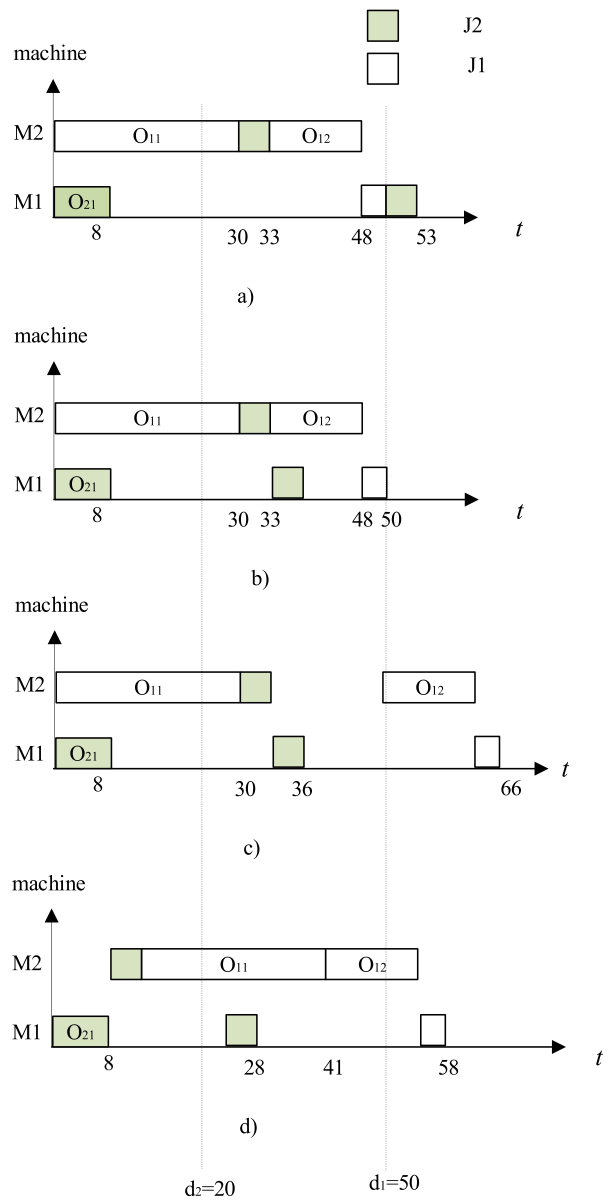


FIGURE 3. Gantt form for the scheduling example.

1) Schedule without left-shift: makespan = 53, E/T = 189;
2) Schedule with left-shift: makespan = 50, E/T = 96;
3) Schedule with backward procedure: makespan = 66, E/T = 144;
4) Schedule with modified backward procedure: makespan = 58, E/T = 72.

From the above results we can see that, the results obtained by (2) and (4) are two optimal solutions, while the other two solutions are not optimal. Therefore, in the scheduling phase, *neighborhood structure III* and *neighborhood structure IV* are applied for minimizing the makespan and the E/T criteria, respectively.

V. EXPERIMENT RESULTS

This section discusses the computational experiments that were used to evaluate the performance of the proposed algorithm. Our algorithm was implemented in C++ on an Intel Core i5 3.3 GHz PC with 4GB memory. The compared algorithms were NSGA-II [96], MOEA/D [97], DHS [98] and EEM [86]. All five compared algorithms utilize the same coding mechanism, the same initialization function, and the same stopping criterion.

In order to made detailed comparisons for solving the FJSP problems with due date constraints, we randomly generated 20 realistic instances after considering the processing data from the Baosteel industries. Several constraints are described as follows.

- There are 10 charges, which can be divided into 1, 2, 3, 5, 6, or 10 sub-lots and are to be processed in five or ten stages.
- The processing times for each sub-lot are randomly generated in the range of [30, 40].
- The release time for each machine is set to zero.
- The transfer times between consecutive phases are generated randomly in the range of [10, 15].
- The processing time for each job contains the setup time.
- The start and end of the due date window are set to $[1440 - \delta, 1440 + \delta]$ according to the minute numbers of a whole working day, where δ is a random integer number in [0, 720].

TABLE 6. Example for a feasible solution.

<operation, machine, processing time>	d_i	α_i	β_i
$(O_{11}, M_2, 30)$	50	5	3
$(O_{12}, M_2, 15)$	50	5	3
$(O_{13}, M_1, 2)$	50	5	3
$(O_{21}, M_1, 8)$	20	2	6
$(O_{22}, M_2, 3)$	20	2	6
$(O_{23}, M_1, 3)$	20	2	6

A. SETTING PARAMETERS

The detailed descriptions of the system parameters are given as follows:

- The population size P_{size} is set to 1000;
- The stop of the criterion is set to $n \times m$ iterations;
- The maximum non-improvement local search parameter $iter_{\max}$ is set to $op\_num/2$;

1) Tabu tenure: The tabu tenure in the algorithm ranged from $Tenure_{min} = op\_num/2$ to $Tenure_{max} = op\_num/2$, where $Tenure_{min}$ and $Tenure_{max}$ represents the minimum and maximum values of the tabu tenure, respectively, and op_num denotes the total number of operations. The adjustment feature of the tabu tenure $Tenure_c$ is given as follows:

$$Tenure_c = Tenure_{min} + (Tenure_{max} - Tenure_{min}) \times (t/T_{max}).$$

TABLE 7. Comparisons of the Pareto number for 20 realistic problems.

