Flexible Job-Shop Rescheduling for New Job Insertion by Using Discrete Jaya Algorithm

Kaizhou Gao, *Member, IEEE*, Fajun Yang, MengChu Zhou, *Fellow, IEEE*, Quanke Pan, and Ponnuthurai Nagaratnam Suganthan, *Fellow, IEEE*

***Abstract*—Rescheduling is a necessary procedure for a flexible job shop when newly arrived priority jobs must be inserted into an existing schedule. Instability measures the amount of change made to the existing schedule and is an important metrics to evaluate the quality of rescheduling solutions. This paper focuses on a flexible job-shop rescheduling problem (FJRP) for new job insertion. First, it formulates FJRP for new job insertion arising from pump remanufacturing. This paper deals with bi-objective FJRPs to minimize: 1) instability and 2) one of the following indices: a) makespan; b) total flow time; c) machine workload; and d) total machine workload. Next, it discretizes a novel and simple metaheuristic, named Jaya, resulting in DJaya and improves it to solve FJRP. Two simple heuristics are employed to initialize high-quality solutions. Finally, it proposes five objective-oriented local search operators and four ensembles of them to improve the performance of DJaya. Finally, it performs experiments on seven real-life cases with different scales from pump remanufacturing and compares DJaya with some state-of-the-art algorithms. The results show that DJaya is effective and efficient for solving the concerned FJRPs.**

***Index Terms*—Ensemble, instability, Jaya, job shop scheduling, remanufacturing, rescheduling.**

I. Introduction

FLEXIBLE job-shop scheduling problem (FJSP) is an extension of the classical job-shop scheduling problem and includes two subproblems, machine assignment and operation sequencing [1], [2]. The former is to select a machine from a candidate set for each operation while the latter is to schedule all operations on all machines to obtain satisfactory schedules. FJSP is very complicated and has been proven to be an NP-hard problem [3], [4].

FJSP arises from many industrial sectors, such as mechanical manufacturing, remanufacturing, semiconductor manufacturing, and automobile assembly and disassembly [5]. Constraints exist and are considered in real-life scheduling problems [6]. Existing literature has considered many constraints and uncertainty related issues. Wang *et al.* [7], [8] studied FJSP with fuzzy processing time using both artificial bee colony (ABC) algorithm and estimation of distribution algorithm (EDA). The influence of parameter settings was considered in both of them. A left-shift scheme was innovatively employed for improving scheduling solutions in a decoding stage. Crossover-based exploitation and variable neighborhood search were employed for improving the performance of the algorithms. For the same problem, our prior work [9] proposed a discrete harmony search (HS) algorithm and compared it with several existing algorithms for minimizing the maximum completion time. Xiong *et al.* [10] researched multiobjective FJSP with random machine breakdowns. Two surrogate measures for its solutions' robustness were developed. One was for machine breakdown and another for both location of float time and machine breakdown. Different from [10], Ahmadi *et al.* [11] used nondominated sorting genetic algorithm II (NSGA-II) and nondominated sorting genetic algorithm. Both stability and makespan are optimized after machine breakdown. Li *et al.* [12] solved FJSP with maintenance activities by using an ABC algorithm. A self-adaptive strategy was proposed to generate new neighboring solutions and Tabu search was employed as local search to improve the scheduling performance. The work compared their proposed algorithm with ten existing algorithms to verify its effectiveness. The work [13], [14] studied resource-constrained FJSP by using a fruit fly optimization algorithm and migrating birds optimizer, respectively. Karimi *et al.* [15] modeled FJSP with transportation time and employed an imperialist competitive algorithm with a simulated annealing-based local search operator to minimize makespan.

FJSP has been studied in a remanufacturing environment. Junior and Filho *et al.* [16] reviewed the studies on production planning and control in remanufacturing. Seventy-six papers were examined and classified. However, there is limited work on reprocessing scheduling in remanufacturing. New job insertion is one of seven major complicating characteristics in

Manuscript received June 29, 2017; revised January 22, 2018; accepted March 14, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61603169, Grant 61773192, and Grant 61503170, and in part by the Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology. This paper was recommended by Associate Editor L. Tang. *(Corresponding author: Mengchu Zhou.)*

K. Gao is with the School of Computer, Liaocheng University, Liaocheng 252059, China (e-mail: gaokaizh@aliyun.com).

F. Yang and P. N. Suganthan are with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798 (e-mail: fjyang@ntu.edu.sg; epnsugan@ntu.edu.sg).

M. Zhou is with the Institute of Systems Engineering, Macau University of Science and Technology, Macau 999078, China, and also with the Department of Electrical and Computer Engineering, New Jersey Institute of Technology, Newark, NJ 07102-1982 USA (e-mail: zhou@njit.edu).

Q. Pan is with the State Key Laboratory of Digital Manufacturing Equipment and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: panquanke@qq.com).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2018.2817240

remanufacturing [17], [18]. In a real-life shop floor, rescheduling is necessary after new job insertion. FJRP is modeled to handle a rescheduling problem in pump remanufacturing. Instability is an important criterion to evaluate the quality of rescheduling solutions [11]. Hence, how to minimize the instability in FJRP after new job insertion is a significant topic and remains open.

To solve optimization problems, especially job shop and flow shop scheduling problems, meta-heuristics are considered as a new trend in recent years. They include genetic algorithm (GA) [19], particle swarm optimization (PSO) [20], ABC [21] and HS [22]. In this paper, a simple and novel metaheuristic, named Jaya, is first discretized and then implemented to solve FJRP for the first time. Jaya is a newly developed metaheuristic [23]. Comparing with most existing metaheuristic algorithms, its advantage is that it just requires two common parameters, i.e., population size and number of iterations, and does not require to adjust any algorithm-specific parameters. It makes it easier to apply to solve real-life discrete combination optimization problems, e.g., flexile job shop scheduling and rescheduling problems. In [23], Jaya has been compared with several metaheuristics, e.g., GA, DE, PSO, ABC, and teaching-learning-based-optimization. The experimental results have verified its competitiveness. In view of its simplicity and effectiveness, Jaya has been employed for dimensional optimization of a microchannel heat sink [24], design optimization [25], welding processing optimization [26], multiobjective optimization of modern machining processes [27], economic optimization of shell-and-tube exchange [28], surface grinding process optimization [29], and traffic light control and optimization in urban traffic network [22]. These studies further verified its effectiveness and efficiency for solving real-life problems.

In this paper, the discrete version of Jaya, called DJaya for short, is developed for tackling FJRP. Some simple heuristics are employed to initialize the population and to obtain initial solutions with high quality. Moreover, based on FJRP features, several objective-oriented local search operators and their ensembles are proposed to improve its performance. This paper presents seven real-life cases with different scales from pump remanufacturing and their solutions to examine its performance and compare it with its peers.

The remainder of this paper is organized as follows. Section II describes the mathematical model of FJRP. The standard Jaya algorithm is introduced in Section III. In Section IV, DJaya is described. Section V presents experimental setup, comparisons, and discussions of the obtained optimization results. Finally, Section VI concludes this paper.

II. Modeling of FJSP and FJRP

A. FJSP Model

In a flexible job shop, each job consists of a sequence of operations. An operation requires only one out of candidate machines. It must be processed on only one machine at a time, while each machine can handle only one operation at a time. The following notations and assumptions are needed to formulate a multiobjective FJSP.

1) Let $J = \{J_i\}$, $1 \le i \le n$, be a set of n jobs to be scheduled. q_i denotes the total number of operations of job i.
2) Let $M = \{M_k\}$, $1 \le k \le m$, be a set of m machines.
3) Each job J_i consists of a predetermined sequence of operations. Let $O_{i,h}$ be operation h of J_i.
4) Each operation $O_{i,h}$ can be performed without interruption on one in $M(O_{i,h})$ that is a set of machines able to perform $O_{i,h}$. Let $P_{i,h,k}$ be the processing time of $O_{i,h}$ on machine M_k.
5) Decision variables

$$x_{i,h,k} = \begin{cases} 1, & \text{if machine } k \text{ is selected for operation } O_{i,h} \\ 0, & \text{otherwise} \end{cases}$$

where the completion time of operation $O_{i,h}$ is denoted as $c_{i,h}$.
6) The objectives considered in this paper are to minimize:
   a) the maximum completion time of all jobs (Makespan) $C_{\text{Max}} = \max_{1 \le i \le n} c_i$, where c_i is the completion time of job J_i;
   b) total flow time, i.e., the total completion time of all jobs $C_{\text{Total}} = \sum_{1 \le i \le n} c_i$, where c_i is the completion time of job J_i;
   c) maximum machine workload, denoted by $W_{\text{Max}} = \max_{1 \le j \le m} w_j$, where w_j is the workload of machine M_j; and
   d) total machine workload, denoted by $W_{\text{Total}} = \sum_{1 \le j \le m} w_j$, where w_j is the workload of machine M_j.

