Hindawi
Mathematical Problems in Engineering
Volume 2017, Article ID 2857564, 20 pages
https://doi.org/10.1155/2017/2857564

Research Article

A Novel Memetic Algorithm Based on Decomposition for Multiobjective Flexible Job Shop Scheduling Problem

Chun Wang, Zhicheng Ji, and Yan Wang

Engineering Research Center of IoT Technology Applications, Ministry of Education, Jiangnan University, Wuxi 214122, China

Correspondence should be addressed to Yan Wang; wangyan88@jiangnan.edu.cn

Received 12 May 2017; Revised 1 November 2017; Accepted 9 November 2017; Published 29 November 2017

Academic Editor: Josefa Mula

Copyright © 2017 Chun Wang et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

A novel multiobjective memetic algorithm based on decomposition (MOMAD) is proposed to solve multiobjective flexible job shop scheduling problem (MOFJSP), which simultaneously minimizes makespan, total workload, and critical workload. Firstly, a population is initialized by employing an integration of different machine assignment and operation sequencing strategies. Secondly, multiobjective memetic algorithm based on decomposition is presented by introducing a local search to MOEA/D. The Tchebycheff approach of MOEA/D converts the three-objective optimization problem to several single-objective optimization subproblems, and the weight vectors are grouped by K-means clustering. Some good individuals corresponding to different weight vectors are selected by the tournament mechanism of a local search. In the experiments, the influence of three different aggregation functions is first studied. Moreover, the effect of the proposed local search is investigated. Finally, MOMAD is compared with eight state-of-the-art algorithms on a series of well-known benchmark instances and the experimental results show that the proposed algorithm outperforms or at least has comparative performance to the other algorithms.

1. Introduction

The job shop scheduling problem (JSP) is one of the most important and difficult problems in the field of manufacturing which processes a set of jobs on a set of machines. Each job consists of a sequence of successive operations, and each operation is allowed to process on a unique machine. Different from JSP which one operation is merely allowed to process on a specific machine, the flexible job shop scheduling problem (FJSP) permits one operation processed by any machine from its available machine set. Since FJSP needs to assign operations to their suited machine as well as sequence those operations assigned on the same machine, it is a complex NP-hard optimization problem [1].

The existing literatures [2–5] about solving single-objective FJSP (SOFJSP) over the past decades mainly concentrated on minimizing one specific objective such as makespan. However, in practical manufacturing process, single-objective optimization cannot fully satisfy the production requirements since many optimized objectives are usually in conflict with each other. In recent years, multiobjective flexible job shop scheduling problem (MOFJSP) has received much attention, and, until now, many algorithms have been developed to solve this kind of problem. These methods can be classified into two groups: one is a priori approach and the other is Pareto approach.

Multiple objectives are usually linearly combined into a single one by weighted sum approach in the a priori method, which can be illustrated as $F = \sum_{i=1}^{M} \lambda_i f_i$, where $\sum_{i=1}^{M} \lambda_i = 1$, $0 \leq \lambda_i \leq 1$. However, we can get only one or several Pareto solutions by using this approach, which may not well reflect the tradeoffs among different objectives, and it would be difficult to assign an appropriate weight for each problem. Even more important, the performance of the algorithm deteriorates when solving the problems contains nonconcave Pareto front (PF). The Pareto approach mainly focuses on searching the Pareto set (PS) of optimization problems by comparing two solutions based on Pareto dominance relation [6]. A solution $\mathbf{x}$ is said to dominate solution $\mathbf{y}$ iff $\mathbf{x}$ is not worse than $\mathbf{y}$ in all objectives and there exists at least one objective in which $\mathbf{x}$ is better than $\mathbf{y}$. $\mathbf{x}^*$ is called a Pareto optimal solution iff there is no solution $\mathbf{x} \in \Omega$ that dominates

$\mathbf{x}^*$. All the Pareto optimal solutions constitute the PS, and PF is the mapped vector of PS in the objective space. Since Pareto approach can achieve a set of Pareto solutions rather than a specific one, it has received much more attention than a priori approach and is recognized to be more suitable to solve MOFJSP.

Because the three objectives, makespan, total workload, and critical workload, are conflicted with each other, it is better to handle this model with knowledge about their PF. Multiobjective evolutionary algorithm (MOEA) is a kind of mature global optimization method with high robustness and wide applicability. Due to the fact that MOEAs have low requirements on the optimization problem itself and high ability to obtain multiple Pareto solutions during each run, they are suitable for solving multiobjective optimization problems (MOPs). The multiobjective evolutionary algorithm based on decomposition (MOEA/D) that integrates mathematical programming with evolutionary algorithm (EA) can obtain a set of Pareto solutions by aggregating multiple objectives into different single-objectives with many predefined weight vectors [7]. MOEA/D has shown great superiority on continuous optimization problems [8–12]; thus it is necessary to investigate its performance on multiobjective combinatorial optimization problems (MO-COPs) such as MOFJSP. To the best of our knowledge, in the literature reported, although MOEA/D has been applied to different kinds of multiobjective scheduling problems such as multiobjective flow shop scheduling problem (MOFSP) [13], multiobjective permutation flow shop scheduling problem (MOPFSP) [14], multiobjective stochastic flexible job shop scheduling problem (MOSFJSP) [15], and multiobjective job shop scheduling problem (MOJSP) [16], there is seldom corresponding application on MOFJSP.

The primary aim of this paper is to solve MOFJSP in a decomposition manner by proposing a multiobjective memetic algorithm based on decomposition (MOMAD) hybridizing MOEA/D with local search. With the purpose of making the proposed algorithm more applicable, four aspects are studied: (1) integration of different machine assignment and operation sequencing strategies are presented to construct the initial population; (2) objective normalization is incorporated into Tchebycheff approach to convert an MOP into a number of single-objective optimization subproblems; (3) all weight vectors are divided into a few groups based on K-means clustering: some superior individuals corresponding to different weight vector groups are selected by using a selection mechanism; (4) local search based on moving critical operations is applied on selected individuals. To evaluate the effectiveness of the proposed algorithm, some benchmark instances are tested with three purposes: (1) investigating the effects of different aggregation functions and validating the effectiveness of local search; (2) analyzing the influence of the key parameters on the performance of the algorithm; (3) comparing the performance of MOMAD with other state-of-the-art algorithms for solving MOFJSP.

The rest of the paper is organized as follows. Next section presents a short overview of the existing related work. In Section 3, the background knowledge of MOFJSP is introduced. Section 4 introduces the framework of MOMAD. The implementation details of the proposed MOMAD including genetic global search and problem specific local search are described in Section 5. Afterwards, experimental studies are provided in Section 6. Finally, Section 7 concludes this paper and outlines some avenues for future research.

2. Related Work

As mentioned above, there are two main methods to solve MOFJSP: a priori approach and Pareto approach. As for a priori approach, Xia and Wu [17] discussed a hybrid algorithm, where particle swarm optimization (PSO) and simulated annealing (SA) were employed in global search and local search, respectively. A bottleneck shifting-based local search was incorporated into genetic global search by Gao et al. [18]. Zhang et al. [19] introduced an effective hybrid PSO algorithm combined with tabu search (TS). Xing et al. [20] used ten different weight vectors to collect effective solution sets. A hybrid TS (HTS) algorithm was structured by combining adaptive rules with two neighborhoods. In this algorithm, three weight coefficients λ_1, λ_2, and λ_3 with different settings were given to test different problems [21]. An effective estimation of distribution algorithm was proposed by Wang et al. [22], in which the new individuals were generated by sampling a probability model.

Contrary to the a priori approach, a PS can be obtained by using Pareto approach, and the tradeoffs among different objectives can be presented. The integration of fuzzy logic and EA was proposed by Kacem et al. [23]. A guide local search was incorporated into EA to enhance the convergence [24]. With the aim of keeping population diversity, immune and entropy principle were adopted in multiobjective genetic algorithm (MOGA) [25]. Two memetic algorithms (MAs) were, respectively, proposed, both of which integrate non-dominated sorting genetic algorithm II (NSGA-II) [6] with effective local search techniques [26, 27]. Several effective neighborhood approaches were used in variable neighborhood search to enhance the convergence ability in a hybrid Pareto-based local search (PLS) [28]. Chiang and Lin [29] proposed a simple and effective evolutionary algorithm which only requires two parameters. Both the neighborhoods of machine assignment and operation sequence are considered in Xiong et al. [30]. An effective Pareto-based EDA was proposed by Wang et al. [31]. A novel path-relinking multiobjective TS algorithm was proposed in [32], in which a routing solution is identified by problem-specific neighborhood search and is then further refined by the TS with back-jump tracking for a sequencing decision. In addition to the successful use of EA, several swarm intelligence algorithms have also been widely used for global search. PSOs were used as global search algorithms in [33–36]. Besides, shuffled frog leaping [37] and artificial bee colony [38] were integrated with local search in related hybrid algorithms.

Besides the successful using in many scheduling problems, MOEA/Ds have also been widely dedicated to other MO-COPs. A novel NBI-style Tchebycheff approach was used in MOEA/D to solve portfolio management MOP with disparately scaled objectives [39]. Mei et al. [40] developed an effective MA by combining MOEA/D with extended neighborhood search to solve capacitated arc routing problem. Hill climbing, SA, and evolutionary gradient search were, respectively, embedded into EDA for solving multiple traveling salesmen problem (MTSP) in a decomposition manner [41]. A hybrid MOEA was established by combining ant colony optimization with MOEA/D [42], and then it was adopted to solve multiobjective 0-1 knapsack problem (MOKP) and MTSP, respectively. Then, aiming at the same two problems, Ke and Zhang proposed a hybridization of decomposition and PLS [43].

As mentioned before, MOEA/D is a kind of popular MOEA which is very suitable for solving MO-COPs such as scheduling problem. In this paper, a MOMAD is proposed that integrates MOEA/D algorithm with local search to enrich the tool-kit for solving MOFJSP.

3. Related Background Knowledge

3.1. Problem and Objective of MOFJSP. The MOFJSP can be formulated as follows. There are a set of n jobs $\mathcal{J} = \{J_1, J_2, \ldots, J_n\}$ and m machines $\mathcal{M} = \{M_1, M_2, \ldots, M_m\}$; each job J_i $(i = 1, 2, \ldots, n)$ contains one or more operations to be processed in accordance with the predetermined sequence. Each operation can be processed on any machine among its corresponding operable machine set $M_{ij} \in \mathcal{M}$. The problem is defined as T-FJSP iff $M_{ij} = \mathcal{M}$; otherwise, it is called P-FJSP [44]. MOFJSP not only assigns suitable processing machines for each operation but also determines the most reasonable processing sequence of operations assigned on the same machine in order to simultaneously optimize several objectives.

The following constraints should be satisfied in the process:

(1) At a certain time, a machine can process one operation at most, and one operation can be processed by only one machine at a certain moment.

(2) Each operation cannot be interrupted once processed.

(3) All jobs and machines are available at time 0.

(4) Different jobs share the same priority.

(5) There exists no precedence constraint among the operations of different jobs, but there exist precedence constraints among the operations belonging to the same job.

An instance of P-FJSP with three jobs and three machines is illustrated in Table 1. Let C_{ij} and p_{ijk} be the completion time of operation O_{ij} and its processing time on machine k, respectively. C_i denotes the completion time of job J_i. Three considered objectives are makespan, total workload, and critical workload which are formulated as follows:

$$\begin{aligned}
\min:\quad & \mathbf{F} = (f_1, f_2, f_3)^T, \\
& f_1\colon C_{\max} = \max\{C_i \mid i = 1, 2, \ldots, n\}, \\
& f_2\colon W_T = \sum_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n_i} p_{ijk} u_{ijk}, \\
& f_3\colon W_{\max} = \max_{1\le k\le m}\left\{\sum_{i=1}^{n}\sum_{j=1}^{n_i} p_{ijk} u_{ijk}\right\}, \\
& u_{ijk} \\
& = \begin{cases} 1, & \text{if machine } k \text{ is selected for operation } O_{ij} \\ 0, & \text{otherwise.} \end{cases}
\end{aligned} \tag{1}$$

Table 1: An instance of P-FJSP with 3 jobs on 3 machines.

Job	Operation	Machine		
		M_1	M_2	M_3
J_1	O_{11}	2	-	4
	O_{12}	6	3	-
	O_{13}	-	2	2
J_2	O_{21}	3	3	4
	O_{22}	-	2	5
	O_{23}	-	3	2
J_3	O_{31}	4	3	2
	O_{32}	4	-	2
	O_{33}	3	4	3

3.2. Disjunctive Graph Model. Disjunctive graph model $G = (V, U, E)$ has been adapted for representing feasible schedules of FJSP. V denotes nodes set, and each of them represents an operation. The virtual starting and ending operations are represented by two virtual nodes, 0 and $*$, respectively. U is the set of conjunctive arcs which connect adjacent operations of the same job, and each arc indicates the precedence constraint within the same job. E is the set of disjunctive arcs corresponding to the adjacent operations scheduled on the same machine. $E = \bigcup_{k=1}^{m} E_k$, where E_k denotes the disjunctive arcs set of machine k. The weight value above the node O_{ij} denotes p_{ijk}, and the selected machine to operate O_{ij} is labeled under the node O_{ij}, $p_0 = p_* = 0$.