Problem	Scale	Pareto number values				
		NSGA-II	MOEA/D	DHS	EMM	HPTSA
Case1	50-jobs	2	3	3	4	**6**
Case2	50-jobs	3	4	3	2	**5**
Case3	50-jobs	0	0	3	3	**5**
Case4	50-jobs	1	3	2	**4**	3
Case5	50-jobs	2	2	3	2	**4**
Case6	100-jobs	0	**5**	1	**5**	**5**
Case7	100-jobs	3	3	3	3	**5**
Case8	100-jobs	2	2	2	4	**5**
Case9	100-jobs	1	1	2	2	**10**
Case10	100-jobs	0	1	2	5	**6**
Case11	150-jobs	2	3	3	3	**5**
Case12	150-jobs	0	2	2	4	**8**
Case13	150-jobs	1	2	2	2	**8**
Case14	150-jobs	2	1	1	5	**7**
Case15	150-jobs	3	1	2	3	**5**
Case16	200-jobs	2	2	3	1	**6**
Case17	200-jobs	0	2	2	2	**5**
Case18	200-jobs	1	2	3	**5**	**5**
Case19	200-jobs	0	1	2	5	**4**
Case20	200-jobs	0	1	3	**5**	**5**
Mean		1.25	2.05	2.35	3.45	**5.6**

- the best values are in bold

2) Neighborhood size: The neighborhood size represents the searching strength, which ranged from $nb_{min} = op\_num/5$ to $nb_{max} = op\_num$.
3) Tabu element: In this study, the weighted sum of the three objectives ($F(c)$) was employed as the structure of the tabu list.

B. COMPARISON METRICS

To test the performance of the proposed algorithm, we utilized the three performance metrics that were discussed in [99], i.e., the average Pareto distance V_{pd}, total number of optimal solutions or non-dominated solutions V_{np}, and the ratio of the non-dominated solutions V_{rd}. Let S^P denote the reference solution set which was obtained by running all the compared algorithms for 3000 iterations. Let S^j ($j = 1, 2, 3, 4$) represent the non-dominated solution set that was obtained by algorithm j, where $S^P = \cup S^j$. Then, the detailed computation processes of the three metrics are as follows.

1) AVERAGE PARETO DISTANCE V_{pd}

Let $V_{pd} = \frac{1}{|S^P|}\sum_{y\in S^P} d_y(S^P)$ and

$$d_y\left(S^j\right) = \{\sum_{i=1}^{2}(\frac{f_i(x) - f_i(y)}{f_i^{max}(.) - f_i^{min}(.)})^2\},\quad y \in S^P,$$

where $f_i(.)$ represents the ith objective value, and $f_i^{max}(.)$ is the maximum value of the ith objective value in the Pareto referent point set S^P, whereas, $f_i^{min}(.)$ is the minimum value. $d_y\left(S^j\right)$ represents the shortest normalized distance from a reference solutions y in S^P to the solution set S^j. V_{pd} indicates the average of those shortest normalized distances from all the reference points to the solution set S^j.

The average Pareto distance is usually used to evaluate the spread and distribution of the obtained solution set. That is, a smaller V_{pd} indicates that the obtained solution set has better distribution and better approximation to the reference set S^P. The most promising situation is that V_{pd} equals 0, which means that the set of obtained solutions is equal to the reference point set.

2) NUMBER OF NON-DOMINATED SOLUTIONS V_{np}

The number of non-dominated solutions is the number of obtained solutions that are not dominated by the reference solutions. A larger value of V_{np} indicates that there are more non-dominated solutions in the obtained solutions set S^j. The computational process uses the following formulation:

$$V_{np} = \{S^j - \{x \in S^j | \exists y \in S^P : y \prec x\}\},$$

where $y \prec x$ means that solution y dominates solution x.

3) RATIO OF NON-DOMINATED SOLUTIONS V_{rd}

The metric V_{rd} was used to compute the ratio of non-dominated solutions in the obtained solution set S^j. A larger value of V_{rd} represents a solution set with a higher probability for the obtained solution to be a non-dominated solution. If V_{rd} is close to one, the obtained solution set is equal to or near the non-dominated solutions set, whereas if V_{rd} is close to zero, each obtained solution will be dominated by one of the solution in the reference solution set. The computational process uses the following formulation:

$$V_{rd} = \frac{V_{np}}{|S^j|}$$

C. COMPARISON RESULTS

The computational results for the Pareto number, Pareto rate, and the Pareto distance among the five compared algorithms are reported in Tables 7, 8, and 9, respectively.

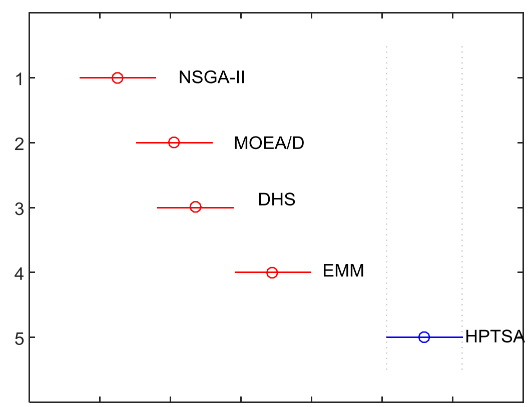
Table 7 shows the comparison results of the Pareto number. It can be concluded from the computational results for the Pareto number that: (1) for the 20 realistic FJSP problems, the proposed HPTSA obtained 19 optimal solutions out of 20 instances, whereas the second-best algorithm EMM only obtained five optimal instances; (2) it can be observed from the last line that on average, the proposed algorithm performed the best. Further, we also take a multifactor analysis of variance (ANOVA) to test whether the differences are significant. Fig. 4 shows that, considering the Pareto number comparison results, the five pairs of compared algorithms show significant differences with each other.

Table 8 shows the results of the Pareto rate, where there are 12 columns. The 1th column gives the tested instance name. Then, the next column tells the problem scale represented by the number of jobs in the system. Then, the next

TABLE 8. Comparisons of the Pareto rate for 20 realistic problems.