B. Rescheduling and Instability Metrics for New Job Insertion

New job insertion is frequent in a remanufacturing environment [18], [31]. A rescheduling operator is necessary when a new job is inserted (at time t) into an existing schedule that is being executed on a shop floor. For existing jobs, the start time for rescheduling is the insertion time or the completion time of the current processing operation. For each machine, the start time for rescheduling is the insertion time or the completion time of the on-going operation on this machine. Hence, FJRP is FJSP with different job and machine start time. The start time of existing job i (S_{J_i}) and machine k (S_{M_k}) in rescheduling can be defined as follows:

$$S_{J_i} = \begin{cases} t, & \text{No operation of job } i \text{ is being processed at } t \\ c_{i,h}, & \text{Operation } O_{i,h} \text{ is being processed at } t \end{cases}$$

$$S_{M_k} = \begin{cases} t, & \text{No operation is being processed on } M_k \text{ at } t \\ c, & \text{Operation } O \text{ is being processed on } M_k \text{ at } t. \end{cases}$$

In a rescheduling phase, the flow time of a new job is calculated based on its arriving time and completion time. To explain FJRP for job insertion more clearly, we give an example shown in Fig. 1. Fig. 1(a) shows the result with no rescheduling after inserting J_4 directly. An existing scheduling scheme is retained and J_4 is scheduled when the last operation on each machine is completed. Fig. 1(b) shows a rescheduling solution. Both J_4 and all nonstarted operations of existing jobs are rescheduled when J_4 is inserted at Time 3. As a result,

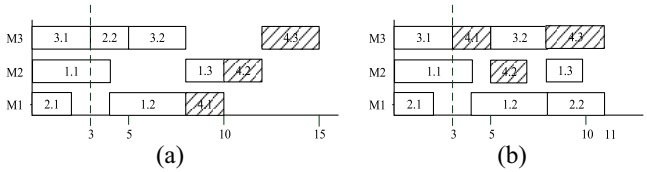


Fig. 1. Example of new job insertion. (a) Shows the result with no rescheduling after inserting new job while (b) shows a rescheduling result.

rescheduling is able to save four time units, representing great time saving for this example case.

Instability is defined as the percentage of changed operations of the existing jobs on their respective processing machines during rescheduling. Fewer operation changes mean the lower instability or higher stability of rescheduling. For example, in Fig. 1(a), all the operations of existing jobs remain on their original machines, the operations of J_4 are scheduled after finishing the existing operations. Fig. 1(a) shows the lowest instability of rescheduling. In Fig. 1(b), makespan is 11 which is less than 15 in Fig. 1(a). Operation $O_{2,2}$ is moved from M_3 to M_1 and the processing time becomes larger. The instability of rescheduling in Fig. 1(b) is higher than that in Fig. 1(a). To describe it more clearly, instability is defined as follows:

$$F = 100 \times \frac{\sum_{i=1}^{n} \sum_{h=1}^{q_i} x_{i,h}}{\sum_{i=1}^{n} q_i}\%$$

$$x_{i,h} = \begin{cases} 0, & M_k \text{ remains for } O_{i,h} \\ 1, & M_k \text{ is changed for } O_{i,h} \end{cases}$$

where q_i is the operation count of job i in rescheduling and M_k is the machine for operation $O_{i,h}$. The objective is to minimize the instability of rescheduling solutions. The job count n here is the number of existing jobs before new job insertion. It is clear that smaller instability value means the better stability of a rescheduling solution.

C. Four Bi-Objective Functions

A multiobjective optimization problem can be stated as follows:

$$\text{Min} \quad F(X) = (f_1(x), f_2(x), \ldots, f_m(x))^T$$
$$\text{Subject to} \quad X \in \Omega$$

where the decision vector $x = (x_1, x_1, \ldots, x_n)$ belongs to the decision space Ω. The objective function vector $F : \Omega \to \wedge$ consists of multiple objectives. In this paper, functions f_1 to f_4 are set as the four objectives mentioned in Section II-A while the instability of a rescheduling solution is set as the fifth function f_5

$$f_1 : C_{\text{Max}} = \max_{1 \le i \le n} c_i$$
$$f_2 : C_{\text{Total}} = \sum_{1 \le i \le n} c_i$$
$$f_3 : W_{\text{Max}} = \max_{1 \le j \le m} w_j$$
$$f_4 : W_{\text{Total}} = \sum_{1 \le j \le m} w_j$$
$$f_5 : F = 100 \times \frac{\sum_{i=1}^{n} \sum_{h=1}^{q_i} x_{i,h}}{\sum_{i=1}^{n} q_i}\%.$$

Bi-objective functions related to instability f_5 and one of other objectives f_1 to f_4 are defined as follows:

$$\text{Min} \quad F_i(X) = (f_i(x), f_5(x))^T$$
$$1 \le i \le 4.$$

Four bi-objective functions are considered, where the instability objective is optimized with f_1 to f_4 one by one. $F_1(X)$ is the bi-objective function for makepsan and instability. $F_2(X)$ is the one for total flow time and instability. $F_3(X)$ and $F_4(X)$ are the ones for the maximum machine workload and instability, and total machine workload and instability, respectively. Here, a widely used strategy, Pareto domination, is employed to compare and rank solutions for bi-objective functions. For two solutions $x = (x_1, x_1, \cdots, x_n)$ and $x' = (x'_1, x'_2, \ldots, x'_n)$, x domimates x' (denotes as $x \prec x'$) if and only if $\forall p \in \{i, 5\},\ 1 \le i \le 4, f_p(x) \le f_p(x')$ and $\exists q \in \{i, 5\}, 1 \le i \le 4,\ f_q(x) < f_q(x')$. Solution x is an optimal solution in the Pareto set if there is no solution x' that dominates x. The Pareto optimal set is the collection of all Pareto optimal solutions and the corresponding image in the objective space is the Pareto front (PF). In this paper, an archive set (AS) is used to record the nondominated solutions during the iterations. During the search process in the following algorithms, if a new solution dominates one or more solutions in AS, the new solution is used to replace the dominated ones. As FJSP has been proven to be an NP-hard problem, the flexible job shop rescheduling problem in this paper is also NP-hard. The computational complexity becomes very high as the problem size increases.

III. Jaya Algorithm

The definition of Jaya is victory in Sanskrit. In Jaya, the applied strategy always tries to become victorious by reaching the best solution and hence it is named Jaya, and it is reported as a simple and applicable optimization approach [23]. Its first step is to form random initial population with the size N_P. Next, the best and worst solutions are extracted from the initial population and are used for updating other solutions for the next iteration. The best and worst solutions have to be updated at each iteration. The simple movement strategy (i.e., updating equation) for a new solution is defined by the following equation:

$$X_i(t+1) = X_i(t) + r_1 \times (X_B(t) - X_i(t)) + r_2 \times (X_W(t) - X_i(t)),\ i = 1, 2, \ldots, N_P$$

where $X_i(t+1)$ is the updated position of $X_i(t)$ (i.e., current position), $X_B(t)$ and $X_W(t)$ are the best and worst solutions extracted from the current population, respectively. r_1 and r_2 are two uniformly distributed random numbers in range [0, 1] and t is an iteration index. For a selection purpose, if the objective function value of $X_i(t+1)$ is better than that of $X_i(t)$, then $X_i(t+1)$ is accepted to replace $X_i(t)$. All the accepted solutions at the end of each iteration are maintained and these values become the input population for the next iteration.

It is worth mentioning that Jaya does not require any initial parameter except population size and the maximum number

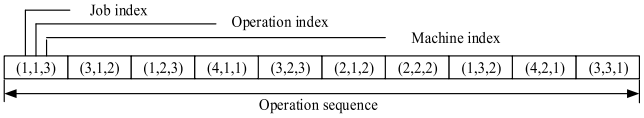


Fig. 2. Example for encoding.