Figure 1 shows the disjunctive graph of a feasible solution corresponding to the instance shown in Table 1, in which every disjunctive arc confirms a direction. This graph is called digraph. In digraph G, the longest path from node a to node b is termed as critical path, the length of which denotes the makespan of corresponding schedule. Besides, each operation on critical path is called critical operation. In Figure 1, there is one critical path $0 \to O_{11} \to O_{12} \to O_{33} \to *$ whose length equals 13.

Given that the local search in the following section can be well described, some concepts based on digraph G are denoted here. Suppose h is a node in G, and its corresponding operation is O_h. $M(G, h)$ denotes the corresponding machine to process O_h. $S^E(G, h)$ and $C^E(G, h)$ denote its earliest starting time and earliest completion time. Let $p_{h,M(G,h)}$ be the processing time of operation O_h on $M(G, h)$; the latest starting time $S^L(G, h, C_{\max}(G))$ and latest completion time

```
(1) Generate a set of weight vectors $\Lambda \leftarrow \{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_N\}$
(2) Generate weight vector groups $\{\zeta_1, \zeta_2, \ldots, \zeta_{n_c}\}$
(3) Get the neighborhood $B(i)$ of each weight vector $\boldsymbol{\lambda}_i$, $i = 1, 2, \ldots, N$, where $\{\boldsymbol{\lambda}_{i_1}, \boldsymbol{\lambda}_{i_2}, \ldots, \boldsymbol{\lambda}_{i_T}\}$ are the $T$
    closest weight vectors to $\boldsymbol{\lambda}_i$.
(4) $P0 \leftarrow$ Initialize the population( )
(5) Initialize Idealpoint: $\mathbf{z}^* = (z_1^*, z_2^*, \ldots, z_M^*)^T$
(6) Find the non-dominated solutions in initial population to construct the archive $Arc$
(7) while the termination criterion is not satisfied do
(8)   for $i = 1, 2, \ldots, N$ do
(9)     if rand(1) < $\delta$ then
(10)      $PP \leftarrow B(i)$
(11)    else
(12)      $PP \leftarrow \{1, 2, \ldots, N\}$
(13)    end if
(14)    Randomly select two different indexes $k, l$ from $PP$, $(k \neq i,\ l \neq i)$
(15)    $\mathbf{y} \leftarrow$ GeneticOperators $(\mathbf{x}_k, \mathbf{x}_l)$
(16)    UpdateIdealPoint $(\mathbf{y}, \mathbf{z}^*)$
(17)    Objective Normalization
(18)    Update Current Population
(19)    Update External Archive $Arc$
(20)  end for
(21)  $P \leftarrow$ LocalSearch($P$)
(22) end while
```

Algorithm 1: Framework of the proposed MOMAD.

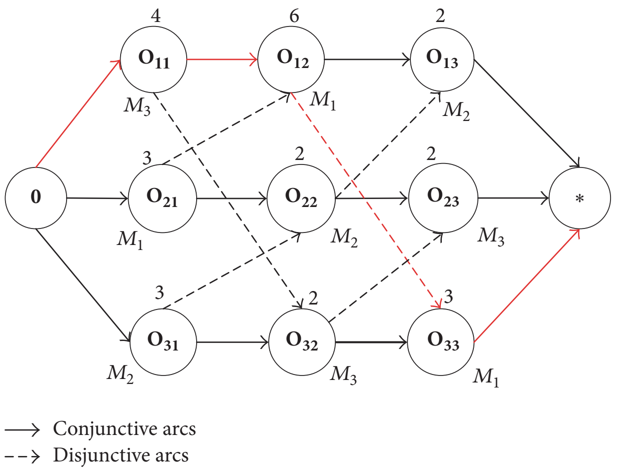


Figure 1: Illustration of the disjunctive graph.

$C^L(G, h, C_{\max}(G))$ without delaying the required makespan $C_{\max}(G)$ can be calculated by

$$\begin{aligned} C^E(G,h) &= S^E(G,h) + p_{h,M(G,h)}, \\ C^L(G,h,C_{\max}(G)) &= S^L(G,h,C_{\max}(G)) + p_{h,M(G,h)}. \end{aligned} \tag{2}$$

The predecessor and successor operation scheduled on the same machine right before or after O_h are denoted as $PM(G,h)$ and $SM(G,h)$, respectively. In addition, $PJ(G,h)$ and $SJ(G,h)$ are the predecessor and successor operation of O_h in the same job, respectively. O_h is a critical operation if and only if $C^E(G,h) = C^L(G,h,C_{\max}(G))$.

4. Framework of the Proposed MOMAD

The framework of MOMAD algorithm is formed by hybridizing MOEA/D with local search, which is given in Algorithm 1. First, a set of uniformly distributed weight vectors $\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_N$ is generated by Das and Dennis's approach [45], where each vector $\boldsymbol{\lambda}_i$ corresponds to subproblem i. Next, all the weight vectors are divided into n_c groups by K-means clustering. After calculating the Euclidean distance between any two weight vectors, the neighborhood $B(i)$ of $\boldsymbol{\lambda}_i$ is set by gathering T closest weight vectors. Then, the population containing N solutions is initialized. The ideal point vector $\mathbf{z}^*$ is obtained by calculating the infimum found so far of f_i. The archive Arc is established by founding the nondominated solutions in initial population. In Steps (9)–(13), the two mating parent solutions $\mathbf{x}_k$ and $\mathbf{x}_l$ are chosen from PP formed by $B(i)$ with the probability δ or by whole population with the probability $1 - \delta$. Then, the new solution $\mathbf{y}$ is generated by crossover and mutation, and finally $\mathbf{y}$ is used to update $\mathbf{z}^*$.

Steps (17)–(21) contain the updating and local search phase. The objective normalization is first adopted before population updating. Suppose $\boldsymbol{\lambda}_j = (\lambda_j^1, \lambda_j^2, \ldots, \lambda_j^M)^T$ is jth weight vector and z_i^{nad} is the largest value of f_i in the current population; then $\mathbf{y}$ is compared with the solutions from PP one by one, and the one that has poorer fitness in terms of (3) will be replaced by $\mathbf{y}$. It should be noted that the updating procedure will be terminated as soon as the predefined maximal replacing number n_r which benefits from keeping population diversity is reached or PP is empty. Afterwards, the updating of Arc is held. If no solutions in Arc dominate $\mathbf{y}$, then copy $\mathbf{y}$ into Arc and remove all the repeated and dominated solutions. Finally, after selecting the super

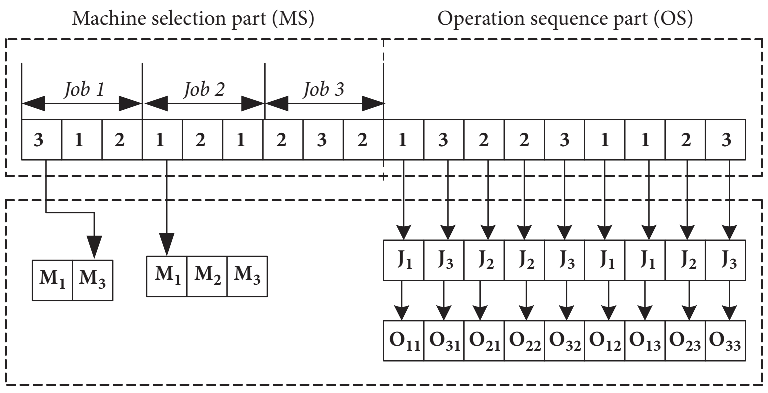

Machine selection part (MS)
Operation sequence part (OS)
Job 1
Job 2
Job 3
3 1 2 1 2 1 2 3 2 1 3 2 2 3 1 1 2 3
M1 M3
M1 M2 M3
J1 J3 J2 J2 J3 J1 J1 J2 J3
O11 O31 O21 O22 O32 O12 O13 O23 O33

Figure 2: Chromosome encoding.

solutions in current population, the local search is applied to get some improved solutions, and then they are rejected into the population to ameliorate it.

$$g^{te}\left(\mathbf{x} \mid \boldsymbol{\lambda}_j, \mathbf{z}^*\right) = \max_{1\le i\le M}\left\{\lambda_j^i\left|\frac{f_i(\mathbf{x}) - z_i^*}{z_i^{\text{nad}} - z_i^*}\right|\right\}. \qquad (3)$$

5. Detailed Description of Exploration and Exploitation in MOMAD

5.1. Chromosome Encoding and Decoding. In MOMAD, a chromosome coding consists of two parts: machine selection (MS) part and operation sequence (OS) part, which are encoded by machine selection and operation sequence vector, respectively. Each integer in MS vector represents the corresponding machine assigned for each operation in turn. As for OS vector, each gene is directly encoded with the job number. When compiling the chromosome from left to right, the kth occurrence of the job number refers to the kth operation of the corresponding job. Figure 2 shows a chromosome encoding of an P-FJSP instance which is shown in Table 1. M_3 is selected to process operation O_{11} and M_1 is selected to process operation O_{21}. The operation sequence can be interpreted as $O_{11} \rightarrow O_{31} \rightarrow O_{21} \rightarrow O_{22} \rightarrow O_{32} \rightarrow O_{12} \rightarrow O_{13} \rightarrow O_{23} \rightarrow O_{33}$.

Since it has been verified that the optimal schedule exists in active schedule [3], the greedy inserting algorithm [25] is employed for chromosome encoding to make sure the operation is positioned in the earliest capable time of its assigned machine. It should be noted that operation O_{ij} may be started earlier than O_{lk} while O_{lk} appears before O_{ij} in the OS. In order to make the encoding be able to reflect the actual processing sequence, the operation sequence in original OS will be reordered in the light of their actual starting time after decoding.

5.2. Population Initialization. Population initialization plays an important role in MOMAD performance since a high quality initial population with more diversity can avoid falling into premature convergence. Here, four machine assignment rules are used to produce the machine assignment vectors for MOFJSP. The first two rules are global selection and local selection proposed by Zhao et al. [46]. The third rule prefers to select a machine from the candidate machine set at random. The aim of the last rule is assigning each operation to a machine with the minimum processing time. In our MOMAD, for machine assignment initialization, 50% of individuals are generated by rule 1, 10% of individuals are formed with rule 4, and rule 2 and rule 3 take share of the rest of the population.

Once the machines are assigned to each operation, the sequence of operations should be considered next. A mixture of four operation sequencing rules is employed to generate the initial operation sequencing vectors. The probabilities of using four operation sequencing rules are set as 0.3, 0.2, 0.3, and 0.2, respectively.

Operation sequencing rules are as follows:

(1) Most Work Remaining [47]. The operations which have the most remaining processing time will be put into the operation sequencing vector first.

(2) Most Number of Operations Remaining (MOR) [47]. The operations which have the most subsequent operations in the same job will be preferentially taken into account.

(3) Shortest Processing Time (SPT) [48]. The operations with the shortest processing time will be firstly processed.

(4) Random Dispatching. It randomly orders the operations on each machine.

5.3. Exploration Using Genetic Operators. The problem-specific crossover and mutation operators are applied to produce the offspring, both of which are performed on each vector independently since the encoding of one chromosome has two components.

Crossover. For the MS, uniform crossover [3] is adopted. First of all, a subset of $r \in [1, D]$ positions is uniformly chosen at random, where D equals the total number of all operations. Then, two new individuals are generated by changing the gene between parent chromosomes corresponding to the selected positions. With respect to OS vector, the precedence preserving order-based (POX) [3] crossover is applied.

```
(1) for $j = 1$ to $n_c$ do
(2)     Randomly select a weight $\boldsymbol{\lambda}$ from the weight vector set $\zeta_i$;
(3)     Select all solutions $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_i}\}$ respectively correspond to weight vectors $\{\boldsymbol{\lambda}_1, \boldsymbol{\lambda}_2, \ldots, \boldsymbol{\lambda}_{\mathbf{n_i}}\}$ which
        belong to $\zeta_i$.
(4)     (MS, OS) ←Tournament Selection $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{n_i})$
(5)     $E'_j \leftarrow$ LocalSearchForIndividual ((MS, OS), $\boldsymbol{\lambda}_i$)
(6)     UpdateNeighborhood $(E'_j, B(\boldsymbol{\lambda}))$
(7)     $Arc \leftarrow$ UpdateArchive $(E'_j, Arc)$
(8) end for
```

Algorithm 2: LocalSearch (P).

Mutation. As for MS, two positions are chosen arbitrarily, and the genes are replaced by changing the different machine from their corresponding machine set. With regard to OS, the mutation is achieved by exchanging two randomly chosen operations.