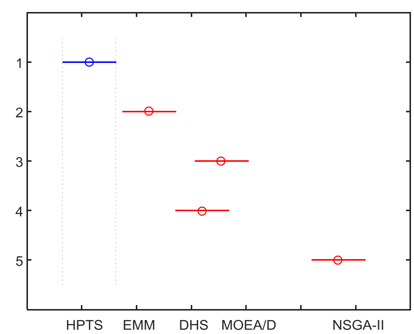
Problem	Scale	Pareto rate values					*dev*				
		NSGA-II	MOEA/D	DHS	EMM	HPTSA	NSGA-II	MOEA/D	DHS	EMM	HPTSA
Case1	50-jobs	0.12	0.15	0.32	0.36	**0.45**	0.0	0.3	1.7	2.0	**2.8**
Case2	50-jobs	0.13	0.20	0.36	0.21	**0.61**	0.0	0.5	1.7	0.6	**3.6**
Case3	50-jobs	0.11	0.09	0.31	0.18	**0.56**	0.2	0.0	2.4	1.0	**5.2**
Case4	50-jobs	0.19	0.21	0.15	0.25	**0.31**	0.2	0.4	0.0	0.7	**1.1**
Case5	50-jobs	0.27	0.28	0.23	**0.35**	0.32	0.2	0.2	0.0	**0.5**	0.4
Case6	100-jobs	0.17	0.19	0.22	0.19	**0.52**	0.0	0.1	0.3	0.1	**2.1**
Case7	100-jobs	0.13	0.23	0.23	0.31	**0.61**	0.0	0.7	0.7	1.3	**3.6**
Case8	100-jobs	0.10	0.25	0.18	0.42	**0.72**	0.0	1.5	0.8	3.1	**6.1**
Case9	100-jobs	0.11	0.31	**0.46**	0.42	0.31	0.0	1.8	**3.2**	2.8	1.8
Case10	100-jobs	0.01	0.18	0.31	0.25	**0.90**	0.0	17.0	30.0	24.0	**89.0**
Case11	150-jobs	0.08	0.19	0.56	0.50	**0.82**	0.0	1.4	6.0	5.3	**9.3**
Case12	150-jobs	0.09	0.20	0.60	0.42	**0.71**	0.0	1.2	5.7	3.7	**6.9**
Case13	150-jobs	0.07	0.16	0.35	0.38	**0.62**	0.0	1.3	4.0	4.4	**7.9**
Case14	150-jobs	0.10	0.19	0.39	0.33	**0.72**	0.0	0.9	2.8	2.3	**6.1**
Case15	150-jobs	0.03	0.31	0.41	0.21	**0.65**	0.0	9.3	12.7	6.0	**20.7**
Case16	200-jobs	0.14	0.11	0.32	0.38	**0.53**	0.3	0.0	1.9	2.5	**3.8**
Case17	200-jobs	0.04	0.21	**0.48**	0.28	0.45	0.0	4.3	**11.0**	6.0	10.3
Case18	200-jobs	0.18	0.23	0.30	0.39	**0.46**	0.0	0.3	0.7	1.2	**1.6**
Case19	200-jobs	0.09	0.40	0.58	0.22	**0.60**	0.0	3.4	5.5	1.5	**5.7**
Case20	200-jobs	0.10	0.37	0.33	0.33	**0.49**	0.0	2.7	2.3	2.3	**3.9**
Mean		0.11	0.22	0.35	0.32	**0.57**	0.0	2.4	4.7	3.6	**9.6**

- the best values are in bold



4 groups have means significantly different from HPTSA

FIGURE 4. Means and 95% LSD interval for the Pareto number.



4 groups have means significantly different from HPTSA

FIGURE 5. Means and 95% LSD interval for the Pareto rate.

five columns describe the average Pareto rate obtained by NSGA-II, MOEA/D, DHS, EMM, and HPTSA. The last five columns give the dev values based on the Pareto rate, where dev value was computed as follows: dev $= (f_c - f_{min})/f_{min}$, where f_{min} is the minimum values collected by the five compared algorithms. It can be seen in Table 6 that: (1) for the 20 realistic FJSP problems, the proposed HPTSA obtained 17 optimal solutions out of 20 instances; (2) the last line shows that on average, the proposed algorithm performed the best; and (3) from the *dev* values, we found that HPTSA shows better performance. Fig. 5 illustrates the means and the 95% LSD (Fisher's Least Significant Difference) interval for the average Pareto rate. It can be concluded from Fig. 5 that, considering the Pareto rate comparison results, the five pairs of compared algorithms show significant differences with each other.

Table 9 shows the comparison results of the Pareto distance. It can be concluded from the computational results for the Pareto distance that: (1) for the 20 realistic FJSP problems, the proposed HPTSA obtained 17 optimal solutions out of 20 instances; (2) the last line shows that on average, the proposed algorithm performed the best, following with the EMM algorithm; and (3) from the *dev* values, we found that HPTSA shows better performance. Fig. 6 shows that, considering the Pareto distance comparison results, the

TABLE 9. Comparisons of the Pareto distance for 20 realistic problems.