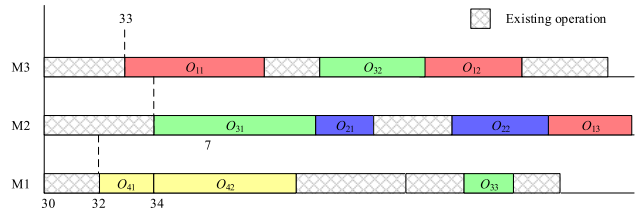


Fig. 3. Example for rescheduling solution decoding.

of function evaluations. Note that both of them are considered as common user parameters as required by all population-based intelligent optimization methods. The flow chart of Jaya algorithm is shown in the supplementary material.

IV. Proposed Discrete Jaya Algorithm

A. Encoding and Decoding

In the proposed discrete Jaya algorithm (DJaya), one candidate solution should include both machine assignment and operation sequence for FJRP. A simple example candidate solution is shown in Fig. 2 to describe an encoding strategy. There are four jobs, three machines, and ten operations. Each element in the solution includes three values, which are job, operation, and machine indices. For example, (1, 1, 3) means that the first operation of job 1 is processed on machine 3. The total number of elements (which is 10 in this case) is the total number of operations. In this encoding strategy, both operation sequence and machine assignment are considered at the same time. When a candidate solution is encoded to a Gantt chart, there is a one-to-one mapping between the elements in the solution and the operations in the Gantt chart. If the four jobs are new jobs inserted into the existing schedule at time 30 and the start time of three machines for rescheduling are 32, 34, and 33, respectively, the new jobs and the nonstarted operations of existing jobs are rescheduled and the rescheduling solution can be decoded to a Gantt chart as shown in Fig. 3.

B. Population Initialization

In meta-heuristics, the quality of initial population often affects the speed of convergence to a satisfactory solution. Therefore, it is a critical step to generate a high-quality initial population. To initialize the population of DJaya, a random rule [32], [33] and two simple heuristics are employed, which are related to the completion time and machine workload objectives. The first one is global minimum-processing time rule [32] and the second one is the minimum-completion time rule [33]. The detailed procedures of the two simple heuristics are shown as follows.

1) Global Minimum-Processing Time Rule: This rule considers both the processing time and workload of machines, and starts from the operation with the global minimum processing. The processing time is added to the machine workload. For other operations, the machine with minimum processing time (workload) is fixed and assigned. The machine's workload update is also performed. This rule considers the global workload among all machines and can potentially find a schedule with small makespan.

2) Minimum-Completion Time Rule:

Step 1: For each operation O_{ij}, select machine M_1 with the minimum processing time and M_2 with the earliest feasible time from the selectable machine set.

Step 2: Calculate the completion time T_1 and T_2 of operation O_{ij} on machines M_1 and M_2.

Step 3: If $T_1 < T_2$, M_1 is selected for processing operation O_{ij}; otherwise, M_2 is selected.

C. Discretization Strategy

Jaya was initially proposed for solving continuous optimization problems (real values). FJRP is a discrete combinatorial optimization problem. Based on the encoding strategy in Section IV-A, a discretization procedure is developed to make Jaya applicable for handling discrete optimization problems. There are three binary random numbers (i.e., 0 or 1) considered for the selection purpose. The following updating equation is for DJaya:

$$X_i'(t+1) = b_1 X_i(t) + b_2 X_B(t) + b_3 X_W(t), \quad i = 1, 2, \ldots, N_P$$
$$\text{with } b_1, b_2, b_3 \in \{0, 1\} \text{ and } \sum_{j=1}^{3} b_j = 1.$$

As can be seen from the above equations, there is a chance for one and only one item to be selected depending on the values of random binary numbers (b_1, b_2, and b_3) for a new solution. At the same time, when one operation is selected from X_i, X_B, or X_W, it should be checked that this operation is not selected already to make sure no repetition of a solution. The complexity of DJaya is the same as that of the Jaya algorithm while DJaya can be easily used for FJRP.

D. Objective-Oriented Ensemble of Local Search Operators

According to the encoding strategy shown in Section IV-A, an operation sequence would be updated or improved in each iteration. Next, to improve the exploitation performance of DJaya for machine assignment, five objective-oriented local search operators are proposed for makespan, the maximum machine workload, total flow time, total machine workload, and instability. Considering the computing complexity, we no longer perform them once the concerned objective is improved. The five local search operators are presented as LS1–LS5.

To match the four bi-objective functions in Section II-C, four ensembles are developed by integrating the local search operator for instability (**LS5**) and other local search operators (**LS1** to **LS4**) one by one. The detailed procedures of the ensembles are shown in Ensembles.

LS1: Local Search Operator for Makespan

For a solution X, compute C_{Max}, find the final completed operation $O_{i,h}$ on machine M
If $O_{i,h}$ has more than one candidate machine
a. Move $O_{i,h}$ to a different machine M' and find a suitable position to obtain a new solution X'
b. If the makespan of X' is less than C_{Max}
c. X' replaces X and update the makespan
d. Endif
Else
Do
Find the previous operation O' on machine M and the previous operation $O''$of job i.
If the completion time of O' and/or O'' is the same as the start time of $O_{i,h}$
For O' and/or O'', execute Steps a to d.
Endif
While (existing O' and/or O'')
Endif

LS2: Local Search Operator for the Maximum Machine Workload

For a solution X, compute workloads for all machines
Select machine M with the maximum workload W_{Max}
For each operation $O_{i,h}$ on machine M
If $O_{i,h}$ has more than one candidate machine,
Move $O_{i,h}$ to a different machine M' and obtain a new solution X'
If the current workload of $M'$is less than W_{Max}
X' replaces X and update the maximum workload
Endif
Endif
Endfor

LS3: Local Search Operator for Total Flow Time C_{Total}

For a solution X, compute C_{Total}, $h = q_i$
For $1 \leq\ i \leq n$
If $O_{i,h}$ has more than one candidate machines
Move $O_{i,h}$ to a different machine M' and obtain a new solution X'
If C'_{Total} of X' is less than C_{Total}
X' replaces X and update C_{Total}
Else
$i = i + 1$
If $i > n$
$i = 1, h = h - 1$
Endif
Endif
Endif
Endfor

LS4: Local Search Operator for Total Machine Workload W_{Total}

For a solution X, select one operation $O_{i,h}$ on machine M randomly
If $O_{i,h}$ has more than one candidate machine
Select the machine M' with the minimum processing time
If $M' \neq M$
Move $O_{i,h}$ to M' and update W_{Total}
Endif
Else
Execute above operators for other operations on M
Endfor

LS5: Local Search Operator for Instability

For a solution X, select one operation $O_{i,h}$ on machine M randomly
If $O_{i,h}$ has more than one candidate machine
Find machine M' before rescheduling
If $M' \neq M$
Move $O_{i,h}$ to M'
Endif
Else
Execute above operators for other operations on M.
Endfor

Ensembles

Select one solution X from population
Execute **LS5** to obtain a new solution X'
Execute one local search operator out of **LS1-LS4** to get a new solution X''
Check the domination relationship among X, X', and X''
If $X' \prec X,\ X'' \prec X$
Randomly select X'or X'' to replaces X and put another one into AS if it is not dominated by those in AS
Else
If $(X' \prec X,\ X \prec X'')$ or $(X \prec X',\ X'' \prec X)$
Select the one dominated X' or X'' to replaces X
Else
If X, X', and X'' cannot dominate each other
Randomly select X' or X'' to replace X and put the other two into AS if they are not dominated by those in AS
Endif
Endif
Endif

E. Framework of DJaya

The ensembles are embedded in DJaya in each iteration to improve the local search performance for the corresponding bi-objectives. Limited by the paper length, the framework of DJaya with ensembles is shown in the supplementary material.

In DJaya, the solutions are updated based on the domination strategy described in Section II-C. If new solutions and the current ones in population are nondominated with each other, the former is used to update AS.

V. Experiments and Discussions

A. Experimental Setup

To evaluate the performance of the proposed DJaya for solving FJRP, we perform computer experiments and analyze the results. Seven cases with different scales are solved, which

come from the real-life orders of a remanufacturing enterprise in Singapore. The scales of the cases are from 8 jobs, 8 machines, and 64 operations to 20 jobs, 15 machines, and 355 operations. There are 14 new jobs inserted into an existing schedule of seven instances. Each insertion has different inserting time, job count, and operation count. The detailed information of all cases is shown in the first table in the supplementary material.

We first test the validity of the five local search operators. Then, DJaya is compared with three well-known multiobjective optimization algorithms, i.e., NSGA-II [34], MOEA/D [35], and MOABC [36]. All algorithms have been coded in C++ and run on an Intel 3.40 GHz PC with 8 GB memory. For all algorithms, the population size and maximum iteration count are set to 50 and 1000, respectively, for the sake of having fair comparisons. The parameters' setting of three compared algorithms is the same as those in the corresponding literature. All experiments are carried out in 30 replications.