5.4. Exploitation Using Local Search. It is widely accepted that integrating local search can effectually improve the performance of EAs, and it is an effective strategy to solve FJSP [31], so it is of great importance to design an effect local search method to make MOMAD keep good balance between exploration and exploitation. First of all, with the aim of saving the computational complexity as well as maintaining the population diversity, only a number of $\lfloor N \times P_{\text{ls}} \rfloor$ individuals are selected from entire evolutionary population to apply local search in each generation. Then, the following two critical issues will be introduced in detail in the next two subsections.

(1) How to select appropriate solutions to apply local search?

(2) Which local search method will be used?

5.4.1. Selection of Individuals for Local Search. When selecting a candidate individual each time, at first, a weight vector $\boldsymbol{\lambda}$ is chosen from the vector set ζ_i at random. Then, all incumbent solutions corresponding to different weight vectors which belong to ζ_i are selected. Next, all the selected individuals are compared according to (3) with the weight vector $\boldsymbol{\lambda}$, and the one which has the best fitness is selected as a candidate solution to apply local search. Finally, an improved solution obtained by local search is adopted to update neighborhood and archive. Denote n_i as the number of weight vectors which belong to ζ_i. The basic framework of local search can be summarized as Algorithm 2.

5.4.2. Description of Local Search Method. Considering that makespan is the most important and the hardest objective to be optimized among the three optimization objectives, the local search is performed on the decoded schedule of one chromosome rather than the chromosome itself. Suppose there are a set of $nc(G)$ critical operations $c(G) = \{\text{cor}_1, \text{cor}_2, \ldots, \text{cor}_{nc(G)}\}$, and let $C_{\max}(G)$ be the makespan of G. Then, the following theorems based on disjunctive graph are summarized as follows [2]:

(1) It has been proven that the makespan can only be reduced by moving critical operations $O_{ij} \in c(G)$.

(2) Suppose G_i^- is obtained by deleting one critical operation cor_i in G. Let ϕ_k be the set of operations processed on M_k which are sorted by the increasing of starting time (note that $\text{cor}_i \notin \phi_k$) in G_i^-; then two subsequences of ϕ_k denoted as R_k and L_k are defined as follows:

$$\begin{aligned} R_k &= \left\{v \in \phi_k \mid S^E(G, v) + p_{v,k} > S^E(G_i^-, \text{cor}_i)\right\}, \\ L_k &= \left\{v \in \phi_k \mid S^L(G, v, C_{\max}(G)) \right. \\ &\qquad \left. < S^L(G_i^-, \text{cor}_i, C_{\max}(G))\right\}. \end{aligned} \tag{4}$$

It has been verified that a feasible schedule G' can be achieved by inserting cor_i into a position $v \in \Upsilon_k$, where Υ_k contains all positions before all the operations of $R_k \backslash L_k$ and after all the operations of $L_k \backslash R_k$.

(3) The insertion of moving cor_i on position v must satisfy the following constraint:

$$\begin{aligned} &\max\left\{C^E(G_i^-, PM(G_i^-, v)), C^E(G_i^-, PJ(\text{cor}_i))\right\} \\ &\quad + p_{\text{cor}_i,k} \le \min\left\{S^L(G_i^-, v, C_{\max}(G)),\right. \\ &\quad \left. S^L(G_i^-, SJ(\text{cor}_i), C_{\max}(G))\right\}. \end{aligned} \tag{5}$$

When considering an action of finding a position on M_k for cor_i to insert into G_i^-, which is denoted as $\text{cor}_i \mapsto M_k$, only the positions in Υ_k should be taken into account. Insert cor_i into one position v in Υ_k if v satisfies (5), and other positions in Υ_k will no longer be considered. Suppose M_{cor_i} is the alternative machine set to operate cor_i; n_{cor_i} denotes the total actions of $\text{cor}_i \mapsto M_k$. Let $\varphi(c(G), M)$ be the action set $\{\text{cor}_i \mapsto M_k \mid i = 1, 2, \ldots, nc,\ M_k \in M_{\text{cor}_i}\}$, which contains $\sum_{i=1}^{n_c} n_{\text{cor}_i}$ actions. Now, a hierarchical strategy [27] is adopted to calculate Δt and Δc which, respectively, represent

```
(1) iter ← 0
(2) $G$ ← ChromosomeDecoding{MS, OS}
(3) $G_{best}$ ← $G$
(4) while $G \neq \emptyset$ and $iter$ < $\text{iter}_{max}$ do
(5)     $G'$ ← GetNeighborhood($G$)
(6)     if $G' \neq \emptyset$ then
(7)         if ($\mathbf{F}(G') \prec \mathbf{F}(G_{best})$) or (($\mathbf{F}(G') \parallel \mathbf{F}(G_{best})$)&($\mathbf{F}(G') \neq \mathbf{F}(G_{best})$)) then
(8)             $G_{best}$ ← $G'$
(9)         end if
(10)    end if
(11)    $G$ ← $G'$
(12)    iter = iter + 1
(13) end while
(14) $E'$ ← ChromosomeEncoding($G_{best}$)
(15) return $E'$
```

Algorithm 3: LocalSearchForIndividual $\{(\text{MS}, \text{OS}), \boldsymbol{\lambda}_i\}$.

the variation of total workload and critical workload. All actions in $\varphi(c(G), M)$ are sorted by nondescending order of Δt. If both actions have the same Δt, then the lowest Δc is chosen as a second criterion. The value of Δt, Δc is calculated as follows:

$$\Delta t\left(\text{cor}_i \longmapsto M_k\right) = p_{\text{cor}_i,k} - p_{\text{cor}_i,M(G,\text{cor}_i)},$$
$$\Delta c\left(\text{cor}_i \longmapsto M_k\right) = W_k\left(G\right) + p_{\text{cor}_i,k}. \quad (6)$$

These actions in sorted $\varphi(c(G), M)$ are considered in succession until a neighborhood G' is established. Whereafter, we compare G' with G_{best} by adopting an acceptance rule described as Step (7) in Algorithm 3. G_{best} will be replaced by G', providing that G' is not dominated by G and they are different from each other. It should be noted that the length of once local search is controlled by two points in order to save the computing cost. First, when getting a neighborhood of G, once an effective action in $\varphi(c(G), M)$ is found, a neighbor schedule G' is formed. Then, the remaining actions in $\varphi(c(G), M)$ will no longer be considered. The second effort is to set max iteration number $\text{iter}_{\max}$ so that the search will be terminated when the $\text{iter}_{\max}$ is exhausted.

5.4.3. Computational Complexity Analysis. The major computational costs in MOMAD are involved in Step (16)~Step (18) and Step (21). Step (16) performs $O(M)$ comparisons and assignments. The objective normalization in Step (17) requires $O(M)$ operations. Because the computing g^{te} for T neighborhood solutions and $O(M)$ basic operations are required in one computation, $O(MT)$ basic operations are needed for Step (18). Therefore, the total computational cost in Step (16)~Step (18) is $O(MN^2) + O(MN^2) + O(MNT)$ since it computes N times. When comparing $\mathbf{F}(G)$ and $\mathbf{F}(G_{\text{best}})$ in local search, it requires $O(M)$ basic operations in one iteration, and the worst case is $O(M\text{iter}_{\max})$ if the maximal iterations are reached. The computational cost of Step (21) is $O(MNP_{\text{ls}}\text{iter}_{\max})$ since it has $P_{\text{ls}} \times N$ passes. Thus, the total computational complexity of MOMAD is $O(MN^2) + O(MNP_{\text{ls}}\text{iter}_{\max})$.

Table 2: Parameter settings of the proposed MOMAD algorithm.

Parameters	Values
Population size (N)	120
Crossover probability (P_c)	1.0
Mutation probability (P_m)	0.1
Neighborhood size (T)	$0.1 \times N$
Controls parameters (δ)	0.9
Maximal replacing number (n_r)	1
Division (H)	14
Local search probability (P_{ls})	0.1
Number of weight vector cluster (ζ)	$P_{\text{ls}} \times N$
Maximal iterations of local search ($\text{iter}_{\max}$)	50

6. Experiments and Results

With the aim of testing the performance of MOMAD, 5 well-known Kacem instances (Ka 4×5, Ka 8×8, Ka 10×7, Ka 10×10, and Ka 15 × 10) [23] and 10 BRdata instances (MK01~MK10) [49] are underinvestigated in the experiment. The MOMAD is implemented in C++ language and on an Intel Core i3-4170 3.70 GHz processor with 4 GB of RAM.

6.1. Parameter Settings. With the purpose of eliminating the influence of random factors on the performance of the algorithm, it is necessary to independently run the proposed MOMAD 10 times on each test instance, and the algorithm will be terminated when reaching the maximal number of objective function evaluations in one run. The parameters used in MOMAD algorithm are listed in Table 2. Moreover, because of the different complexity of each test instance, the predefined maximal numbers of objective evaluations corresponding to different problems are listed in the second column in Table 3.

6.2. Performance Metrics. In order to quantitatively evaluate the performance of the compared algorithms, three

Table 3: Comparison of three different aggregation functions on average IGD and HV values over 10 independent runs for all Kacem and BRdata instances.

Instance	NFs[§]	IGD			HV		
		MOMAD	MOMAD-WS	MOMAD-PBI	MOMAD	MOMAD-WS	MOMAD-PBI
ka 4 × 5	40000	0.1054	0.2108	0.1845	0.5910	0.5843	0.5860
ka 8 × 8	40000	0.1071	0.1252	0.1541	0.6574	0.6191	0.5917
ka 10 × 7	40000	0.0298	0.0400	0.0651	1.0594	1.0481	0.9762
ka 10 × 10	40000	0.1394	0.0767	0.1676	0.6802	**0.9127**‡	0.6873
ka 15 × 10	40000	0.4002	0.4108	0.4405	0.3727	0.3474	0.2849
MK01	50000	0.0304	0.0254	0.0619†	1.0316	1.0368	0.9975†
MK02	50000	0.0415	0.0495	0.0972†	0.9908	0.9772	0.9374†
MK03	50000	0.0233	0.1251	0.2837†	0.8296	0.7027	0.4929†
MK04	50000	0.0340	0.0282	0.1128†	1.1437	1.1455	1.0758†
MK05	50000	0.0072	0.0057	0.1181†	1.0576	1.0636	0.9120†
MK06	60000	**0.0473**	0.0529†	0.0993†	0.9700	0.9610	0.8906†
MK07	50000	0.0218	0.0313	0.1288†	**0.9906**	0.9734†	0.8484†
MK08	50000	0.0000	0.0075	0.1943†	0.5755	0.5705	0.3832†
MK09	50000	0.0365	0.0396	0.1489†	**1.1893**	1.1725†	1.0642†
MK10	70000	**0.0391**	0.0765†	0.0597†	**1.1480**	1.0751†	1.0743†

§ means the number of function evaluations; † means that the results are significantly outperformed by MOMAD; ‡ means that the results are significantly better than MOMAD.

quantitative indicators are employed to make the comparison more convincing, and they are described as follows.

(1) Inverted Generational Distance (IGD) [50]. Let P_{known} be the approximate PF obtained by the compared algorithm and P^* be the reference PF uniformly distributed in the object space. IGD metric measures the distance between P_{known} and P^* with smaller value representing better performance. The IGD metric can well reflect the convergence and diversity simultaneously to some extent if P^* is larger enough to well represent the reference PF; that is,

$$\text{IGD}\left(P^*, P_{\text{known}}\right) = \frac{\sum_{\mathbf{y}^* \in P^*} d\left(\mathbf{y}^*, P_{\text{known}}\right)}{|P^*|},$$
$$d\left(\mathbf{y}^*, P_{\text{known}}\right) = \min_{\mathbf{y} \in P_{\text{known}}} \left\{ \sqrt{\sum_{i=1}^{M} \left(y_i^* - y_i\right)^2} \right\}, \; \mathbf{y}^* = \left(y_1^*, y_2^*, \ldots, y_M^*\right)^T, \; \mathbf{y} = \left(y_1, y_2, \ldots, y_M\right)^T. \quad (7)$$

In this experiment, since all benchmark instances are tested without knowledge about their actual PFs, P^* used in calculating IGD metric is obtained by two steps. First, we merge all final nondominated solutions obtained by all compared algorithms during all the independent runs. Then, we select the nondominated solutions from the mixed solution set as P^*.

(2) Set Coverage (C) [51]. C metric can directly reflect the dominance relationship between two groups of approximate PFs. Let A and B be two approximate PFs that are obtained by two different algorithms; then $C(A,B)$ is defined as

$$C(A,B) = \frac{|\{b \in B \mid \exists a \in A : a \prec b\}|}{|B|}. \quad (8)$$

Although $C(A,B)$ represents the percentage of solutions in B that are dominated by at least one solution in A, in general, $C(A,B) \neq 1 - C(B,A)$. If $C(A,B)$ is larger than $C(B,A)$, algorithm A is better than algorithm B to some extent.