Problem	Scale	Pareto distance values					*dev*				
		NSGA-II	MOEA/D	DHS	EMM	HPTSA	NSGA-II	MOEA/D	DHS	EMM	HPTSA
Case1	50-jobs	0.51	0.37	0.18	0.23	**0.12**	3.25	2.08	0.50	0.92	**0.00**
Case2	50-jobs	0.43	0.25	0.26	0.31	**0.13**	2.31	0.92	1.00	1.38	**0.00**
Case3	50-jobs	0.37	0.33	0.12	0.13	**0.10**	2.70	2.30	0.20	0.30	**0.00**
Case4	50-jobs	0.55	0.32	0.17	0.15	**0.10**	4.50	2.20	0.70	0.50	**0.00**
Case5	50-jobs	0.37	0.34	0.24	0.25	**0.13**	1.85	1.62	0.85	0.92	**0.00**
Case6	100-jobs	0.42	0.35	0.19	0.18	**0.12**	2.50	1.92	0.58	0.50	**0.00**
Case7	100-jobs	0.51	0.25	0.28	0.28	**0.15**	2.40	0.67	0.87	0.87	**0.00**
Case8	100-jobs	0.32	0.41	0.49	0.15	**0.13**	1.46	2.15	2.77	0.15	**0.00**
Case9	100-jobs	0.28	0.42	0.15	0.13	**0.12**	1.33	2.50	0.25	0.08	**0.00**
Case10	100-jobs	0.33	0.32	0.31	0.12	**0.08**	3.13	3.00	2.88	0.50	**0.00**
Case11	150-jobs	0.37	**0.08**	0.15	0.18	0.12	3.63	**0.00**	0.88	1.25	0.50
Case12	150-jobs	0.42	0.32	0.12	0.09	**0.08**	4.25	3.00	0.50	0.13	**0.00**
Case13	150-jobs	0.25	0.29	0.13	0.12	**0.09**	1.78	2.22	0.44	0.33	**0.00**
Case14	150-jobs	0.37	0.24	0.25	**0.03**	0.05	11.33	7.00	7.33	**0.00**	0.67
Case15	150-jobs	0.28	0.38	0.18	0.03	**0.02**	13.00	18.00	8.00	0.50	**0.00**
Case16	200-jobs	0.29	0.29	0.19	0.21	**0.10**	1.90	1.90	0.90	1.10	**0.00**
Case17	200-jobs	0.22	0.19	**0.03**	0.23	0.08	6.33	5.33	**0.00**	6.67	1.67
Case18	200-jobs	0.34	0.23	0.17	0.12	**0.08**	3.25	1.88	1.13	0.50	**0.00**
Case19	200-jobs	0.41	0.33	0.26	0.05	**0.04**	9.25	7.25	5.50	0.25	**0.00**
Case20	200-jobs	0.43	0.32	0.15	0.02	**0.01**	42.00	31.00	14.00	1.00	**0.00**
Mean		0.37	0.30	0.20	0.15	**0.09**	6.11	4.85	2.46	0.89	**0.14**

- the best values are in bold

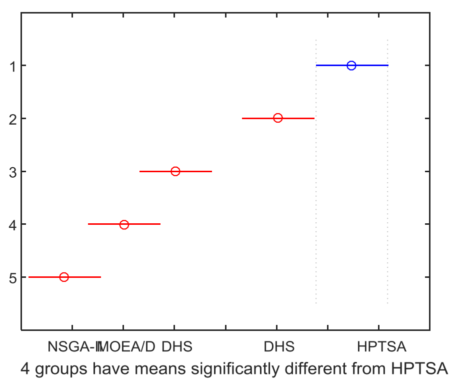


FIGURE 6. Means and 95% LSD interval for the Pareto distance.

five pairs of compared algorithms show significant differences with each other.

D. COMPARISON ANALYSIS

From the experimental comparisons, it can be concluded that the proposed algorithm is efficient and competitive to other efficient multi-objective optimization algorithms. The main advantages of the proposed algorithm are as follows: (1) considering both the problem characteristics and the objective features, in the proposed algorithm, several initialization approaches are utilized to produce initial solutions with high performance; (2) five types of neighborhood structures that consider both problem structures are developed to enhance the exploitation and exploration capabilities; (3) to transfer the four objectives to a three-objective optimization problem, which can use the efficient performance of Pareto-based optimization methods; and (4) a well-designed backward method is embedded to optimize the E/T criteria without affecting the other objectives.

VI. CONCLUSIONS

In this study, we considered four objectives simultaneously, minimization of the makespan, the maximal workload, the total workload, and the earliness/tardiness (E/T) criteria, for solving the distributed multi-objective FJSP. The main contributions are as follows: (1) detailed reviews of the meta-heuristics, task assignment problems, multi-objective optimization algorithms, and FJSP problems were described, which gave a brief and clear display of the development of algorithms and problems related to this study; (2) several approaches considering both the problem characteristics and the objective features were used to generate the initial population; (3) five types of neighborhood structures which consider both problem structures and the balance of global search and local search abilities were developed; and (4) a well-designed backward method was proposed to optimize the E/T criteria.

In our future works, we will consider following issues: (1) applying the proposed algorithm to solve other complex and realistic production problems, such as steelmaking casting problem with more realistic constraints, and distributed flexible job shop scheduling problem; (2) improving the capabilities of handling more objectives, such as the energy consumptions, and the multi-modal processing features; (3) investigating the self-adaptive searching procedure to balance the global and local abilities of the proposed algorithm; (4) to consider the assemble stage and the travel time between each factory; and (5) to verify the efficiency of the proposed algorithm in a theoretical way.