A widely-used performance indicator, inverted generational distance (IGD) [35] is used to evaluate the quality of nondominated solutions by all compared algorithms. The purpose is to find nondominated solutions with high quality. For each algorithm, there is a set of nondominated solutions in 30 runs. The actual PF is unknown for FJRP. Here, its approximation is obtained by comparing nondominated solutions by all four algorithms. Let P^* be the set of uniformly distributed points in PF while P is the set of nondominated solutions obtained via four algorithms. IGD is defined as follows:

$$\mathrm{IGD}(P^*, P) = \frac{\sum_{v \in P^*} d(v, P)}{|P|}$$

where $d(v, P)$ is the minimum Euclidean distance between the objective v and the points in P. To have a low IGD value, P must be very close to PF.

PF is obtained by comparing all nondominated solutions of all algorithms. One algorithm has better performance if this algorithm can find a larger number of solutions in PF. Hence, the proportion of the solutions in PF found by the ith algorithm is evaluated by the following equation:

$$P^i = \frac{N^i}{N^A} \times 100$$

where N^A is the number of solutions in PF, N^i is the number of solutions in PF which are found by the ith algorithm. It is obvious that larger P^i means better performance.

B. Validity Test on Five Local Search Operators

In order to test the validity of the five proposed local search operators, one side t-test is employed. The null hypothesis is that DJaya with and without local search operators has the same performance, and the alternative is that DJaya with a local search operator has better performance than that without it. In order to save space, we only record the number of pairwise comparison results, as shown in Table I.

Table I lists the results of t-test of DJaya with and without five local search operators for seven cases. For all the statistic tests, we set the level of significance as 0.05. The symbol "+," "=," and "−" indicate that DJaya with a local search operator performs significantly better than, almost the same as, and significantly worse than DJaya it. From Table I, by t-test, we can conclude that DJaya with each local search operator performs better than DJaya without it. In seven cases, the number of times that DJaya with a local search operator is better than DJaya without it is 6, 5, 6, 6, and 5, respectively, for five local search operators. For the small-scale case (case 1, with 8 job, 8 machines, and 64 operations), DJaya with and without local search operators can both obtain the optimal results. With the increasing problem size, the effect of local search operators is shown more clearly. There is no case for which DJaya with a local search operator is worse than DJaya without it. Hence, the five local search operators are effective to improve the performance of DJaya.

TABLE I
ONE SIDE t-TEST COMPARISON FOR DJAYA WITH AND WITHOUT LOCAL SEARCH

Local search	+ (Better)	= (Same)	- (Worse)
LS 1	6	1	0
LS 2	5	2	0
LS 3	6	1	0
LS 4	6	1	0
LS 5	5	2	0

TABLE II
IGD VALUES FOR MAKESPAN VERSUS INSTABILITY

Case	NSGAII	MOEA/D	MOABC	Djaya
1	0.00	0.00	0.00	0.00
2	0.16	0.09	0.03	0.00
3	0.26	0.23	0.22	0.00
4	0.47	0.24	0.21	0.00
5	0.93	0.88	0.41	0.00
6	0.81	0.76	0.49	0.00
7	1.56	1.07	0.59	0.00

C. Bi-Objective for Instability and Completion Time

This section discusses the results about instability and the completion time-related objectives, i.e., F_1 and F_2. The nondominated solutions by four algorithms for F_1, makespan and instability, are recorded in the second table in the supplementary material. The results about IGD and proportion for F_1 are shown in Tables II and III, where "N" is the number of dominated solutions by the corresponding algorithms. The IGD and proportion results for F_2 are presented in Tables IV and V while the nondominated solutions by four algorithms are presented in the third table in the supplementary material.

First, the results about F_1, makespan and instability, are discussed. For case 1, four algorithms find the same results. To show the differences among four algorithms clearly, the nondominated results for F_1 are presented in Fig. 4. Since the results of case 1 are the same by all algorithms, Fig. 4 just shows the results for cases 2–7. For case 2, DJaya obtains

TABLE III
PROPORTION RESULTS FOR MAKESPAN VERSUS INSTABILITY

Case	PF	NSGAII		MOEA/D		MOABC		DJaya	
	N^A	N	P^i %	N	P^i %	N	P^i %	N	P^i %
1	3	3	100	3	100	3	100	3	100
2	3	4	0.0	4	33.3	4	50.0	3	100
3	3	4	0.0	4	33.3	3	33.3	3	100
4	5	6	0.0	4	0.0	5	0.0	5	100
5	5	5	0.0	5	0.0	4	0.0	5	100
6	3	4	0.0	5	0.0	4	0.0	3	100
7	3	4	0.0	3	0.0	2	0.0	3	100

three nondominated solutions, all of which are optimal solutions. Other algorithms cannot find optimal solution or just find a portion of optimal solutions. DJaya has better performance than NSGA-II, MOEA/D, and MOABC for case 2. For case 3, DJaya can find five optimal solutions while MOEA/D and MOABC can just find one. NSGA-II cannot find any optimal solution. For case 4, DJaya can find five optimal solutions while the three peers cannot any optimal solution. With an increasing scale, the competitiveness of DJaya becomes more and more obvious, especially for cases 5–7.

According to Table II, the IGD results from four algorithms for case 1 are the same (0.00) since four algorithms obtain the same nondominated solutions and these solutions are also in PF. For cases 2–7, the IGD results via DJaya are smallest (0.00) while the results via NSGA-II, MOEA/D, and MOABC are larger than zero. It means that all nondominated solutions obtained by DJaya are Pareto optimal in PF. Hence, DJaya has the best IGD results among four algorithms for the bi-objective problem with makespan and instability.

It can be seen from Table III that the proportion values of NSGA-II are nonzero (100) for case 1 and zero for cases 2–7. It means that NSGA-II can just find Pareto optimal solutions for case 1. MOEA/D and MOABC can obtain Pareto optimal solutions for cases 1 and just find a portion of Pareto optimal solutions for cases 2 and 3 since the proportion values of these two algorithms are less than 100, {(33.3, 33.3), (50.0, 33.3)}. The proportion value of DJaya is 100 for seven cases, which means that DJaya can find all Pareto optimal solutions in PF.

Next, the results about F_2, total flow time and instability, are discussed. Since the total flow time is the sum of the completion time of all jobs, the difference and gap among four algorithms become more obvious. For the smallest case, case 1, DJaya obtains all optimal solutions while the three peers can find some but not all the optimal solutions. The difference among four algorithms is not very obvious. As the cases become more and more complex, the difference and gap among four algorithms become larger and larger. To show the differences among nondominated solutions from four algorithms, the results for cases 2–7 are presented in Fig. 5. It can be seen from Fig. 5 that DJaya obtains the best nondominated solutions among four algorithms for cases 2–7. For example, DJaya obtains three nondominated solutions for

TABLE IV
IGD VALUES FOR TOTAL FLOW TIME VERSUS INSTABILITY

Case	NSGAII	MOEA/D	MOABC	Djaya
1	0.04	0.03	0.01	0.00
2	1.96	0.92	0.67	0.00
3	1.28	1.07	0.57	0.00
4	1.39	0.70	0.51	0.00
5	1.27	0.98	0.67	0.00
6	1.50	1.12	0.47	0.00
7	3.03	2.60	0.81	0.00

TABLE V
PROPORTION COMPARISONS FOR TOTAL FLOW TIME VERSUS INSTABILITY

Case	PF	NSGAII		MOEA/D		MOABC		DJaya	
	N^A	N	P^i %	N	P^i %	N	P^i %	N	P^i %
1	4	4	25.0	4	50.0	4	75.0	4	100
2	3	5	0.0	4	0.0	4	0.0	3	100
3	3	7	0.0	5	0.0	5	0.0	3	100
4	3	4	0.0	3	0.0	4	0.0	3	100
5	4	4	0.0	5	0.0	4	0.0	4	100
6	5	6	0.0	4	0.0	6	0.0	5	100
7	4	4	0.0	5	0.0	3	0.0	4	100

case 2, which can dominate the results by NSGA-II, MOEA/D, and MOABC. For cases 3–7, DJaya algorithm can find different number of nondominated results. All these results by DJaya can dominate those by the three peers for their corresponding cases.