(3) Hypervolume (HV) [51]. HV metric is employed to measure the volume of hypercube enclosed by PF A and reference vector $\mathbf{r}_{\text{ref}} = (r_1, r_2, \ldots, r_M)^T$ with lager values representing better performance. It can be obtained by

$$\text{HV}(A) = \bigcup_{a \in A} \text{vol}(a), \quad (9)$$

where vol(a) is the volume of hypercube enclosed by solution a in PF A and reference vector $\mathbf{r}_{\text{ref}} = (r_1, r_2, \ldots, r_M)^T$. HV measure can reflect both convergence and diversity of corresponding PF to a certain degree.

For the convenience of computation, all objective vectors of the Pareto solutions are normalized based on (10) before calculating all three metrics. Thus, the reference vectors of

Table 4: Comparison of three different aggregation functions on average C value over 10 independent runs for all 15 problems.

Instance	MOMAD(A) versus MOMAD-WS(B)		MOMAD(A) versus MOMAD-PBI(C)	
	$C(A, B)$	$C(B, A)$	$C(A, C)$	$C(C, A)$
ka 4×5	0.0000	0.0000	0.0000	0.0000
ka 8×8	0.0250	0.0000	0.0983	0.0333
ka 10×7	0.1000	0.0000	0.1167	0.0333
ka 10×10	0.0583	0.0667	0.0250	0.0333
ka 15×10	0.3167	0.2000	0.2500	0.2500
MK01	0.2070	0.2221	**0.4386**	0.0373
MK02	0.2296	0.1261	**0.6180**	0.1143
MK03	0.2034	0.0453	**0.5069**	0.0121
MK04	0.1689	0.1868	**0.3457**	0.0704
MK05	0.1727	0.1300	**0.5658**	0.0174
MK06	0.5158	0.3765	0.3546	0.2746
MK07	**0.3294**	0.1690	**0.5408**	0.1346
MK08	0.0143	0.0000	**0.3300**	0.0000
MK09	**0.5997**	0.2937	**0.5915**	0.1200
MK10	**0.5408**	0.1637	0.2795	0.3700

Table 5: Performance evaluation of the effect of local search using IGD, HV, and C values over 10 independent runs for all 15 problems.

Instance	IGD		HV		MOMAD(A) versus MOEA/D(B)	
	MOMAD	MOEA/D	MOMAD	MOEA/D	$C(A, B)$	$C(B, A)$
ka 4×5	**0.1054**	0.2635	**0.5910**	0.5810	0.0000	0.0000
ka 8×8	0.1071	0.1398	0.6574	0.6188	0.0500	0.0000
ka 10×7	0.0298	0.0687	1.0594	0.9828	0.1333	0.0333
ka 10×10	**0.1394**	0.3617	**0.6802**	0.3178	0.1667	0.0333
ka 15×10	0.4002	0.3666	0.3727	0.4776	**0.3833**	0.0000
MK01	**0.0304**	0.0537	**1.0316**	1.0161	**0.5218**	0.0358
MK02	**0.0415**	0.0740	**0.9908**	0.9263	**0.4055**	0.0386
MK03	0.0233	0.0077	0.8296	0.8363	0.0258	0.0330
MK04	**0.0340**	0.0441	**1.1437**	1.1297	**0.2585**	0.0800
MK05	0.0072	0.0062	1.0576	1.0621	**0.2326**	0.0616
MK06	**0.0473**	0.0540	**0.9700**	0.9223	0.3511	**0.4992**
MK07	**0.0218**	0.0418	**0.9906**	0.9552	0.2602	0.1224
MK08	0.0000	0.0000	0.5755	0.5755	0.0000	0.0000
MK09	0.0365	0.0406	**1.1893**	1.1622	**0.7050**	0.2499
MK10	**0.0391**	0.0737	**1.1480**	1.0525	**0.5546**	0.2328

all benchmark instances for calculating HV value are set as $(1.1, 1.1, 1.1)^T$.

$$\widetilde{f_i(\mathbf{x})} = \frac{\left(f_i(\mathbf{x}) - f_i^{\min}\right)}{\left(f_i^{\max} - f_i^{\min}\right)}, \tag{10}$$

where $f_i^{\max}$ and $f_i^{\min}$ are the supremum and infimum of $f_i(\mathbf{x})$ over all nondominated solutions obtained by gathering all compared algorithms.

6.3. Performance Comparison with Several Variants of MOMAD. Because the implementation of algorithm framework is not unique and there are some different strategies employed to instantiate it, several variants of MOMAD will be first studied. MOEA/D is a simplified algorithm designed by eliminating local search from MOMAD to investigate its effectiveness. In the interest of studying the effect of different aggregation functions, MOMAD-WS and MOMAD-PBI are formed by replacing the Tchebycheff approach with weighted sum (WS) [7] and penalty-based boundary intersection (PBI) approach [7]. The Wilcoxon signed-rank test [52] is performed on the sample data of three metrics obtained after 10 independent runs with the significance of 0.05, and the one which is significantly better than others is marked in bold. Three metric values are listed in Tables 3–5.

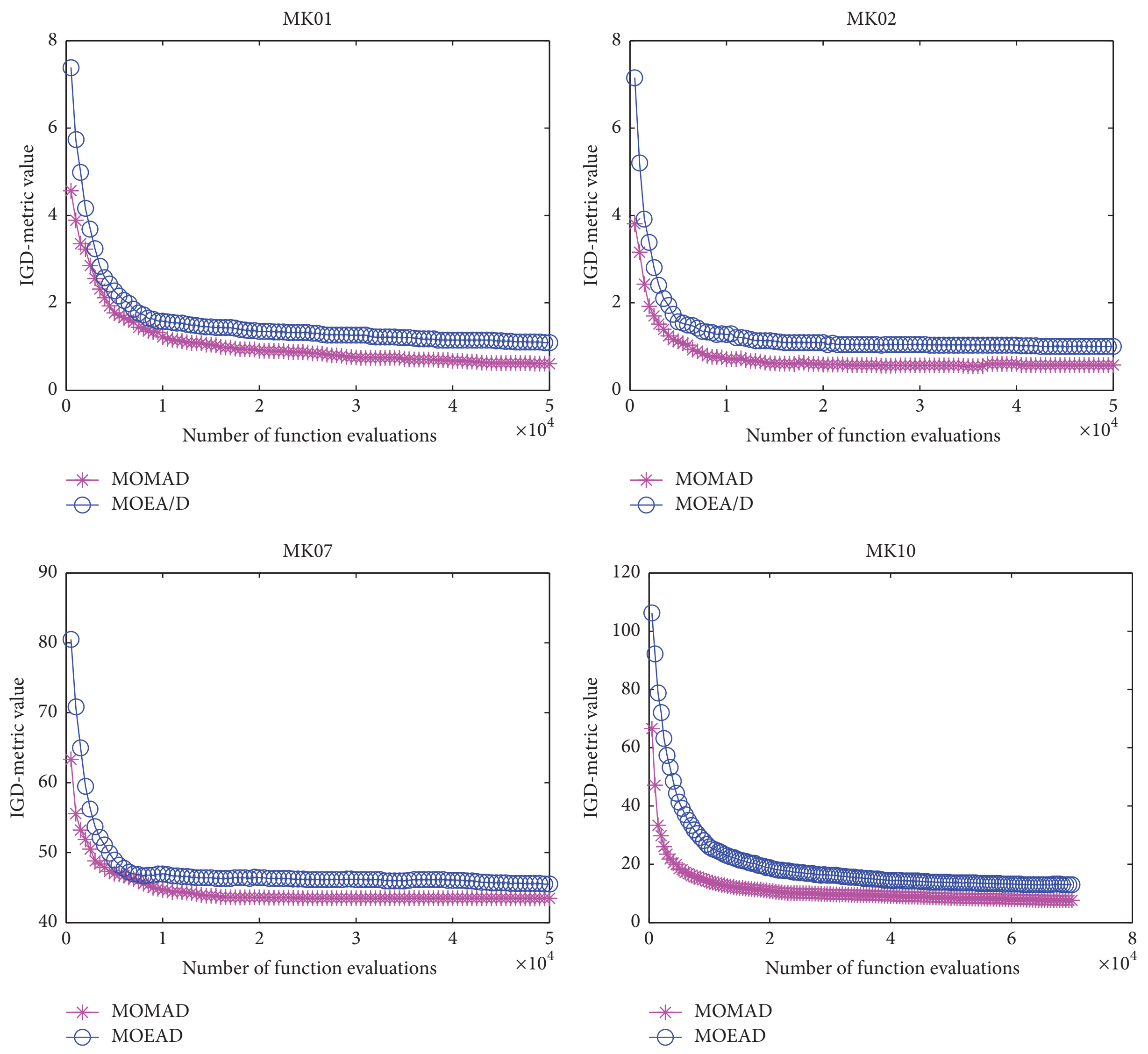


Figure 3: Convergence graphs in terms of average IGD value obtained by MOMAD and MOEA/D for MK01, MK02, MK07, and MK10 problems.

In Tables 3 and 4, the results of performance comparison between MOMAD, MOMAD-WS, and MOMAD-PBI are listed. MOMAD is significantly better than MOMAD-WS on MK10 for IGD, HV, and C values. Besides, MOMAD performs better than MOMAD-WS on MK07 and MK09 in terms of HV and C value and is also better than MOMAD-WS on MK06 for IGD metric. In contrast, MOMAD-WS only achieves a better HV value on Ka 10×10. When comparing with MOMAD-PBI, MOMAD is better than MOMAD-PBI for all the BRdata instances in terms of IGD and HV values. In addition, MOMAD also obtains better C metric values on 8 out of 10 instances. In summary, the presented results indicate that Tchebycheff aggregation function is more suitable to be utilized in MOEA/D framework when solving MOFJSP.

To understand the effectiveness of problem-specific local search, the comparison between MOMAD and MOEA/D is conducted, and the three metric values over 10 independent runs are shown in Table 5. It is easily observed that, as for IGD value, there are significant differences in the performances of the two algorithms on 8 test problems, and MOMAD significantly outperforms MOEA/D on all these instances. As for the other two metrics, the situations are similar to IGD metric. MOMAD outperforms MOEA/D on 9 out of total 15 instances in terms of HV metric and 7 out of 15 instances for C metric, while MOEA/D only obtains a better C value on MK06. Based on the above comparison results and analyses, MOMAD is much powerful than MOEA/D, which well verifies the effectiveness of local search.

With the aim of intuitively comparing the convergence performance of the two algorithms, the evolutions of average IGD values in MOMAD and MOEA/D on four selected BRdata instances are illustrated in Figure 3. As can be clearly seen from this figure, with the increasing number of function evaluations, the IGD values in all four instances

Table 6: Comparison of the Kacem instance by listing the nondominated solutions.

Instance	F	PSO + SA		hGA	PSO + TS		Xing		HTSA			EDA				MOMAD			
		1	2	1	1	2	1	2	1	2	3	1	2	3	4	1	2	3	4
ka 4 × 5	f_1				**11**		**12**		**11**	**12**		**11**	**11**	**12**		**11**	**11**	**12**	**13**
	f_2				**32**		**32**		**32**	**32**		**34**	**32**	**32**		**34**	**32**	**32**	**33**
	f_3				**10**		**8**		**10**	**8**		**9**	**10**	**8**		**9**	**10**	**8**	**7**
ka 8 × 8	f_1	**15**	**16**	**14**	**14**	**15**	**14**	15	**14**	**15**		**14**	**15**			**14**	**15**	**16**	**16**
	f_2	**75**	**73**	**77**	**77**	**75**	**77**	76	**77**	**75**		**77**	**75**			**77**	**75**	**73**	**77**
	f_3	**12**	**13**	**12**	**12**	**12**	**12**	12	**12**	**12**		**12**	**12**			**12**	**12**	**13**	**11**
ka 10 × 7	f_1						**11**	**11**	**11**	**11**		**11**	**11**			**11**	**11**	**12**	
	f_2						**61**	**62**	**61**	**62**		**61**	**62**			**61**	**62**	**60**	
	f_3						**11**	**10**	**11**	**10**		**11**	**10**			**11**	**10**	**12**	
ka 10 × 10	f_1	7		**7**	7		**7**	**8**	**7**	**7**	**8**	**7**	**7**	**8**	**8**	**7**	**7**	**8**	**8**
	f_2	44		**43**	43		**42**	**42**	**43**	**42**	**42**	**43**	**42**	**41**	**42**	**43**	**42**	**41**	**42**
	f_3	6		**5**	6		**6**	**5**	**5**	**6**	**5**	**5**	**6**	**7**	**5**	**5**	**6**	**7**	**5**
ka 15 × 10	f_1	12		**11**	11		**11**	**11**	**11**			**11**	**11**			**11**	**11**		
	f_2	91		**91**	93		**91**	**93**	**93**			**91**	**93**			**91**	**93**		
	f_3	11		**11**	11		**11**	**10**	**10**			**11**	**10**			**11**	**10**		

Table 7: Comparison between MOMAD and other algorithms using IGD metric for Kacem and BR data instances.