REFERENCES

[1] P. Zhang and M. Zhou, "Dynamic cloud task scheduling based on a two-stage strategy," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 772–783, Apr. 2018.
[2] H. Yuan, J. Bi, W. Tan, and B. Li, "Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 14, no. 1, pp. 337–348, Jan. 2017.
[3] Y. Xiong, S. Huang, M. Wu, J. She, and K. Jiang, "A Johnson's-rule-based genetic algorithm for two-stage-task scheduling problem in data-centers of cloud computing," *IEEE Trans. Cloud Comput.*, to be published, doi: 10.1109/TCC.2017.2693187.
[4] J. Wang, B. Gong, H. Liu, and S. Li, "Multidisciplinary approaches to artificial swarm intelligence for heterogeneous computing and cloud scheduling," *Appl. Intell.*, vol. 43, no. 3, pp. 662–675, 2015.
[5] J. Li, Q. Pan, and S. Xie, "An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems," *Appl. Math. Comput.*, vol. 218, no. 18, pp. 9353–9371, 2012.
[6] Q.-K. Pan and R. Ruiz, "Local search methods for the flowshop scheduling problem with flowtime minimization," *Eur. J. Oper. Res.*, vol. 222, no. 1, pp. 31–43, 2012.
[7] H.-Y. Sang, P.-Y. Duan, and J. Li, "An effective invasive weed optimization algorithm for scheduling semiconductor final testing problem," *Swarm Evol. Comput.*, vol. 38, pp. 42–53, May 2018.
[8] B. Marzouki, O. B. Driss, and K. Ghédira, "Decentralized tabu searches in multi agent system for distributed and flexible job shop scheduling problem," in *Proc. IEEE/ACS 14th Int. Conf. Comput. Syst. Appl. (AICCSA)*, Oct./Nov. 2017, pp. 1019–1026.
[9] Y.-H. Jia *et al.*, "A dynamic logistic dispatching system with set-based particle swarm optimization," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 48, no. 9, pp. 1607–1621, Sep. 2017, doi: 10.1109/TSMC.2017.2682264.
[10] B. Liu, L. Wang, Y. Liu, B. Qian, and Y.-H. Jin, "An effective hybrid particle swarm optimization for batch scheduling of polypropylene processes," *Comput. Chem. Eng.*, vol. 34, no. 4, pp. 518–528, 2010.
[11] Z. Zheng and J. Li, "Optimal chiller loading by improved invasive weed optimization algorithm for reducing energy consumption," *Energy Buildings*, vol. 161, pp. 80–88, Feb. 2018.
[12] H.-Y. Sang, Q.-K. Pan, P.-Y. Duan, and J.-Q. Li, "An effective discrete invasive weed optimization algorithm for lot-streaming flowshop scheduling problems," *J. Intell. Manuf.*, vol. 29, no. 6, pp. 1337–1349, 2015, doi: 10.1007/s10845-015-1182-x.
[13] Y. Liu, Y.-C. Jiao, Y.-M. Zhang, and Y.-Y. Tan, "Synthesis of phase-only reconfigurable linear arrays using multiobjective invasive weed optimization based on decomposition," *Int. J. Antennas Propag.*, vol. 2014, Oct. 2014, Art. no. 630529, doi: 10.1155/2014/630529.
[14] J.-Q. Li, Q.-K. Pan, and K. Mao, "A discrete teaching-learning-based optimisation algorithm for realistic flowshop rescheduling problems," *Eng. Appl. Artif. Intell.*, vol. 37, no. 1, pp. 279–292, 2015.
[15] P.-Y. Duan, J.-Q. Li, Y. Wang, H.-Y. Sang, and B.-X. Jia, "Solving chiller loading optimization problems using an improved teaching-learning-based optimization algorithm," *Optim. Control Appl. Methods*, vol. 39, no. 1, pp. 65–77, 2018.
[16] K. Yu *et al.*, "Multiobjective optimization of ethylene cracking furnace system using self-adaptive multiobjective teaching-learning-based optimization," *Energy*, vol. 148, no. 4, pp. 469–481, 2018.
[17] J.-Q. Li, Q.-K. Pan, and K. Mao, "A hybrid fruit fly optimization algorithm for the realistic hybrid flowshop rescheduling problem in steelmaking systems," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 932–949, Apr. 2016.
[18] L. Wang, X. L. Zheng, and S. Y. Wang, "A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem," *Knowl.-Based Syst.*, vol. 48, no. 2, pp. 17–23, 2013.
[19] Q.-K. Pan, H.-Y. Sang, J.-H. Duan, and L. Gao, "An improved fruit fly optimization algorithm for continuous function optimization problems," *Knowl.-Based Syst.*, vol. 62, pp. 69–83, May 2014.
[20] J.-Q. Li, Q.-K. Pan, and M. F. Tasgetiren, "A discrete artificial bee colony algorithm for the multi-objective flexible job-shop scheduling problem with maintenance activities," *Appl. Math. Model.*, vol. 38, no. 3, pp. 1111–1132, 2014.
[21] P. Zhang, H. Liu, and Y. Ding, "Dynamic bee colony algorithm based on multi-species co-evolution," *Appl. Intell.*, vol. 40, no. 3, pp. 427–440, 2014.
[22] C. Hu, H. Liu, and P. Zhang, "Cooperative co-evolutionary artificial bee colony algorithm based on hierarchical communication model," *Chin. J. Electron.*, vol. 25, no. 3, pp. 570–576, 2016.
[23] J.-Q. Li, Q.-K. Pan, and P.-Y. Duan, "An improved artificial bee colony algorithm for solving hybrid flexible flowshop with dynamic operation skipping," *IEEE Trans. Cybern.*, vol. 46, no. 6, pp. 1311–1324, Jun. 2016.
[24] B. Marzouki, O. B. Driss, and K. Ghédira, "Multi-agent model based on combination of chemical reaction optimisation metaheuristic with Tabu search for flexible job shop scheduling problem," *Int. J. Intell. Eng. Inform.*, vol. 6, nos. 3–4, pp. 242–265, 2018.
[25] J. Q. Li, Q. Pan, P. Duan, K. Gao, and H. Sang, "Solving multi-area environmental/economic dispatch by Pareto-based chemical-reaction optimization algorithm," *IEEE/CAA J. Autom. Sinica*, to be published, doi: 10.1109/JAS.2017.7510454.
[26] Z. Ren, Y. Liang, L. Wang, A. Zhang, B. Pang, and B. Li, "Anisotropic adaptive variance scaling for Gaussian estimation of distribution algorithm," *Knowl.-Based Syst.*, vol. 146, pp. 142–151, Apr. 2018.
[27] M. Jiang, L. Qiu, Z. Huang, and G. G. Yen, "Dynamic multi-objective estimation of distribution algorithm based on domain adaptation and nonparametric estimation," *Inf. Sci.*, vol. 435, pp. 203–223, Apr. 2018, doi: 10.1016/j.ins.2017.12.058.
[28] A. Sioud and C. Gagné, "Enhanced migrating birds optimization algorithm for the permutation flow shop problem with sequence dependent setup times," *Eur. J. Oper. Res.*, vol. 264, no. 1, pp. 66–73, 2018.
[29] Z.-X. Zheng, J.-Q. Li, and P.-Y. Duan, "Optimal chiller loading by improved artificial fish swarm algorithm for energy saving," *Math. Comput. Simul.*, vol. 155, pp. 227–243, Jan. 2019, doi: 10.1016/j.matcom.2018.04.013.
[30] W. Jia, D. Zhao, Y. Zheng, and S. Hou, "A novel optimized GA–Elman neural network algorithm," *Neural Comput. Appl.*, to be published, doi: 10.1007/s00521-017-3076-7.
[31] X.-W. Zheng, D.-J. Lu, X.-G. Wang, and H. Liu, "A cooperative coevolutionary biogeography-based optimizer," *Appl. Intell.*, vol. 43, no. 1, pp. 95–111, 2015.
[32] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Trans. Cybern.*, vol. 43, no. 6, pp. 1656–1671, Dec. 2013.
[33] H. A. Taboada, J. F. Espiritu, and D. W. Coit, "MOMS-GA: A multi-objective multi-state genetic algorithm for system reliability optimization design problems," *IEEE Trans. Rel.*, vol. 57, no. 1, pp. 182–191, Mar. 2008.
[34] R. Battiti and A. Passerini, "Brain–computer evolutionary multiobjective optimization: A genetic algorithm adapting to the decision maker," *IEEE Trans. Evol. Comput.*, vol. 14, no. 5, pp. 671–687, Oct. 2013.
[35] S. Li, N. Wang, T. Jia, Z. He, and H. Liang, "Multiobjective optimization for multiperiod reverse logistics network design," *IEEE Trans. Eng. Manag.*, vol. 63, no. 2, pp. 223–236, May 2016.
[36] J. Yi, D. Huang, H. He, T. Li, and S. Fu, "Multi-objective bacterial foraging optimization algorithm based on parallel cell entropy for aluminum electrolysis production process," *IEEE Trans. Ind. Electron.*, vol. 63, no. 4, pp. 2488–2500, Apr. 2016.
[37] J.-Q. Li, H.-Y. Sang, Y.-Y. Han, C.-G. Wang, and K.-Z. Gao, "Efficient multi-objective optimization algorithm for hybrid flow shop scheduling problems with setup energy consumptions," *J. Cleaner Prod.*, vol. 181, pp. 584–598, Apr. 2018.
[38] K.-Z. Gao, P. N. Suganthan, Q.-K. Pan, T. J. Chua, T. X. Cai, and C.-S. Chong, "Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling," *Inf. Sci.*, vol. 289, pp. 76–90, Dec. 2014.
[39] L. Li and J. Z. Huo, "Multi-objective flexible job-shop scheduling problem in steel tubes production," *Syst. Eng.-Theory Pract.*, vol. 29, no. 8, pp. 117–126, 2009.