It can be seen from Table IV that the IGD results for the total flow time and instability by using NSGA-II, MOEA/D, and MOABC are nonzero for all cases. However, the IGD results by using DJaya are the smallest (0.00) for all cases. It means that all nondominated solutions by DJaya are Pareto optimal. Hence, DJaya has the best IGD results among four algorithms for the bi-objective about total flow time and instability.

In Table V, the proportion values of NSGA-II, MOEA/D, and MOABC are just nonzero (25.0, 50.0, and 75.0) for case 1 and zero for cases 2–7. It means that they can just find a portion of Pareto optimal solutions for case 1, since their proportion values are less than 100. The proportion value of DJaya is 100 for seven cases, thus implying that DJaya can find all Pareto optimal solutions in PF. In one word, DJaya has the best performance among four algorithms for the problem aiming to minimize the instability and completion time.

D. Bi-objective Optimization for Instability and Machine Workload

This section discusses the results of bi-objective FJRP aiming to minimize instability and either the maximum machine workload (F_3), or the total machine workload (F_4). The nondominated solutions by the four algorithms for F_3 are recorded in the fourth table in the supplementary material.

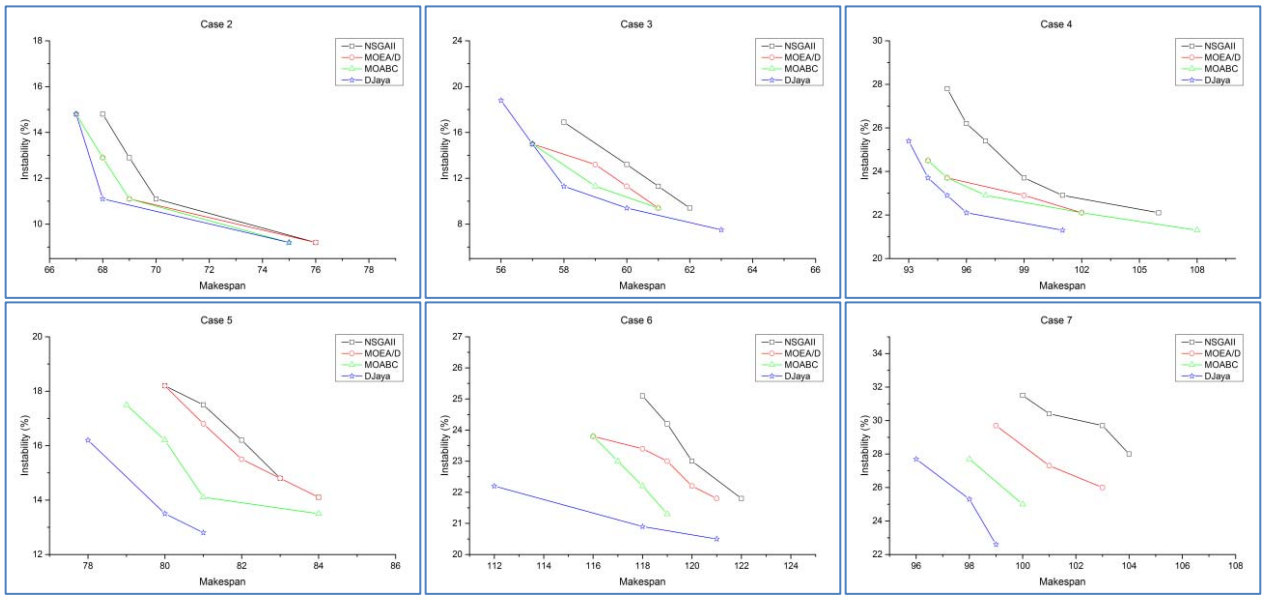


Fig. 4. Pareto results of makespan versus instability.

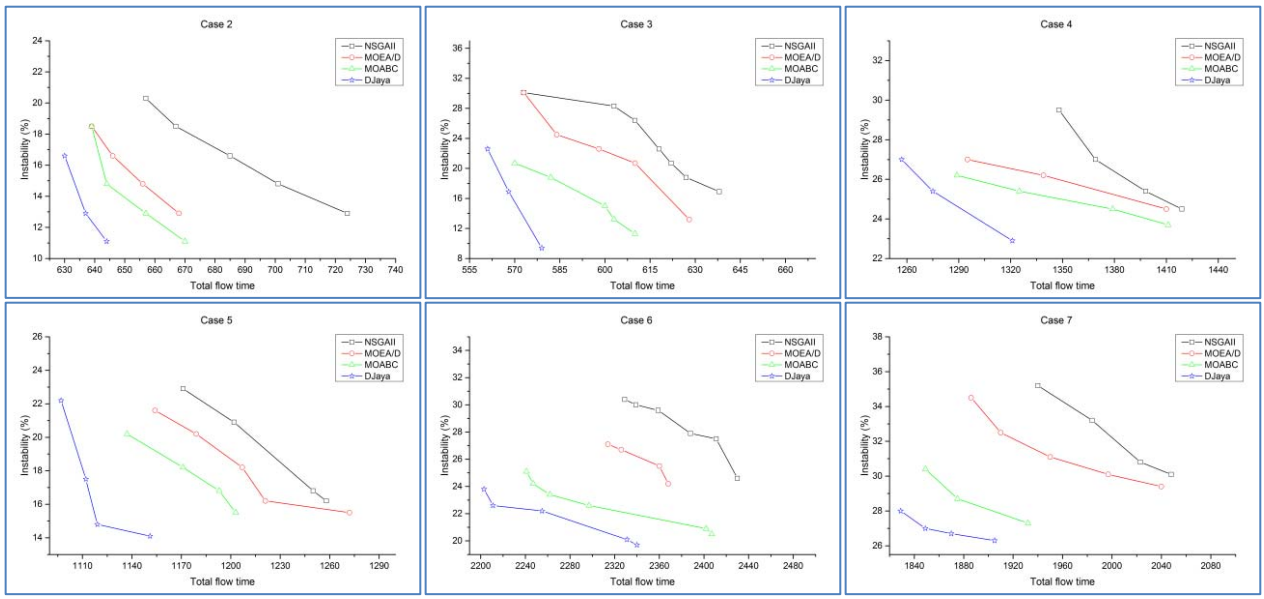


Fig. 5. Pareto results of total flow time versus instabilit.

The results about IGD and proportion for F_3 are shown in Tables VI and VII. The IGD and proportion results for F_4 are presented in Tables VIII and IX while the nondominated solutions by four algorithms are presented in the fifth table in the supplementary material.

First, we examine the results about the maximum machine workload and instability (F_3). For case 1, four algorithms find the same results. Similar to the F_1 situation, differences among the results from four algorithms increase as FJRP scale. To show the differences of the nondominated solutions by four algorithms, the results of F_3 for cases 2 to 7 are presented in Fig. 6. For case 2, four algorithms can find almost the same results, i.e., three nondominated solutions, beside the second nondominated solution by NSGA-II. For case 3, DJaya obtains two nondominated results, which are the optimal solutions. MOEA/D and MOABC can find the same one of the two optimal solutions while NSGA-II cannot find any. For cases 4–7, only DJaya can find the optimal solutions. DJaya has the better performance than its three peers for bi-objective F_3, especially for the large-scale cases.

As shown in Table VI, the IGD results by four algorithms for case 1 are the same (0.00). For case 2, the results obtained by MOEA/D, MOABC, and DJaya are zero while the result by NSGA-II is 0.08. For cases 3–7, the IGD results by DJaya are smallest (0.00) while the results by NSGA-II, MOEA/D, and MOABC are larger than zero. It means that all nondominated solutions by DJaya are Pareto optimal. Hence, DJaya has the best IGD results among four algorithms