Instance	Algorithm								
	MOMAD	MOGA	PLS	HSFLA	HMOEA	SEA	P-EDA	hDPSO	PRMOTS + IS
ka 4 × 5	**0.0000**	0.1954	**0.0000**	—	**0.0000**	**0.0000**	**0.0000**	—	**0.0000**
ka 8 × 8	**0.1414**	0.2532	**0.1414**	**0.1414**	0.2532	**0.1414**	**0.1414**	**0.1414**	**0.1414**
ka 10 × 7	**0.0000**	—	**0.0000**	—	**0.0000**	**0.0000**	**0.0000**	—	**0.0000**
ka 10 × 10	**0.0000**	0.1250	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
ka 15 × 10	**0.0000**	0.3571	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	0.1429	**0.0000**
MK01	**0.0037**	0.1525	0.0525	0.1909	0.0300	0.0078	0.0307	0.0984	0.0042
MK02	**0.0000**	0.0680	0.0662	0.1493	0.0119	0.0357	0.0119	**0.0000**	0.0287
MK03	**0.0643**	0.3933	0.2119	0.2119	0.0838	**0.0643**	0.1911	0.0949	**0.0643**
MK04	**0.0242**	0.1470	—	0.1508	—	0.0271	0.0617	0.0540	0.0340
MK05	0.0245	0.2486	—	**0.0185**	—	0.0245	0.0245	0.0596	0.0223
MK06	0.0243	0.1457	—	0.1283	—	0.0296	0.0774	0.1377	**0.0245**
MK07	0.0924	**0.0243**	—	0.1174	—	0.1029	0.0924	0.0976	0.0841
MK08	**0.0440**	0.1709	0.1519	0.1519	**0.0440**	0.0567	**0.0440**	0.0540	**0.0440**
MK09	0.0115	0.2491	—	0.1083	—	**0.0056**	0.0648	0.1520	0.0305
MK10	**0.0186**	0.1111	—	0.0902	—	0.0419	0.0762	0.1653	0.0517

For each instance, the minimal IGD values obtained by the compared algorithms are marked in bold.

gradually decrease and tend to be stable, which indicates that both algorithms have good convergence. Since MOMAD achieves lower IGD convergence curves, it is easily observed that MOMAD achieves better convergence property and convergence efficiency than MOEA/D.

6.4. Performance Comparison with Other Algorithms. In this subsection, comparison between MOMAD and several state-of-the-art algorithms are made. First, to compare MOMAD with algorithms solving MOFJSP by using a priori approach, MOMAD is compared on five Kacem instances with PSO + SA [17], hGA [18], PSO + TS [19], Xing et al.'s algorithm [20], HTSA [21], and EDA [22]. All Pareto solutions marked in bold are listed in Table 6. Next, in Tables 7–10, MOMAD is compared with eight algorithms recently proposed for solving MOFJSP by using Pareto approach that are MOGA [25], PLS [28], HSFLA [37], HMOEA [30], SEA [29], P-EDA [31], hDPSO [34], and PRMOTS + IS [32]. It should be pointed out that these compared algorithms list the results after predefined runs rather than each run in their original literatures. Therefore, the statistical comparisons as made before no longer apply. In this subsection, the three metrics are computed for the set of PFs collected over predefined runs of each algorithm.

As shown in Table 6, first, the MOMAD can obtain more Pareto solutions than all other algorithms for solving the five instances except EDA for Ka 10×10 and Ka 15×10 and Xing for Ka 15×10. Second, as for Ka 10×10, the solution (7, 44, 6)

Table 8: Comparison between MOMAD and other algorithms using HV metric for Kacem and BR data instances.

Instance	Algorithm								
	MOMAD	MOGA	PLS	HSFLA	HMOEA	SEA	P-EDA	hDPSO	PRMOTS + IS
ka 4 × 5	**0.5977**	0.5777	**0.5977**	—	**0.5977**	**0.5977**	**0.5977**	—	**0.5977**
ka 8 × 8	**0.5185**	0.3385	**0.5185**	**0.5185**	0.3385	**0.5185**	**0.5185**	**0.5185**	**0.5185**
ka 10 × 7	**0.4560**	—	**0.4560**	—	**0.4560**	**0.4560**	**0.4560**	—	**0.4560**
ka 10 × 10	**0.9060**	0.6560	**0.9060**	**0.9060**	**0.9060**	**0.9060**	**0.9060**	**0.9060**	**0.9060**
ka 15 × 10	**1.0167**	0.2739	**1.0167**	**1.0167**	**1.0167**	**1.0167**	**1.0167**	0.7310	**1.0167**
MK01	1.2130	1.0833	1.1820	0.7579	1.2003	1.2110	1.2022	1.1680	**1.2132**
MK02	**0.9691**	0.9152	0.9173	0.8032	0.9538	0.9402	0.9586	**0.9691**	0.9370
MK03	**0.6659**	0.6296	0.5443	0.5443	0.6574	**0.6659**	0.6405	0.6331	**0.6659**
MK04	**1.1118**	1.0882	—	0.9804	—	1.1084	1.0893	1.0990	1.1044
MK05	0.8274	0.7805	—	0.8234	—	0.8274	**0.8290**	0.7912	0.8131
MK06	0.9018	0.8163	—	0.7816	—	0.8766	0.8121	0.8307	**0.9454**
MK07	**0.1926**	0.7863	—	0.1805	—	0.1804	**0.1926**	0.1900	0.1894
MK08	**0.5521**	0.4469	0.4819	0.4819	**0.5521**	0.5451	**0.5521**	0.5375	**0.5521**
MK09	**1.3115**	1.2575	—	1.2742	—	1.2858	1.2939	1.3013	1.2897
MK10	**1.0365**	0.8106	—	0.9372	—	0.9496	0.9369	0.7976	0.9652

For each instance, the greater HV values obtained by the compared algorithms are marked in bold.

Table 9: Comparison between MOMAD and other algorithms using C metric for Kacem and BR data instances.

Instance	MOMAD(A) versus MOGA(B)		MOMAD(A) versus PLS(C)		MOMAD(A) versus HSFLA(D)		MOMAD(A) versus HMOEA(E)	
	$C(A, B)$	$C(B, A)$	$C(A, C)$	$C(C, A)$	$C(A, D)$	$C(D, A)$	$C(A, E)$	$C(E, A)$
ka 4 × 5	**0.0000**	**0.0000**	**0.0000**	**0.0000**	—	—	**0.0000**	**0.0000**
ka 8 × 8	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
ka 10 × 7	—	—	**0.0000**	**0.0000**	—	—	**0.0000**	**0.0000**
ka 10 × 10	**0.2500**	0.0000	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
ka 15 × 10	**0.6667**	0.0000	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
MK01	**0.7500**	0.0000	**0.7143**	0.0000	**0.9091**	0.0000	**0.5833**	0.0000
MK02	**0.5000**	0.0000	**0.7500**	0.0000	**1.0000**	0.0000	**0.2500**	0.0000
MK03	**0.1000**	0.0000	**0.8571**	0.0000	**0.8571**	0.0000	**0.0000**	**0.0000**
MK04	**0.4000**	0.1034	—	—	**0.8000**	0.1724	—	—
MK05	**0.0000**	**0.0000**	—	—	**0.3571**	0.0000	—	—
MK06	**0.3000**	0.1387	—	—	**0.8667**	0.0438	—	—
MK07	0.0000	**0.5000**	—	—	**0.6667**	0.0000	—	—
MK08	0.0000	**0.1111**	**0.6250**	0.0000	**0.6250**	0.0000	**0.0000**	**0.0000**
MK09	**0.6667**	0.0000	—	—	**1.0000**	0.0000	—	—
MK10	**1.0000**	0.0000	—	—	**0.6000**	0.1747	—	—

For each instance, the greater set coverage values obtained by the compared algorithms are marked in bold.

obtained by PSO + SA and solution (7, 43, 6) obtained by PSO + TS are both dominated by (7, 42, 6) and (7, 43, 5) obtained by MOMAD. Besides, when comparing with Xing, one solution (15, 76, 12) of Ka 8 × 8 obtained by Xing is dominated by (15, 75, 12) got by MOMAD. By analyzing the results of Ka 15 × 10, two solutions (i.e., (12, 91, 11) and (11, 93, 11)) which are, respectively, obtained by PSO + SA and PSO + TS are, respectively, dominated by (11, 91, 11) and (11, 93, 10) achieved by MOMAD. According to the above comparison results and analyses, when comparing with the algorithms based on a priori approach, MOMAD can obtain more nondominated solutions with higher quality.

Tables 7 and 8 show the comparison results of IGD and HV values between MOMAD and eight Pareto-based algorithms. First, it can be observed that except MOGA and hDPSO, MOMAD and other 6 algorithms can find all the nondominated solutions of Ka 4×5, Ka 10×7, Ka 10×10, and Ka 15 × 10. Although there exist no algorithms that can find all nondominated solutions of Ka 8 × 8, MOMAD and other 6 algorithms are better than MOGA and HMOEA. Next, we

Table 10: Comparison between MOMAD and other algorithms using C metric for Kacem and BR data instances.

Instance	MOMAD(A) versus SEA(B)		MOMAD(A) versus P-EDA(C)		MOMAD(A) versus hDPSO(D)		MOMAD(A) versus PRMOTS + IS(E)	
	$C(A,B)$	$C(B,A)$	$C(A,C)$	$C(C,A)$	$C(A,D)$	$C(D,A)$	$C(A,E)$	$C(E,A)$
ka 4 × 5	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
ka 8 × 8	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
ka 10 × 7	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
ka 10 × 10	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**
ka 15 × 10	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.6667**	0.0000	**0.0000**	**0.0000**
MK01	**0.1818**	0.0000	**0.4545**	0.0000	**0.0000**	**0.0000**	0.1818	**0.3000**
MK02	**0.1429**	0.0000	**0.2500**	0.0000	**0.0000**	**0.0000**	**0.4444**	0.0000
MK03	**0.0000**	**0.0000**	**0.1111**	0.0000	**0.0909**	0.0000	**0.0000**	**0.0000**
MK04	0.0000	**0.4138**	**0.1875**	0.0345	0.0000	**0.3448**	0.0870	**0.2414**
MK05	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.4000**	0.0000
MK06	0.1165	**0.6350**	**0.7869**	0.1241	**0.1250**	0.1241	0.2619	**0.5912**
MK07	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.1000**	0.0000	**0.3478**	0.0000
MK08	**0.0000**	**0.0000**	**0.0000**	**0.0000**	**0.1111**	0.0000	**0.0000**	**0.0000**
MK09	0.3281	**0.5714**	**0.9481**	0.0286	**0.1429**	0.1286	**0.8817**	0.1143
MK10	**0.5072**	0.2651	**0.7250**	0.1024	**0.2857**	0.0653	**0.5602**	0.2349

For each instance, the greater set coverage values obtained by the compared algorithms are marked in bold.

focus on the BRdata test set. When considering IGD metric, MOMAD obtains the best values on 6 out of 10 instances and the second-best IGD values on three instances. The situation of HV metric is much similar to IGD metric. MOMAD achieves the best HV values on 7 out of 10 instances and yields the second-best HV values on all the other instances. MOGA obtains the best IGD value on MK07, but MOMAD is better than MOGA on other 9 instances. The situation is much similar to SEA, P-EDA, and PRMOTS + IS. Although they exhibit superior performance over MOMAD on several instances for IGD and HV values, in most cases, they perform relatively worse than MOMAD.

MOMAD is compared with eight algorithms in Tables 9 and 10 by using set coverage value. It is clearly indicated that MOMAD is much superior to five algorithms except for MOGA, SEA, and PRMOTS + IS. When comparing with MOGA, MOMAD is worse than MOGA on MK07 and MK08, but MOMAD is better than MOGA on 7 BRdata instances. Compared with SEA, MOMAD is generally better on MK01, MK02, and MK10 instances, but on MK04, MK06, and MK09, SEA generally shows higher performance. As for PRMOTS + IS, MOMAD is superior to it on MK02, MK05, MK07, MK09, and MK10, whereas PRMOTS + IS only wins in MK01, MK04, and MK06.

Table 11 shows the comparison between MOMAD and MOGA on 18 DPdata instances which are designed in [53]. The predefined maximal function evaluation of each instance is shown in second column, and MOMAD is independently run 5 times at a time. From the IGD and HV value, it is easily observed that the performance of MOMAD is much better than MOGA since MOGA only obtains a better IGD value on 07a and a better HV value on 02a. The comparison of C metric is similar to the IGD and HV, MOMAD achieves 16 significantly better results, and there is no significant difference between MOMAD and MOGA on 02a and 09a. In addition, it can also be observed that C (MOMAD, MOGA) equals one on 12 test instances which means, as for these 12 instances, every solution obtained by MOGA is dominated by at least one solutions by MOMAD.

The objective ranges of MOMAD and PRMOTS + IS for DPdata instances are given in Table 12. The MOMAD finds a wider spread of nondominated solutions than PRMOTS + IS especially in terms of makespan and total workload. Thus, it can be concluded that MOMAD is more effective than PRMOTS + IS in terms of exploring a search space.