[40] Y. Yuan, H. Xu, B. Wang, B. Zhang, and X. Yao, "Balancing convergence and diversity in decomposition-based many-objective optimizers," *IEEE Trans. Evol. Comput.*, vol. 20, no. 2, pp. 180–198, Apr. 2016.
[41] H. Wang, L. Jiao, and X. Yao, "Two_Arch2: An improved two-archive algorithm for many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 19, no. 4, pp. 524–541, Aug. 2015.
[42] X. Cai, Z. Yang, Z. Fan, and Q. Zhang, "Decomposition-based-sorting and angle-based-selection for evolutionary multiobjective and many-objective optimization," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2824–2837, Sep. 2017.
[43] R. C. P. Silva, M. Li, T. Rahman, and D. A. Lowther, "Surrogate-based MOEA/D for electric motor design with scarce function evaluations," *IEEE Trans. Magn.*, vol. 53, no. 6, Jun. 2017, Art. no. 7400704.
[44] Z. Wang, Q. Zhang, A. Zhou, M. Gong, and L. Jiao, "Adaptive replacement strategies for MOEA/D," *IEEE Trans. Cybern.*, vol. 46, no. 2, pp. 474–486, Feb. 2016.
[45] L. Wang, Q. Zhang, A. Zhou, M. Gong, and L. Jiao, "Constrained subproblems in a decomposition-based multiobjective evolutionary algorithm," *IEEE Trans. Evol. Comput.*, vol. 20, no. 3, pp. 475–480, Jun. 2016.
[46] J. Bai and H. Liu, "Multi-objective artificial bee algorithm based on decomposition by PBI method," *Appl. Intell.*, vol. 45, no. 4, pp. 976–991, 2016.
[47] J. Li, P. Duan, H. Sang, S. Wang, Z. Liu, and P. Duan, "An efficient optimization algorithm for resource-constrained steelmaking scheduling problems," *IEEE Access*, vol. 6, pp. 33883–33894, 2018.
[48] H. Liu, P. Zhang, and B. Hu, "A novel approach to task assignment in a cooperative multi-agent design system," *Appl. Intell.*, vol. 43, no. 1, pp. 162–175, 2015.
[49] Z. Zhang and H. Liu, "Social recommendation model combining trust propagation and sequential behaviors," *Appl. Intell.*, vol. 43, no. 3, pp. 695–706, 2015.
[50] J. Wang, B. Gong, H. Liu, and S. Li, "Model and algorithm for heterogeneous scheduling integrated with energy-efficiency awareness," *Trans. Inst. Meas. Control*, vol. 38, no. 4, pp. 452–462, 2015.
[51] J. Wang, B. Gong, H. Liu, S. H. Li, and J. Yi, "Heterogeneous computing and grid scheduling with parallel biologically inspired hybrid heuristics," *Trans. Inst. Meas. Control*, vol. 36, no. 6, pp. 805–814, 2014.
[52] T. Zhang, W. Chen, Z. Han, and Z. Cao, "A cross-layer perspective on energy-harvesting-aided green communications over fading channels," *IEEE Trans. Veh. Technol.*, vol. 64, no. 4, pp. 1519–1534, Apr. 2015.
[53] T. Zhang, W. Chen, and H. Wang, "Energy harvesting aided multiuser transmission in spectrum sharing networks," *IEEE Access*, vol. 4, pp. 4038–4045, 2016.
[54] H. Zhang, L. Cao, and S. Gao, "A locality correlation preserving support vector machine," *Pattern Recognit.*, vol. 47, no. 9, pp. 3168–3178, Sep. 2014.
[55] H. Zhang and J. Lu, "Creating ensembles of classifiers via fuzzy clustering and deflection," *Fuzzy Sets Syst.*, vol. 161, no. 13, pp. 1790–1802, 2010.
[56] Z. Zeng, Z. Li, H. Zhang, K. Zhan, Y. Yang, and D. Cheng, "Two-stream multirate recurrent neural network for video-based pedestrian reidentification," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3179–3186, Jul. 2018, doi: 10.1109/TII.2017.2767557.
[57] H. Liu, Y. Sun, and Y. Li, "Modeling and path generation approaches for crowd simulation based on computational intelligence," *Chin. J. Electron.*, vol. 21, no. 4, pp. 636–641, 2012.
[58] K. Ji, Z. Chen, R. Sun, K. Ma, Z. Yuan, and G. Xu, "GIST: A generative model with individual and subgroup-based topics for group recommendation," *Expert Syst. Appl.*, vol. 94, pp. 81–93, Mar. 2018.
[59] X. Ren *et al.*, "Drusen segmentation from retinal images via supervised feature learning," *IEEE Access*, vol. 6, pp. 2952–2961, 2018.
[60] Y. Wang, H. Zhang, and F. Yang, "A weighted sparse neighbourhood-preserving projections for face recognition," *IETE J. Res.*, vol. 63, no. 3, pp. 358–367, 2017.
[61] S. Qiao, B. Yan, and J. Li, "Ensemble learning for protein multiplex subcellular localization prediction based on weighted KNN with different features," *Appl. Intell.*, vol. 48, no. 7, pp. 1813–1824, 2018.
[62] Z. Xiaowei, L. Hong, and S. Xiaohong, "Object tracking with an evolutionary particle filter based on self-adaptive multi-features fusion," *Int. J. Adv. Robot. Syst.*, vol. 10, no. 1, p. 61, 2013.