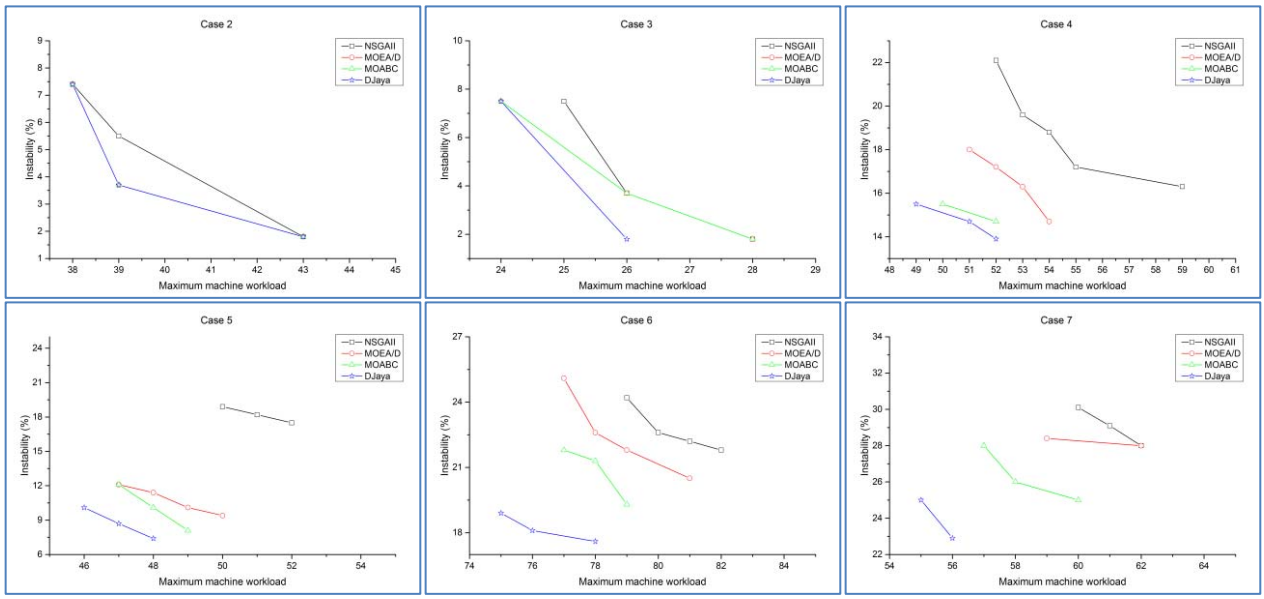


Fig. 6. Pareto results of maximum machine workload versus instability.

TABLE VI
IGD VALUES FOR THE MAXIMUM MACHINE WORKLOAD VERSUS INSTABILITY

Case	NSGAII	MOEA/D	MOABC	Djaya
1	0.00	0.00	0.00	0.00
2	0.08	0.00	0.00	0.00
3	0.39	0.22	0.22	0.00
4	2.23	1.02	0.42	0.00
5	4.09	1.06	0.76	0.00
6	3.42	2.43	1.36	0.00
7	5.37	4.09	2.48	0.00

TABLE VII
PROPORTION COMPARISONS FOR THE MAXIMUM MACHINE WORKLOAD VERSUS INSTABILITY

Case	PF	NSGAII		MOEA/D		MOABC		DJaya	
	N^A	N	P^i %	N	P^i %	N	P^i %	N	P^i %
1	3	3	100	3	100	3	100	3	100
2	3	3	66.7	3	100	3	100	3	100
3	2	3	0.0	3	33.3	3	33.3	2	100
4	3	5	0.0	4	0.0	2	0.0	3	100
5	3	3	0.0	4	0.0	3	0.0	3	100
6	3	4	0.0	4	0.0	3	0.0	3	100
7	2	3	0.0	2	0.0	3	0.0	2	100

for the bi-objective FJRP minimizing the maximum machine workload and instability.

It can be seen from Table VII that NSGA-II has the ideal proportion value (100) just for case 1, 66.7 for case 2, and zero for cases 3–7. It means that NSGA-II can just find Pareto optimal solutions for case 1 and some for case 2. MOEA/D and MOABC can obtain Pareto optimal solutions for cases 1 and 2. For case 3, MOEA/D and MOABC can just find some Pareto optimal solutions. The proportion values of DJaya are 100 for seven cases, indicating that it can find all Pareto optimal solutions in PF.

Next, let us show the results about total machine workload and instability (F_4). Since the total machine workload is the sum of the workload of all machines, the differences among four algorithms become sharp. For case 1 with a small scale, the nondominated solutions by four algorithms are obviously different (see the fourth table in the supplementary material). DJaya finds four optimal solutions for case 1. Among the three peers, only MOABC can obtain one optimal solution. The difference among four algorithms becomes larger and larger with FJRP size. To show the differences among nondominated solutions by four algorithms for cases 2–7, the results by four algorithms are shown in Fig. 7. For case 2, DJaya obtains five nondominated solutions, which dominate the results obtained by its three peers. For cases 3–6, the results by DJaya also dominate those by the three compared algorithms. Especially, both the number and the diversity of nondominated solutions by DJaya are better than those of the compared algorithms for cases 3 and 6. For case 7 with the largest scale, the differences among the nondominated solutions from four algorithms are very clear. The gaps between the results by DJaya and the others become larger and larger with the increasing problem scale.

From Table VIII, the IGD results for F_4 by NSGA-II, MOEA/D, and MOABC are nonzero for all cases. However, the IGD results by DJaya are the smallest (0.00) for all cases. In other words, all the nondominated solutions by DJaya are Pareto optimal. Hence, DJaya has the best IGD results among four algorithms for FJRP minimizing total machine workload and instability.

In Table IX, the proportion values of NSGA-II, MOEA/D, and MOABC are just nonzero for case 1 and zeros for cases 2–7. It means that NSGA-II, MOEA/D, and MOABC

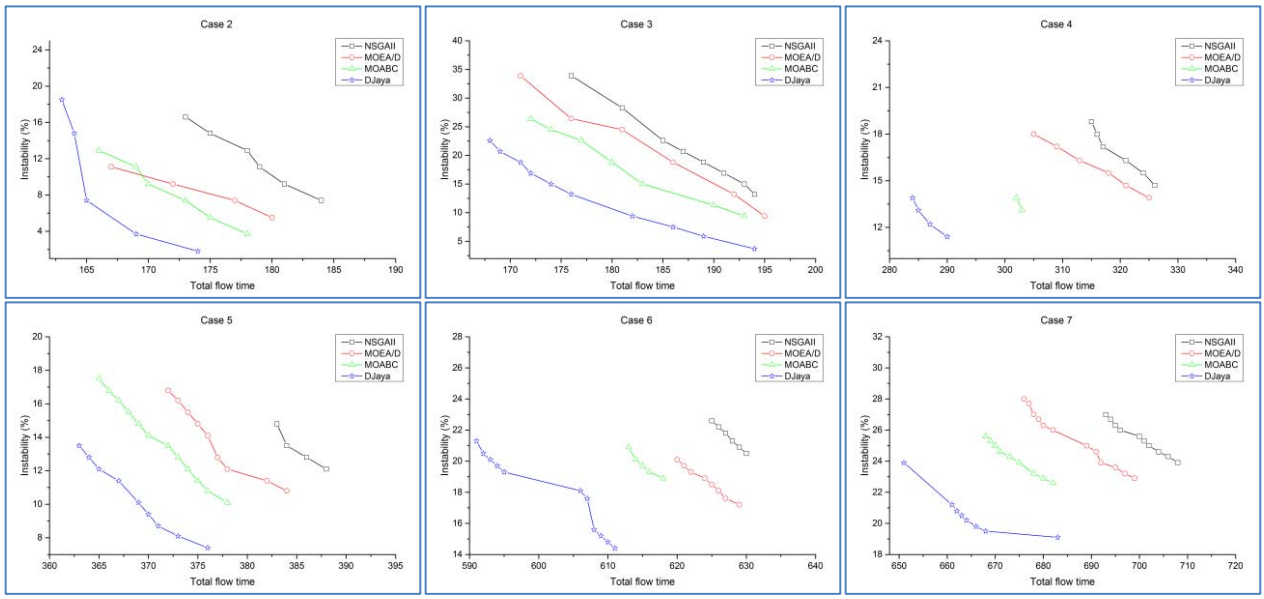


Fig. 7. Pareto results of total machine workload versus instability.

TABLE VIII
IGD VALUES FOR TOTAL MACHINE WORKLOAD VERSUS INSTABILITY

Case	NSGAII	MOEA/D	MOABC	Djaya
1	1.54	1.42	0.59	0.00
2	0.82	0.42	0.30	0.00
3	0.54	0.40	0.29	0.00
4	5.30	3.76	2.58	0.00
5	1.32	0.81	0.50	0.00
6	1.42	1.06	0.80	0.00
7	1.47	1.09	0.72	0.00

TABLE IX
PROPORTION COMPARISONS FOR TOTAL MACHINE WORKLOAD VERSUS INSTABILITY

Case	PF	NSGAII		MOEA/D		MOABC		DJaya	
	N^A	N	P^i %	N	P^i %	N	P^i %	N	P^i %
1	4	4	25.0	4	50.0	4	75.0	4	100
2	3	5	0.0	4	0.0	4	0.0	3	100
3	3	7	0.0	5	0.0	5	0.0	3	100
4	3	4	0.0	3	0.0	4	0.0	3	100
5	4	4	0.0	5	0.0	4	0.0	4	100
6	5	6	0.0	4	0.0	6	0.0	5	100
7	4	4	0.0	5	0.0	3	0.0	4	100

can just find some Pareto optimal solutions for case 1, since the proportion values of these three algorithms are less than 100. The proportion values of DJaya are 100 for all cases, signifying its ability to find all Pareto optimal solutions in PF. In summary, DJaya has better performance than its peers for FJRP minimizing instability and machine workload.