According to the above extensive IGD and HV values, the average performance scores of 10 BRdata instances are further recorded to rank the compared algorithm, which make it easier to quantify the overall performance of these algorithms by intuitive observation. For each specific BRdata instance, suppose $\mathrm{Alg}_1, \mathrm{Alg}_2, \ldots, \mathrm{Alg}_l$, respectively, denote the l algorithms employed in comparison. Let ϑ_{ij} be 1, iff Alg_j obtains smaller IGD and bigger HV value than Alg_i. Then, the performance $P(\mathrm{Alg}_i)$ of each algorithm Alg_i can be calculated as [54]

$$P\left(\mathrm{Alg}_i\right) = \sum_{\substack{j=1 \\ j\neq i}}^{l} \vartheta_{ij}. \tag{11}$$

It should be noted that the smaller the score, the better the algorithm. Here, PLS and HMOEA only consider part of instances, so we first rank all algorithms on MK01~MK03 and MK08 instances, and then we rank these algorithms except PLS and HMOEA on the remaining 6 instances. Figure 4 shows the average performance score of IGD and HV values

Table 11: Comparison between MOMAD and MOGA using IGD, HV, and C metric for DP data.

Instance	NFs[§]	IGD		HV		MOMAD versus MOGA	
		MOMAD	MOGA	MOMAD	MOGA	C(MOMAD, MOGA)	C(MOGA, MOMAD)
01a	50000	**0.0000**	0.2121	**1.3310**	1.0743	**1.0000**	0.0000
02a	50000	**0.2705**	0.3002	0.6117	**0.8576**	**0.0000**	**0.0000**
03a	50000	**0.1852**	0.5899	**0.9950**	0.2615	**0.5000**	0.0000
04a	50000	**0.0000**	0.4022	**1.1059**	0.6042	**1.0000**	0.0000
05a	60000	**0.0000**	0.4845	**1.2857**	0.6156	**1.0000**	0.0000
06a	60000	**0.0000**	0.6834	**1.3126**	0.4432	**1.0000**	0.0000
07a	50000	0.5034	**0.2517**	**1.1886**	0.6220	**0.6667**	0.0000
08a	50000	**0.0755**	0.6998	**1.1682**	0.1780	**0.5000**	0.0000
09a	50000	**0.3162**	0.3802	**1.1149**	0.4647	**0.0000**	**0.0000**
10a	70000	**0.0028**	0.3334	**1.0840**	0.5980	**0.7500**	0.0000
11a	70000	**0.0000**	0.7135	**1.2853**	0.3112	**1.0000**	0.0000
12a	70000	**0.0000**	0.7982	**1.3137**	0.2533	**1.0000**	0.0000
13a	60000	**0.0000**	1.1040	**1.2866**	0.1062	**1.0000**	0.0000
14a	60000	**0.0000**	0.9852	**1.2799**	0.1414	**1.0000**	0.0000
15a	70000	**0.0000**	0.6547	**1.2388**	0.2967	**1.0000**	0.0000
16a	70000	**0.0000**	0.4685	**1.1568**	0.3833	**1.0000**	0.0000
17a	70000	**0.0000**	0.8188	**1.2872**	0.2075	**1.0000**	0.0000
18a	70000	**0.0000**	1.0231	**1.3079**	0.0975	**1.0000**	0.0000

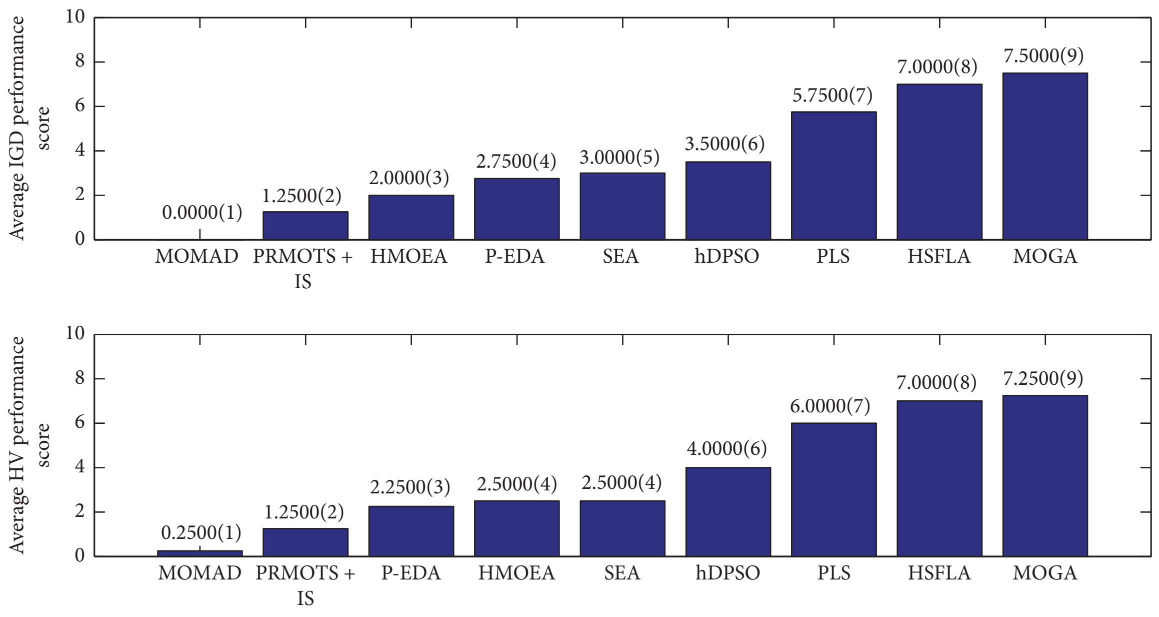
§ means the number of function evaluations.

Table 12: Objective ranges for the DPdata.

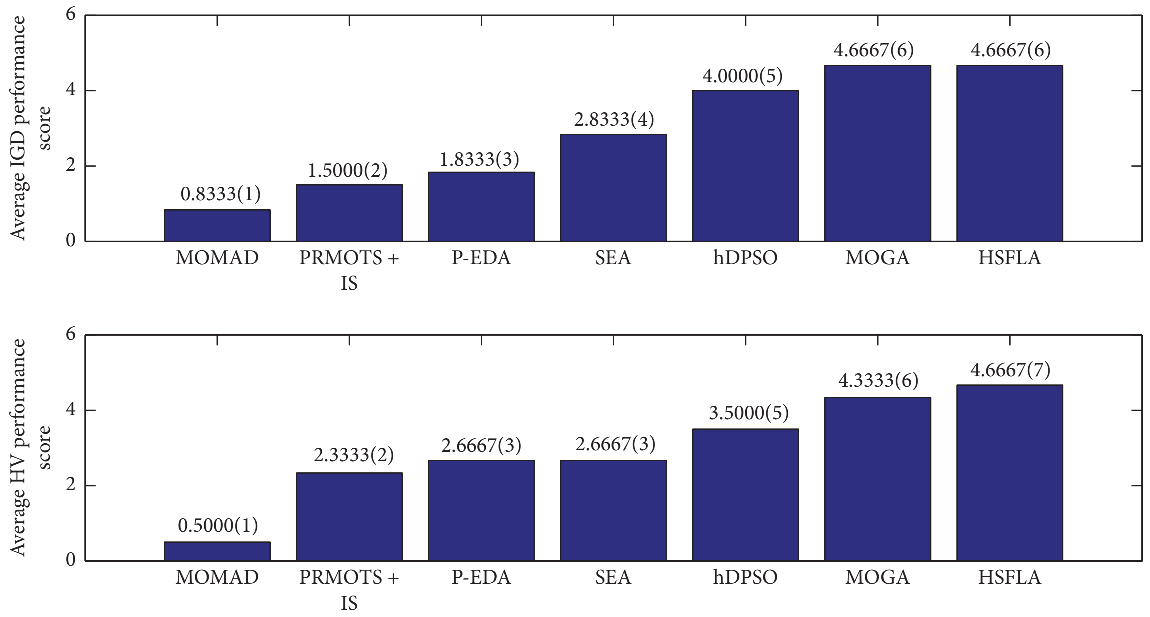
Instance	f_1(min, max)		f_2(min, max)		f_3(min, max)	
	MOMAD	PRMOTS + IS	MOMAD	PRMOTS + IS	MOMAD	PRMOTS + IS
01a	**(2561, 2561)**	(2592, 2596)	**(11137, 11137)**	**(11137, 11137)**	**(2505, 2505)**	(2508, 2549)
02a	**(2340, 2378)**	(2345, 2441)	**(11137, 11137)**	**(11137, 11137)**	(2232, **2236**)	(**2228**, 2238)
03a	**(2267, 2334)**	(2317, 2351)	**(11137, 11137)**	**(11137, 11137)**	(2230, **2236**)	(**2228**, 2245)
04a	**(2533, 2766)**	(2647, 2786)	**(11064, 11081)**	(11064, 11087)	**(2503, 2727)**	**(2503, 2727)**
05a	**(2263, 2792)**	(2426, 2852)	**(10941, 10981)**	(10941, 10988)	(2203, **2465**)	(**2194**, 2497)
06a	**(2202, 2767)**	(2328, 2769)	(10809, 10866)	**(10809, 10848)**	(2166, 2347)	**(2164, 2347)**
07a	**(2454, 2454)**	(2522, 2522)	**(16485, 16485)**	**(16485, 16485)**	**(2187, 2187)**	**(2187, 2187)**
08a	**(2186, 2426)**	(2276, 2465)	**(16485, 16485)**	**(16485, 16485)**	(2064, **2083**)	(**2062**, 2093)
09a	**(2168, 2297)**	(2254, 2458)	**(16485, 16485)**	**(16485, 16485)**	(2064, 2077)	**(2062, 2068)**
10a	(**2443**, 2986)	(2656, **2912**)	(16464, 16518)	**(16464, 16512)**	**(2178, 2607)**	(2178, 2629)
11a	**(2137, 2918)**	(2416, 2954)	(16135, 16231)	**(16135, 16194)**	(2031, 2375)	**(2021, 2328)**
12a	**(2036, 2319)**	(2339, 2745)	(15748, 15828)	**(15748, 15809)**	**(1974, 2082)**	(1974, 2115)
13a	**(2392, 2511)**	(2643, 2708)	**(21610, 21610)**	**(21610, 21610)**	(2216, 2223)	**(2203, 2215)**
14a	**(2297, 2405)**	(2493, 2522)	**(21610, 21610)**	**(21610, 21610)**	(2165, 2174)	**(2165, 2168)**
15a	**(2235, 2337)**	(2549, 2595)	**(21610, 21610)**	**(21610, 21610)**	(2165, 2178)	**(2164, 2173)**
16a	**(2409, 3040)**	(2735, 3236)	**(21478, 21556)**	(21478, 21562)	(2237, 2549)	**(2206, 2478)**
17a	**(2163, 2783)**	(2528, 3002)	**(20875, 20942)**	(20878, 20972)	(2105, 2347)	**(2093, 2268)**
18a	**(2109, 2442)**	(2416, 2901)	(**20562**, 20633)	(20566, **20621**)	(2068, **2171**)	(**2061**, 2198)

over 10 BRdata instances for 9 selected algorithms, and the rank accordance with the score of each algorithm is listed in the corresponding bracket. It is easily observed that MOMAD works well nearly on all the instances in terms of IGD and HV metrics since it achieves good performance on almost all the test problems.

Figures 5 and 6 show the final PFs of MK01~MK10 instances obtained by MOMAD and the reference PFs generated by selecting nondominated solutions from the mixture of eight compared algorithm. As can be seen, MOMAD finds all Pareto solutions for MK02. As for MK01, MK03, MK05, MK07, and MK08, MOMAD almost finds all Pareto



(a)



(b)

FIGURE 4: Ranking in the average performance score over MK01, MK02, MK03, and MK08 problem instances for the compared nine algorithms and over other six BRdata instances for the compared seven algorithms. The smaller the score, the better the overall performance in terms of HV and IGD metrics.

solutions. Besides, MOMAD can find the vast majority of the optimal solutions for MK04, MK06, MK09, and MK10. Thus, MOMAD is capable of finding a wide spread of PF for each instance and showing the tradeoffs among three different objectives.

Average CPU time consumed by MOMAD and other compared algorithms are listed in Table 13. However, the different experimental platform, programming language, and programming skills make this comparison not entirely reasonable. Hence, we enumerate the computational CPU time combined with original experimental platform and programming language for each corresponding algorithm, which help us distinguish the efficiency of the referred algorithms. The values show that the computational time consumed by MOMAD is much smaller than other algorithms except for Ka 4×5.

In summary, from the above-mentioned experimental results and comparisons, it can be concluded that MOMAD outperforms or at least has comparative performance to all the other typical algorithms when solving MOFJSP.