[63] X. Yang, X. D. Li, and J. Cao, "Robust finite-time stability of singular nonlinear systems with interval time-varying delay," *J. Franklin Inst.*, vol. 355, no. 3, pp. 1241–1258, 2018.
[64] X. Li, X. Zhang, and S. Song, "Effect of delayed impulses on input-to-state stability of nonlinear systems," *Automatica*, vol. 76, pp. 378–382, Feb. 2017.
[65] X. Song and H. P. Ju, "Linear minimum mean-square estimation for discrete-time measurement-delay systems with multiplicative noise and Markov jump," *IET Control Theory Appl.*, vol. 10, no. 10, pp. 1161–1169, 2016.
[66] X. Li and J. Wu, "Sufficient stability conditions of nonlinear differential systems under impulsive control with state-dependent delay," *IEEE Trans. Autom. Control*, vol. 63, no. 1, pp. 306–311, Jan. 2018.
[67] X. Song and J. H. Park, "Linear optimal estimation for discrete-time measurement delay systems with multichannel multiplicative noise," *IEEE Trans. Circuis Syst. II, Exp. Brief*, vol. 64, no. 2, pp. 156–160, Feb. 2017.
[68] X. Li and X. Fu, "Effect of leakage time-varying delay on stability of nonlinear differential systems," *J. Franklin Inst.*, vol. 350, no. 6, pp. 1335–1344, 2013.
[69] X. Tan, J. Cao, A. Alsaedi, and X. D. Li, "Leader-following mean square consensus of stochastic multi-agent systems with input delay via event-triggered control," *IET Control Theory Appl.*, vol. 12, no. 2, pp. 299–309, 2017.
[70] X. Tan, J. Cao, and X. D. Li, "Consensus of leader-following multiagent systems: A distributed event-triggered impulsive control strategy," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2017.2786474.
[71] C. Yang *et al.*, "Consensus for non-linear multi-agent systems modelled by PDEs based on spatial boundary communication," *IET Control Theory Appl.*, vol. 11, no. 17, pp. 3196–3200, 2017.
[72] Z. Wang, H. Zhang, X. Song, and H. Zhang, "Consensus problems for discrete-time agents with communication delay," *Int. J. Control, Automat. Syst.*, vol. 15, no. 4, pp. 1515–1523, 2017.
[73] R. Ruiz, J. A. Vázquez-Rodríguez, "The hybrid flow shop scheduling problem," *Eur. J. Oper. Res.*, vol. 205, no. 1, pp. 1–18, 2010.
[74] B. Zhang, Q.-K. Pan, L. Gao, X.-L. Zhang, H.-Y. Sang, and J. Q. Li, "An effective modified migrating birds optimization for hybrid flowshop scheduling problem with lot streaming," *Appl. Soft Comput.*, vol. 52, pp. 14–27, Mar. 2017.
[75] Q.-K. Pan and Y. Dong, "An improved migrating birds optimisation for a hybrid flowshop scheduling with total flowtime minimisation," *Inf. Sci.*, vol. 277, pp. 643–655, Sep. 2014.
[76] 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, Feb. 2018.
[77] C. Chamnanlor, K. Sethanan, M. Gen, and C.-F. Chien, "Embedding ant system in genetic algorithm for re-entrant hybrid flow shop scheduling problems with time window constraints," *J. Intell. Manuf.*, vol. 28, no. 8, pp. 1915–1931, 2017.
[78] Q.-K. Pan, L. Gao, X.-Y. Li, and K. Z. Gao, "Effective metaheuristics for scheduling a hybrid flowshop with sequence-dependent setup times," *Appl. Math. Comput.*, vol. 303, pp. 89–112, Jun. 2017.
[79] J.-Q. Li *et al.*, "A hybrid artificial bee colony for optimizing a reverse logistics network system," *Soft Comput.*, vol. 21, no. 20, pp. 6001–6018, 2017.
[80] D. Lei and Y. Zheng, "Hybrid flow shop scheduling with assembly operations and key objectives: A novel neighborhood search," *Appl. Soft Comput.*, vol. 61, pp. 122–128, Dec. 2017.
[81] Q.-K. Pan, R. Ruiz, and P. Alfaro-Fernández, "Iterated search methods for earliness and tardiness minimization in hybrid flowshops with due windows," *Comput. Oper. Res.*, vol. 80, pp. 50–60, Apr. 2017.
[82] C. Lu, X. Li, L. Gao, W. Liao, and J. Yi, "An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times," *Comput. Ind. Eng.*, vol. 104, pp. 156–174, Feb. 2017.
[83] Y. Zhang, J. Wang, and Y. Liu, "Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact," *J. Cleaner Prod.*, vol. 167, pp. 665–679, Nov. 2017.
[84] X. Gong, Q. Deng, G. Gong, W. Liu, and Q. Ren, "A memetic algorithm for multi-objective flexible job-shop problem with worker flexibility," *Int. J. Prod. Res.*, vol. 56, no. 7, pp. 2506–2522, 2018.
[85] Q. Liu, M. Zhan, F. O. Chekem, X. Shao, B. Ying, and J. W. Sutherland, "A hybrid fruit fly algorithm for solving flexible job-shop scheduling to reduce manufacturing carbon footprint," *J. Cleaner Prod.*, vol. 168, pp. 668–678, Dec. 2017.