VI. CONCLUSION

This paper presents a flexible job shop rescheduling problem with new job insertion and a discrete Jaya (DJaya) algorithm. It focuses on bi-objective optimization cases whose objective is to minimize instability and another performance index among makespan, total job completion time, maximum machine workload, and total machine workload. A discretization procedure is proposed to make Jaya applicable for handling discrete optimization problems. Four ensembles of objective-oriented local search operators are proposed to improve the performance of DJaya. The superior performance of the proposed approach is verified through its comparison with three state-of-the-art methods.

Our future studies intend to pursue the following directions.

1) Improve the performance of DJaya and develop more meta-heuristics for flexible job shop rescheduling problems.
2) Solve them with multiple and many objectives.
3) Develop more problem feature-based local search operators and ensembles to improve the convergence of meta-heuristics.
4) Compare our algorithms with other methods, e.g., swarm intelligence [37], [38], hybrid estimation-of-distribution algorithm [39], ε-constraint method [40] and a robust optimization framework [41] for solving flexible job shop scheduling and rescheduling problems.
5) Apply DJaya to other real-world problems [30], [42], [43].

REFERENCES

[1] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Math. Oper. Res.*, vol. 1, no. 2, pp. 117–129, 1976.
[2] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Computing*, vol. 45, no. 4, pp. 369–375, 1990.
[3] A. S. Jain and S. Meeran, "Deterministic job-shop scheduling: Past, present and future," *Eur. J. Oper. Res.*, vol. 113, no. 2, pp. 390–434, 1999.

[4] I. Kacem, S. Hammadi, and P. Borne, "Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 32, no. 1, pp. 1–13, Feb. 2002.

[5] X. Guo, S. Liu, G. Tian, and M. Zhou, "Disassembly sequence optimization for large-scale products with multiresource constraints using scatter search and Petri nets," *IEEE Trans. Cybern.*, vol. 46, no. 11, pp. 2435–2446, Nov. 2016.

[6] H. Yuan *et al.*, "TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.

[7] L. Wang, G. Zhou, Y. Xu, and M. Liu, "A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3593–3608, 2013.

[8] S. Wang, L. Wang, Y. Xu, and M. Liu, "An effective estimation of distribution algorithm for the flexible job-shop scheduling problem with fuzzy processing time," *Int. J. Prod. Res.*, vol. 51, no. 12, pp. 3778–3793, 2013.

[9] K. Z. Gao, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time," *Int. J. Prod. Res.*, vol. 53, no. 19, pp. 5896–5911, 2015.

[10] J. Xiong, L. N. Xing, and Y. W. Chen, "Robust scheduling for multi-objective flexible job shop problems with random machine breakdown," *Int. J. Prod. Econ.*, vol. 141, no. 1, pp. 112–126, 2013.

[11] E. Ahmadi, M. Zandieh, M. Farrokh, and S. M. Emami, "A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms," *Comput. Oper. Res.*, vol. 73, pp. 56–66, Sep. 2016.

[12] 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.

[13] X.-L. Zheng and L. Wang, "A knowledge-guided fruit fly optimization algorithm for dual resource constrained flexible job-shop scheduling problem," *Int. J. Prod. Res.*, vol. 54, no. 18, pp. 5554–5566, 2016.

[14] 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.

[15] S. Karimi, A. Ardalan, B. Naderi, and M. Mohammadi, "Scheduling flexible job-shops with transportation times: Mathematical models and a hybrid imperialist competitive algorithm," *Appl. Math. Model.*, vol. 41, pp. 667–682, Jan. 2017.

[16] M. L. Junior and M. G. Filho, "Production planning and control for remanufacturing: Literature review and analysis," *Prod. Plan. Control*, vol. 23, no. 6, pp. 419–435, 2012.

[17] J. A. Krupp, "Structuring bills of material for automotive remanufacturing," *Prod. Inventory Manag. J.*, vol. 34, no. 4, pp. 46–52, 1993.

[18] M. Ferguson, V. D. Guide, E. Koca, and G. C. Souza, "The value of quality grading in remanufacturing," *Prod. Oper. Manag.*, vol. 18, no. 3, pp. 300–314, 2009.

[19] J. Gao, L. Sun, and M. Gen, "A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems," *Comput. Oper. Res.*, vol. 35, no. 9, pp. 2892–2907, 2008.

[20] J. Li, J. Zhang, C. Jiang, and M. C. Zhou, "Composite particle swarm optimizer with historical memory for function optimization," *IEEE Trans. Cybern.*, vol. 45, no. 10, pp. 2350–2363, Oct. 2015.

[21] 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.

[22] K. Z. Gao, Y. C. Zhang, A. Sadollah, A. Lentzakis, and R. Su, "Jaya, harmony search and water cycle algorithms for solving large-scale real-life urban traffic light scheduling problem," *Swarm Evol. Comput.*, vol. 37, pp. 58–72, 2017.

[23] R. V. Rao, "Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems," *Int. J. Ind. Eng. Comput.*, vol. 7, no. 1, pp. 19–34, 2016.

[24] R. V. Rao, K. C. More, J. Taler, and P. Ocłoñ, "Dimensional optimization of a micro-channel heat sink using Jaya algorithm," *Appl. Therm. Eng.*, vol. 103, pp. 572–582, Jun. 2016.

[25] R. V. Rao and G. G. Waghmare, "A new optimization algorithm for solving complex constrained design optimization problems," *Eng. Optim.*, vol. 49, no. 1, pp. 60–83, 2017.

[26] R. V. Rao and D. P. Rai, "Optimisation of welding processes using quasi-oppositional-based Jaya algorithm," *J. Exp. Theor. Artif. Intell.*, vol. 29, no. 5, pp. 1099–1117, 2017.

[27] R. V. Rao, D. P. Rai, and J. Balic, "A multi-objective algorithm for optimization of modern machining processes," *Eng. Appl. Artif. Intell.*, vol. 61, pp. 103–125, May 2017.

[28] R. V. Rao and A. Saroj, "Economic optimization of shell-and-tube heat exchanger using Jaya algorithm with maintenance consideration," *Appl. Therm. Eng.*, vol. 116, pp. 473–487, Apr. 2017.

[29] R. V. Rao, D. P. Rai, and J. Balic, "Surface grinding process optimization using Jaya algorithm," *Comput. Intell. Data Min.*, vol. 2, pp. 487–495, Dec. 2015.

[30] N. Q. Wu, M. C. Zhou, and Z. W. Li, "Short-term scheduling of crude-oil operations: Petri net-based control-theoretic approach," *IEEE Robot. Autom. Mag.*, vol. 22, no. 2, pp. 64–76, Jun. 2015.

[31] V. D. R. Guide, "Production planning and control for remanufacturing: Industry practice and research needs," *J. Oper. Manag.*, vol. 18, no. 4, pp. 467–483, 2000.

[32] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Comput. Oper. Res.*, vol. 35, no. 10, pp. 3202–3212, 2008.

[33] K. Z. Gao *et al.*, "Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling," *Inf. Sci.*, vol. 289, pp. 76–90, Dec. 2014.

[34] 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.

[35] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–713, Dec. 2007.

[36] J. Luo *et al.*, "An artificial bee colony algorithm for multi-objective optimisation," *Appl. Soft Comput.*, vol. 50, pp. 235–251, Jan. 2017.

[37] W. Dong and M. Zhou, "A supervised learning and control method to improve particle swarm optimization algorithms," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 7, pp. 1135–1148, Jul. 2017.

[38] Feng, M. Zhou, A. C. Ammari, and K. Sedraoui, "Optimal load scheduling of plug-in hybrid electric vehicles via weight-aggregation multi-objective evolutionary algorithms," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2557–2568, Sep. 2017.