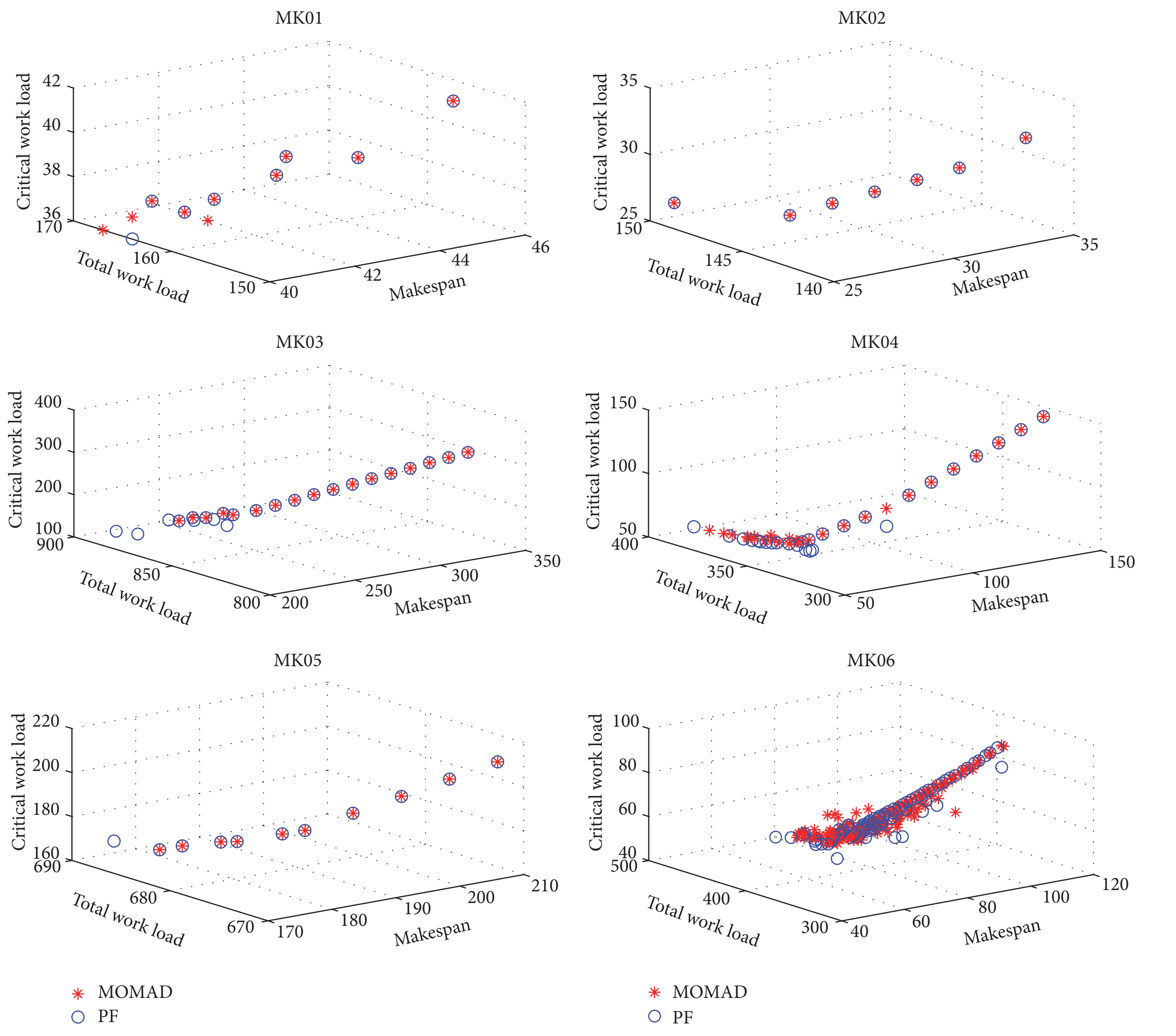


Figure 5: Final nondominated solutions of MK01~Mk06 instance found by MOMAD.

6.5. Impacts of Parameters Settings

(1) Impacts of Population Size N. Since population size N is an important parameter for MOMAD, we test the sensitivity of MOMAD to different settings of N for MK04. All other parameters except N are the same as shown in Table 2. The algorithm independently runs 10 times with each different N independently. As clearly shown in Table 14, the HV value changes weakly with the increasing of N. The IGD value decreases at first; then it will have a growing tendency. Totally speaking, MOMAD is not significantly sensitive to population size N, and properly increasing value of N can improve the algorithm performance to some extent.

(2) Impacts of Neighborhood Size T. T is another key parameter in MOMAD. With the aim of studying the sensitivity of T, MOMAD is implemented by setting different values of T and keeps other parameter settings unchanged. We run the MOMAD with each T 10 times independently for MK06. Table 14 shows how the IGD and HV values change along with the increasing of T from where we can obtain the same observations. Both values are enhanced first with the increasing of T; then the algorithm performance is degraded with the continuous increasing. So the influence of T on MOMAD is similar to N.

7. Conclusions and Future Work

This paper solves the MOFJSP in a decomposition manner for the purpose of simultaneously minimizing makespan, total workload, and critical workload. To propose an effective MOMAD algorithm, the framework of MOEA/D is adopted. First, a mixture of different machine assignment and operation sequencing rules is utilized to generate an initial population. Then, objective normalization technique

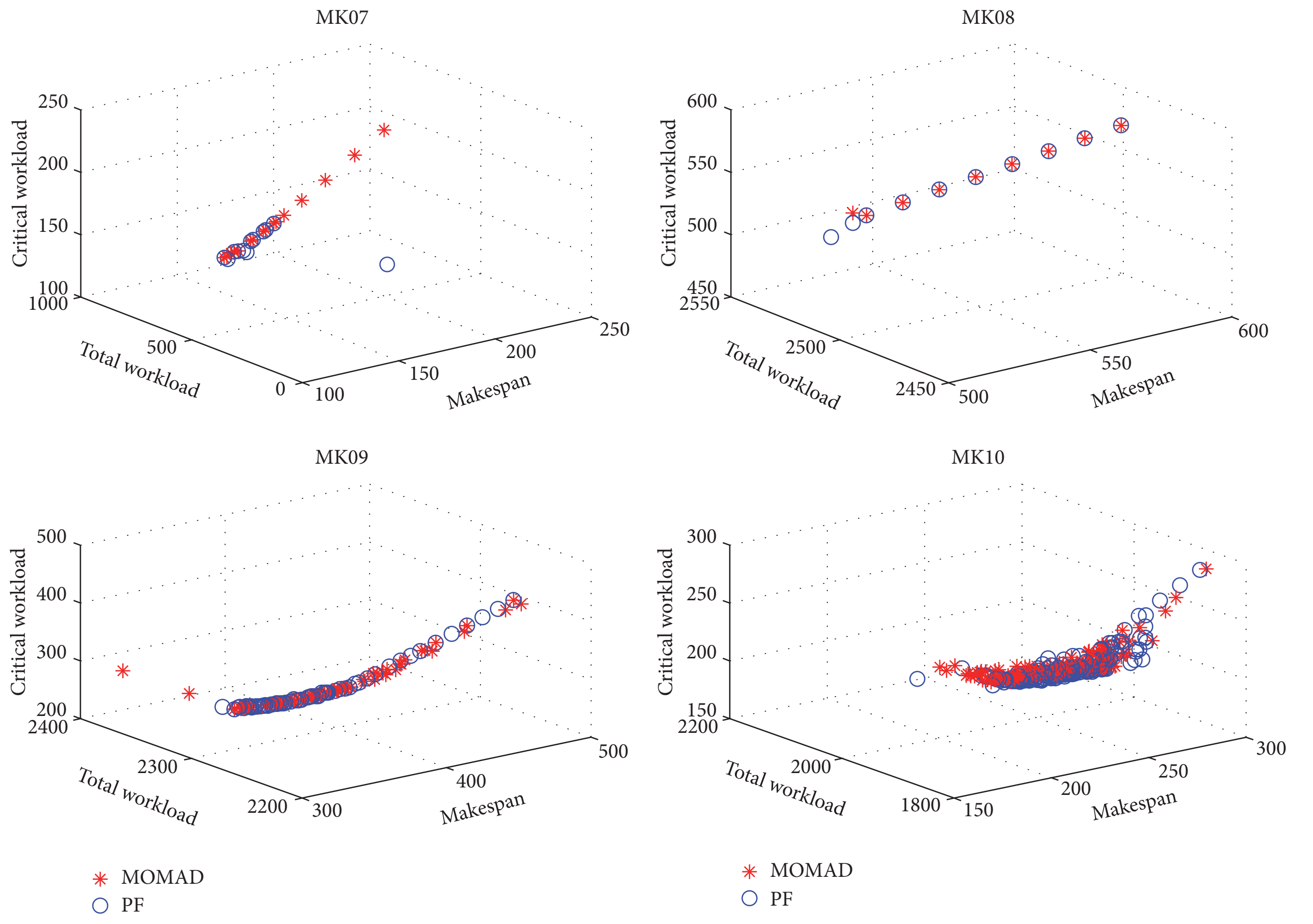


Figure 6: Final nondominated solutions of MK07~Mk10 instance found by MOMAD.

Table 13: The average CPU time (in seconds) consumed by different algorithms on Kacem and BRdata instances.

Instance	Algorithm					
	MOMAD[a]	MOGA[b]	HSFLA[c]	HMOEA[d]	hDPSO[e]	PRMOTS + IS[f]
ka 4 × 5	1.96	5.8	**1.26**	3.09	—	—
ka 8 × 8	**2.69**	9.5	—	9.69	8.2	—
ka 10 × 7	**3.78**	—	10.14	14.52	—	—
ka 10 × 10	**5.50**	14.2	—	14.41	11.7	—
ka 15 × 10	**15.28**	87.5	21.13	32.46	81.6	—
MK01	**7.02**	29.4	172.18	47.27	23.6	58.3
MK02	**13.73**	45.0	229.56	51.26	25.8	61.7
MK03	**22.28**	285.0	139.87	318.13	70.3	185.8
MK04	**16.84**	105.6	426.12	—	66.3	86.1
MK05	**16.93**	140.4	153.12	—	127.8	91.1
MK06	**59.05**	115.8	577.80	—	122.5	218.2
MK07	**30.02**	295.2	185.23	—	231.2	101.3
MK08	**32.93**	722.4	165.48	1674.14	147.1	560.6
MK09	**46.59**	1168.8	565.70	—	723.4	794.1
MK10	**97.58**	1072.2	1072.20	—	883.6	822.8

[a]The CPU time on an Intel Core i3-4170 CPU 3.7 GHz processor in C++. [b]The CPU time on a 2 GHz processor in C++. [c]The CPU time on a Pentium IV 1.8 GHz processor in C++. [d]The CPU time on an Intel Core(TM)2 Duo CPU 2.66 GHz processor in C♯. [e]The CPU time on a 2 GHz processor in C++. [f]The CPU time on an Intel Core TM2 T8100 CPU 2.1 GHz processor in C♯.

Table 14: IGD and HV metric corresponding to different N and T.

	MK04			MK06	
$N(H)$	IGD	HV	T	IGD	HV
55(8)	0.0465	1.1332	6	0.0448	0.9397
120(14)	0.0361	1.1380	12	0.0417	0.9547
153(16)	0.0437	1.1296	24	0.0402	0.9533
210(19)	0.0364	1.1338	48	0.0426	0.9463
253(21)	0.0326	1.1383	96	0.0487	0.9348
300(23)	0.0417	1.1336	120	0.0502	0.9360

is used in Tchebycheff approach, and the MOP is converted into many single-objective optimization subproblems. By clustering the weight vectors into different groups, a local exploitation based on moving critical operations is incorporated in MOEA/D and applied on candidate solutions with best aggregation function compared with solutions whose weight vectors belong to the same group. After embedding the local search into MOEA/D, our MOMAD is established. In simulation experiments, the Tchebycheff approach is studied to be more suitable for using in MOEA/D framework than WS and PBI approaches, and the effectiveness of local search is also verified. Moreover, MOMAD is compared with eight competitive algorithms in terms of three quantitative metrics. Finally, the effect of two key parameters, population size and neighborhood size, are analysed. Extensive computational results and comparisons indicate that the proposed MOMAD outperforms or at least has a comparative performance to other representative approaches, and MOMAD is fit to solve MOFJSP.

In the future, we want to study the MOFJSP with more than three objectives at first. Second, it would be significant to introduce a novel local search method which considers moving more than one critical operation. Finally, it would also be interesting to apply the MOMAD to dynamic scheduling problem.

Conflicts of Interest

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

Acknowledgments

This research work is supported by the National Science Foundation of China (no. 61572238) and the Provincial Outstanding Youth Foundation of Jiangsu Province (no. BK20160001).

References

[1] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. 117–129, 1976.

[2] M. Mastrolilli and L. M. Gambardella, "Effective neighbourhood functions for the flexible job shop problem," *Journal of Scheduling*, vol. 3, no. 1, pp. 3–20, 2000.

[3] G. Zhang, L. Gao, and Y. Shi, "An effective genetic algorithm for the flexible job-shop scheduling problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3563–3573, 2011.