[86] H. Mokhtari and A. Hasani, "An energy-efficient multi-objective optimization for flexible job-shop scheduling problem," *Comput. Chem. Eng.*, vol. 104, pp. 339–352, Sep. 2015.

[87] A. P. Rifai, H.-T. Nguyen, and S. Z. M. Dawal, "Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling," *Appl. Soft. Comput.*, vol. 40, pp. 42–57, Mar. 2016.

[88] S.-W. Lin and K.-C. Ying, "Minimizing makespan for solving the distributed no-wait flowshop scheduling problem," *Comput. Ind. Eng.*, vol. 99, pp. 202–209, Sep. 2016.

[89] H. Bargaoui, O. B. Driss, and K. Ghédira, "A novel chemical reaction optimization for the distributed permutation flowshop scheduling problem with makespan criterion," *Comput. Ind. Eng.*, vol. 111, pp. 239–250, Sep. 2017.

[90] K.-C. Ying, S.-W. Lin, C.-Y. Cheng, and C.-D. He, "Iterated reference greedy algorithm for solving distributed no-idle permutation flowshop scheduling problems," *Comput. Ind. Eng.*, vol. 110, pp. 413–423, Aug. 2017.

[91] M. Komaki and B. Malakooti, "General variable neighborhood search algorithm to minimize makespan of the distributed no-wait flow shop scheduling problem," *Prod. Eng.*, vol. 11, no. 3, pp. 315–329, 2017.

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

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

[94] R. Zhang, P.-C. Chang, S. Song, and C. Wu, "A multi-objective artificial bee colony algorithm for parallel batch-processing machine scheduling in fabric dyeing processes," *Knowl.-Based Syst*, vol. 116, pp. 114–129, Jan. 2017.

[95] L. Gao and Q.-K. Pan, "A shuffled multi-swarm micro-migrating birds optimizer for a multi-resource-constrained flexible job shop scheduling problem," *Inf. Sci.*, vol. 372, pp. 655–676, Dec. 2016.

[96] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[97] 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, Dec. 2007.

[98] K. Z. Gao, P. N. Suganthan, Q. K. Pan, T. J. Chua, T. X. Cai, and C. S. Chong, "Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives," *J. Intell. Manuf.*, vol. 27, no. 2, pp. 363–374, 2016.

[99] Q.-K. Pan, L. Wang, and B. Qian, "A novel differential evolution algorithm for bi-criteria no-wait flow shop scheduling problems," *Comput. Oper. Res.*, vol. 36, no. 8, pp. 2498–2511, 2009.

JUN-QIANG LI (M'18) was with the School of Information Science and Engineering, Shandong Normal University; with the School of Economics and Management, Nanjing Forestry University, Nanjing; and also with the China Key Laboratory of Computer Network and Information Integration, Ministry of Education, Southeast University. He has authored over 10 refereed papers, including applied intelligence, and computers and operations research. His current research interests include swarm intelligent optimization algorithms and scheduling problems. He was a member of ACM.

PEIYONG DUAN received the B.Sc. degree from the Shandong University of Technology (merged with Shandong University in 2000), China, in 1996, and the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 1999. From 1999 to 2014, he was with the School of Information and Electrical Engineering, Shandong Jianzhu University, where he was appointed as an Associate Professor and a Full Professor in 1999 and 2002, respectively. Since 2017, he has been with the School of Information and Engineering, Shandong Normal University, China. He has authored over 60 refereed papers. His current research interests include discrete optimization and scheduling.

JINDE CAO (F'15) has been a Professor with the School of Mathematics, Southeast University, China, since 2000, where he has been the Dean since 2014. From 2014 to 2017, he was appointed as a Highly Cited Researcher simultaneously in three disciplines: Engineering, Computer Science, and Mathematics by Thomson Reuters/Clarivate Analytics. He has authored over 200 refereed papers. His current research interests include complex network and complex system. He was appointed as a member of the European Academy of Sciences and Arts in 2018, a fellow of the Pakistan Academy of Sciences in 2016, and a member of Academia European in 2016. He contributed to the analysis of neural networks.

XIAO-PING LIN received the B.Sc. degree from Qufu Normal University in 2017. Since 2017, she has been a Graduate Student with Shandong Normal University. Her current research interests include discrete optimization and scheduling

YU-YAN HAN received the Ph.D. degree from the School of Control Science and Engineering, China Mining University, Xuzhou, China, in 2016. Since 2016, she has been a Lecture with the School of Computer, Liaocheng University. She has authored over 10 refereed papers. Her current research interests include discrete optimization and scheduling.