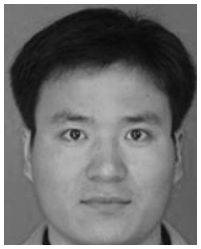
[39] L. Wang, S. Wang, and X. Zheng, "A hybrid estimation of distribution algorithm for unrelated parallel machine scheduling with sequence-dependent setup times," *IEEE/CAA J. Automatica Sinca*, vol. 3, no. 3, pp. 235–246, Jul. 2016.

[40] P. Wu, A. Che, F. Chu, and M. Zhou, "An improved exact ε-constraint and cut-and-solve combined method for biobjective robust lane reservation," *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 3, pp. 1479–1492, Jun. 2015.

[41] B. Wang, X. Xia, H. Meng, and T. Li, "Bad-scenario-set robust optimization framework with two objectives for uncertain scheduling systems," *IEEE/CAA J. Automatica Sinica*, vol. 4, no. 1, pp. 143–153, Jan. 2017.

[42] N. Q. Wu, M. C. Zhou, L. P. Bai, and Z. W. Li, "Short-term scheduling of crude oil operations in refinery with high fusion point oil and two transportation pipelines," *Enterprise Inf. Syst.*, vol. 10, no. 6, pp. 581–610, May 2016.

[43] Y. Hou, N. Q. Wu, M. C. Zhou, and Z. W. Li, "Pareto-optimization for scheduling of crude oil operations in refinery via genetic algorithm," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 47, no. 3, pp. 517–530, Mar. 2017.



Kaizhou Gao (M'16) received the B.Sc. degree from Liaocheng University, Liaocheng, China, in 2005, the master's degree from Yangzhou University, Yangzhou, China, in 2008, and the Ph.D. degree from Nanyang Technological University (NTU), Singapore, 2016.

From 2008 to 2012, he was with the School of Computer, Liaocheng University, Liaocheng, China. In 2012, he was a Research Associate with the School of Electronic and Electrical Engineering, NTU, Singapore, for seven months. From 2013 to 2015, he was a Software Engineer with the Singapore Institute of Manufacturing Technology, A*STAR, Singapore. Since 2015, he has been a Research Fellow with NTU. He has published over 50 refereed papers. His current research interests include intelligent computation, optimization, scheduling, intelligent transportation, container operation, and AGV routing.

Fajun Yang received the B.S. degree in industrial engineering from the Hunan University of Science and Technology, Xiangtan, China, in 2011 and the Ph.D. degree in mechanical engineering from the Guangdong University of Technology, Guangzhou, China, in 2016.

From 2015 to 2016, he was a visiting student with the New Jersey Institute of Technology, Newark, NJ, USA. He is currently a Research Fellow with Nanyang Technological University, Singapore. He has over 10 international journal papers (majority in the IEEE TRANSACTIONS). His current research interests include Petri nets, production planning, discrete event systems, scheduling, and control.

Dr. Yang has served as a reviewer for a number of journals.

MengChu Zhou (S'88–M'90–SM'93–F'03) received the B.S. degree in control engineering from the Nanjing University of Science and Technology, Nanjing, China, in 1983, the M.S. degree in automatic control from the Beijing Institute of Technology, Beijing, China, in 1986, and the Ph.D. degree in computer and systems engineering from Rensselaer Polytechnic Institute, Troy, NY, USA, in 1990.

He joined the New Jersey Institute of Technology, Newark, NJ, USA, in 1990, where he is currently a Distinguished Professor of electrical and computer engineering. He has over 700 publications including 12 books, over 400 journal papers (over 300 in IEEE TRANSACTIONS), and 28 book-chapters. He holds 11 patents and several pending ones. He has led or participated in over 50 research and education projects with total budget over $12M, funded by National Science Foundation, Department of Defense, NIST, New Jersey Science and Technology Commission, and industry. His current research interests include Petri nets, intelligent automation, Internet of Things, big data, and intelligent transportation.

Dr. Zhou was a recipient of the Humboldt Research Award for U.S. Senior Scientists, the Franklin V. Taylor Memorial Award, and the Norbert Wiener Award from IEEE Systems, Man and Cybernetics Society. He is the Founding Editor of IEEE Press Book Series on Systems Science and Engineering and an Editor-in-Chief of the *IEEE/CAA Journal of Automatica Sinica*. He served as an Associate Editor for the IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION, the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING, the IEEE TRANSACTIONS ON SYSTEMS, MAN AND CYBERNETICS: SYSTEMS, and the IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, and an Editor of the IEEE TRANSACTIONS ON AUTOMATION SCIENCE AND ENGINEERING. He served as a Guest-Editor for many journals, including the IEEE INTERNET OF THINGS JOURNAL, the IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS, and the IEEE TRANSACTIONS ON SEMICONDUCTOR MANUFACTURING. He is also an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS and *Frontiers of Information Technology and Electronic Engineering*. He was the General Chair of IEEE Conference on Automation Science and Engineering, Washington, DC, USA, in 2008, the General Co-Chair of 2003 IEEE International Conference on System, Man and Cybernetics (SMC), Washington, DC, USA, in 2003, the Founding General Co-Chair of 2004 IEEE International Conference on Networking, Sensing and Control, Taipei, in 2004, and the General Chair of 2006 IEEE International Conference on Networking, Sensing and Control, Ft. Lauderdale, FL, USA, in 2006. He was the Program Chair of 2010 IEEE International Conference on Mechatronics and Automation, in 2010, XiŠan, China, in 1998 and 2001 IEEE International Conference on SMC and 1997 IEEE International Conference on Emerging Technologies and Factory Automation. He organized and chaired over 100 technical sessions and served on program committees for many conferences. He is a Life Member of the Chinese Association for Science and Technology–USA and served as its President in 1999. He is a fellow of International Federation of Automatic Control and the American Association for the Advancement of Science.

Quanke Pan received the B.Sc. and Ph.D. degrees from the Nanjing University of Aeronautics and Astronautics, Nanjing, China, in 1993 and 2003, respectively.

Since 2003, he has been with the School of Computer Science Department, Liaocheng University, Liaocheng, China, where he became a Full Professor in 2006. He has been with the State Key Laboratory of Digital Manufacturing and Equipment Technology, Huazhong University of Science and Technology, Wuhan, China, since 2014. He has authored one academic book and over 150 refereed papers. His current research interests include intelligent optimization and scheduling.

Dr. Pan acts as the Editorial Board Member for several journals, including *Operations Research Perspective*, *Swarm and Evolutionary Computation*, the *American Journal of Management Studies*, *ISRN Artificial Intelligence*, and *Progress in Intelligent Computing and Applications*.

Ponnuthurai Nagaratnam Suganthan (S'90–M'92–SM'00–F'15) received the B.A. degree, the Postgraduate Certificate, and the M.A. degree in electrical and information engineering from the University of Cambridge, Cambridge, U.K., in 1990, 1992, and 1994, respectively, and the Ph.D. degree from Nanyang Technological University, Singapore, in 1995.

He served as a Predoctoral Research Assistant with the Department of Electrical Engineering, University of Sydney, Sydney, NSW, Australia, from 1995 to 1996 and a Lecturer with the Department of Computer Science and Electrical Engineering, University of Queensland, Brisbane, QLD, Australia, from 1996 to 1999. He moved to Nanyang Technological University in 1999. His SCI indexed publications attracted over 1000 SCI citations in each calendar years 2013, 2014, 2015, 2016, and 2017. His current research interests include swarm and evolutionary algorithms, pattern recognition, big data, deep learning, combinatorial optimization and applications of swarm, and evolutionary and machine learning algorithms.

Dr. Suganthan was a recipient of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award in 2012 for his co-authored published in 2009 and the Highly Cited Researcher Award in computer science by Thomson Reuters in 2015, 2016, and 2017. His former Ph.D. student, Dr. J. J. Liang, won the IEEE CIS Outstanding Ph.D. Dissertation Award, in 2014. He is an Editorial Board Member of the *Evolutionary Computation Journal*, MIT Press. He has been an Associate Editor of the IEEE TRANSACTIONS ON CYBERNETICS since 2012, the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION since 2005, *Information Sciences* (Elsevier) since 2009, *Pattern Recognition* (Elsevier) since 2001, and the *International Journal of Swarm Intelligence Research* since 2009. He has been a Founding Co-Editor-in-Chief of *Swarm and Evolutionary Computation* since 2010, an SCI Indexed Elsevier Journal. He served as the General Chair of the IEEE SSCI 2013. He has been an Elected AdCom Member of the IEEE Computational Intelligence Society in 2014–2016.