[4] L. Wang, S. Wang, Y. Xu, G. Zhou, and M. Liu, "A bi-population based estimation of distribution algorithm for the flexible job-shop scheduling problem," *Computers & Industrial Engineering*, vol. 62, no. 4, pp. 917–926, 2012.

[5] Y. Yuan and H. Xu, "An integrated search heuristic for large-scale flexible job shop scheduling problems," *Computers & Operations Research*, vol. 40, no. 12, pp. 2864–2877, 2013.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[7] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[8] H. Li and Q. Zhang, "Multi-objective optimization problems with complicated pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[9] Q. Zhang, W. Liu, E. Tsang, and B. Virginas, "Expensive multiobjective optimization by MOEA/D with gaussian process model," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 456–474, 2010.

[10] Y.-Y. Tan, Y.-C. Jiao, H. Li, and X.-K. Wang, "A modification to MOEA/D-DE for multiobjective optimization problems with complicated Pareto sets," *Information Sciences*, vol. 213, pp. 14–38, 2012.

[11] Y. Yuan, H. Xu, B. Wang, B. Zhang, and X. Yao, "A new dominance relation-based evolutionary algorithm for many-objective optimization," *IEEE Transactons on Evolutionary Computation*, vol. 20, no. 1, pp. 16–37, 2016.

[12] Y. Yuan, H. Xu, B. Wang, B. Zhang, and X. Yao, "Balancing Convergence and Diversity in Decomposition-Based Many-Objective Optimizers," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 180–198, 2016.

[13] P. C. Chang, S. H. Chen, Q. Zhang, and J. L. Lin, "MOEA/D for flowshop scheduling problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '08)*, pp. 1433–1438, Hong Kong, June 2008.

[14] A. Alhindi and Q. Zhang, "MOEA/D with Tabu Search for multiobjective permutation flow shop scheduling problems," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014*, pp. 1155–1164, China, July 2014.

[15] X.-N. Shen, Y. Han, and J.-Z. Fu, "Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems," *Soft Computing*, pp. 1–24, 2016.

[16] F. Zhao, Z. Chen, J. Wang, and C. Zhang, "An improved MOEA/D for multi-objective job shop scheduling problem," *International Journal of Computer Integrated Manufacturing*, vol. 30, no. 6, pp. 616–640, 2017.

[17] W. Xia and Z. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems," *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 409–425, 2005.

[18] J. Gao, M. Gen, L. Sun, and X. Zhao, "A hybrid of genetic algorithm and bottleneck shifting for multiobjective flexible job shop scheduling problems," *Computers & Industrial Engineering*, vol. 53, no. 1, pp. 149–162, 2007.

[19] G. H. Zhang, X. Y. Shao, P. G. Li, and L. Gao, "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem," *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1309–1318, 2009.

[20] L.-N. Xing, Y.-W. Chen, and K.-W. Yang, "An efficient search method for multi-objective flexible job shop scheduling problems," *Journal of Intelligent Manufacturing*, vol. 20, no. 3, pp. 283–293, 2009.

[21] J.-Q. Li, Q.-K. Pan, and Y.-C. Liang, "An effective hybrid tabu search algorithm for multi-objective flexible job-shop scheduling problems," *Computers & Industrial Engineering*, vol. 59, no. 4, pp. 647–662, 2010.

[22] S. Wang, L. Wang, M. Liu, and Y. Xu, "An estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem," in *Proceedings of the 2013 IEEE Symposium on Computational Intelligence in Scheduling, CISched 2013 - 2013 IEEE Symposium Series on Computational Intelligence, SSCI 2013*, pp. 1–8, Singapore, April 2013.

[23] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic," *Mathematics and Computers in Simulation*, vol. 60, no. 3-5, pp. 245–276, 2002.

[24] N. B. Ho and J. C. Tay, "Solving multiple-objective flexible job shop problems by evolution and local search," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 38, no. 5, pp. 674–685, 2008.

[25] X. Wang, L. Gao, C. Zhang, and X. Shao, "A multi-objective genetic algorithm based on immune and entropy principle for flexible job-shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 51, no. 5-8, pp. 757–767, 2010.

[26] M. Frutos, A. C. Olivera, and F. Tohm, "A memetic algorithm based on a NSGAII scheme for the flexible job-shop scheduling problem," *Annals of Operations Research*, vol. 181, pp. 745–765, 2010.

[27] Y. Yuan and H. Xu, "Multiobjective flexible job shop scheduling using memetic algorithms," *IEEE Transactions on Automation Science and Engineering*, vol. 12, no. 1, pp. 336–353, 2015.

[28] J.-Q. Li, Q.-K. Pan, and J. Chen, "A hybrid Pareto-based local search algorithm for multi-objective flexible job shop scheduling problems," *International Journal of Production Research*, vol. 50, no. 4, pp. 1063–1078, 2012.

[29] T.-C. Chiang and H.-J. Lin, "A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling," *International Journal of Production Economics*, vol. 141, no. 1, pp. 87–98, 2013.

[30] J. Xiong, X. Tan, K. Yang, L. Xing, and Y. Chen, "A Hybrid Multiobjective Evolutionary Approach for Flexible Job-Shop Scheduling Problems," *Mathematical Problems in Engineering*, vol. 2012, pp. 1–27, 2012.

[31] L. Wang, S. Y. Wang, and M. Liu, "A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem," *International Journal of Production Research*, vol. 51, no. 12, pp. 3574–3592, 2013.

[32] S. Jia and Z.-H. Hu, "Path-relinking Tabu search for the multi-objective flexible job shop scheduling problem," *Computers & Operations Research*, vol. 47, pp. 11–26, 2014.

[33] G. Moslehi and M. Mahnam, "A Pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search," *International Journal of Production Economics*, vol. 129, no. 1, pp. 14–22, 2011.

[34] X. Shao, W. Liu, Q. Liu, and C. Zhang, "Hybrid discrete particle swarm optimization for multi-objective flexible job-shop scheduling problem," *The International Journal of Advanced Manufacturing Technology*, vol. 67, no. 9–12, pp. 2885–2901, 2013.

[35] L. C. F. Carvalho and M. A. Fernandes, "Multi-objective Flexible Job-Shop scheduling problem with DIPSO: More diversity, greater efficiency," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014*, pp. 282–289, China, July 2014.

[36] N. Tian and Z. Ji, "Pareto-ranking based quantum-behaved particle swarm optimization for multiobjective optimization," *Mathematical Problems in Engineering*, vol. 2015, Article ID 940592, 10 pages, 2015.

[37] J. Li, Q. Pan, and S. Xie, "An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems," *Applied Mathematics and Computation*, vol. 218, no. 18, pp. 9353–9371, 2012.

[38] L. Wang, G. Zhou, Y. Xu, and M. Liu, "An enhanced Pareto-based artificial bee colony algorithm for the multi-objective flexible job-shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 60, no. 9–12, pp. 1111–1123, 2012.

[39] Q. Zhang, H. Li, D. Maringer, and E. Tsang, "MOEA/D with NBI-style Tchebycheff approach for portfolio management," in *Proceedings of the 2010 6th IEEE World Congress on Computational Intelligence, WCCI 2010 - 2010 IEEE Congress on Evolutionary Computation, CEC 2010*, Spain, July 2010.

[40] Y. Mei, K. Tang, and X. Yao, "Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 151–165, 2011.

[41] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 42, no. 5, pp. 682–691, 2012.

[42] L. Ke and Q. Zhang, "Multiobjective combinatorial optimization by using decomposition and ant colony," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1845–1859, 2013.

[43] L. Ke and Q. Zhang, "Hybridzation of Decomposition and Local Search for Multiobjective Optimization," *IEEE Transactions on Cybernetics*, vol. 44, no. 44, pp. 1808–1819, 2014.

[44] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, vol. 32, no. 1, pp. 1–13, 2002.

[45] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point- based nondominated sorting approach, part I: solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.

[46] S.-K. Zhao, S.-L. Fang, and X.-J. Gu, "Machine selection and FJSP solution based on limit scheduling completion time minimization," *Jisuanji Jicheng Zhizao Xitong/Computer Integrated Manufacturing Systems, CIMS*, vol. 20, no. 4, pp. 854–865, 2014.

[47] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 10, pp. 3202–3212, 2008.

[48] 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," *Information Sciences*, vol. 289, pp. 76–90, 2014.

[49] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Annals of Operations Research*, vol. 41, no. 3, pp. 157–183, 1993.

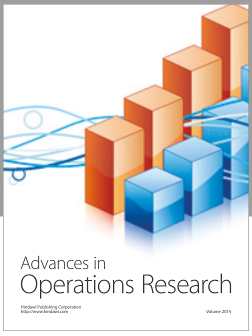
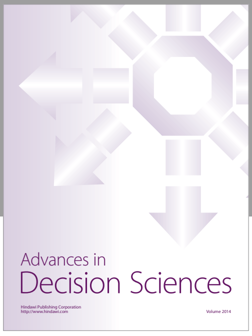
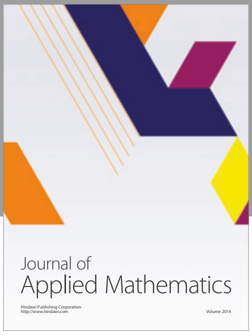
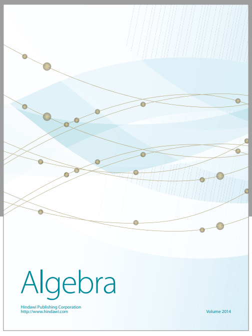
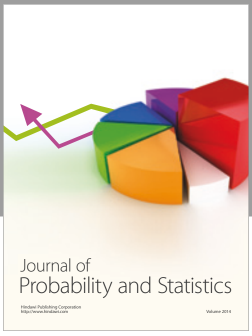
[50] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

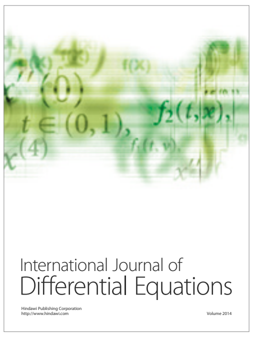
[51] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

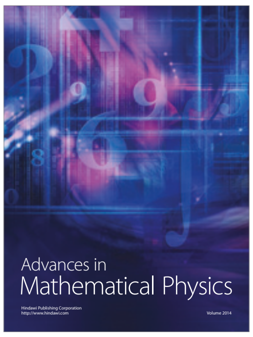
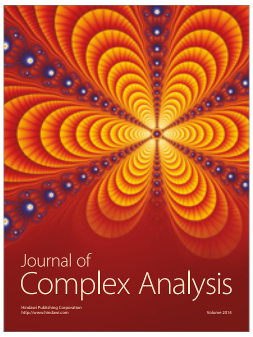
[52] J. Demsar, "Statistical comparisons of classifiers over multiple data sets," *Journal of Machine Learning Research*, vol. 7, pp. 1–30, 2006.

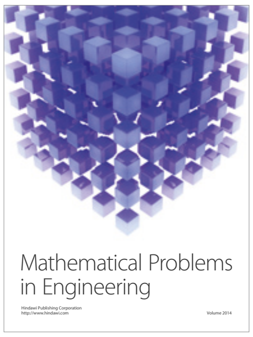
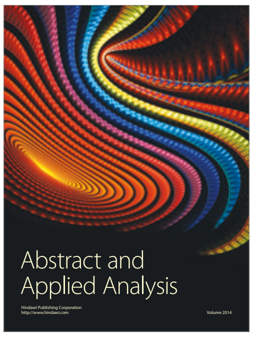
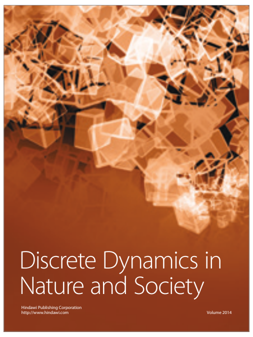
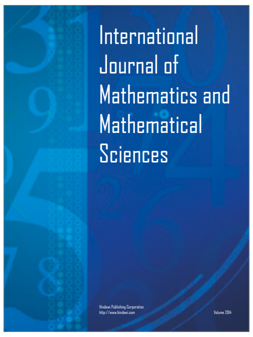
[53] S. Dauzère-Pérès and J. Paulli, "An integrated approach for modeling and solving the general multiprocessor job-shop scheduling problem using tabu search," *Annals of Operations Research*, vol. 70, pp. 281–306, 1997.

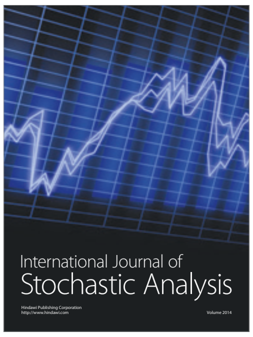
[54] J. Bader and E. Zitzler, "HypE: an algorithm for fast hypervolume-based many-objective optimization," *Evolutionary Computation*, vol. 19, no. 1, pp. 45–76, 2011.





Hindawi

Submit your manuscripts at
https://www.hindawi.com