IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 12, 2020, accepted April 25, 2020, date of publication May 6, 2020, date of current version May 20, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2992478

Improved Jaya Algorithm for Flexible Job Shop Rescheduling Problem

KAIZHOU GAO[1,3], FAJUN YANG[2], JUNQING LI[3], HONGYAN SANG[3], AND JIANPING LUO[4]

[1]Macau Institute of Systems Engineering, Macau University of Science and Technology, Taipa 999078, China
[2]School of Mathematics and Computer Science, University of Hagen, 58097 Hagen, Germany
[3]School of Computer, Liaocheng University, Liaocheng 252000, China
[4]College of Information Engineering, Shenzhen University, Shenzhen 518060, China

Corresponding author: Fajun Yang (fjyang1116@foxmail.com)

This work was supported in part by the Shandong Province Colleges and Universities Youth Innovation Talent Introduction and Education Program, in part by the National Natural Science Foundation of China under Grant 61603169, Grant 61773192, Grant 61503170, and Grant 61803192, in part by the Shandong Provincial Key Laboratory for Novel Distributed Computer Software Technology, and in part by the Alexander von Humboldt Foundation.

ABSTRACT Machine recovery is met from time to time in real-life production. Rescheduling is often a necessary procedure to cope with it. Its instability gauges the number of changes to the existing scheduling solutions. It is a key criterion to measure a rescheduling solution's quality. This work aims at solving a flexible job shop problem with machine recovery, which arises from the scheduling and rescheduling of pump remanufacturing systems. In their scheduling phase, the objective is to minimize makespan. In their rescheduling phase, two objectives are to minimize both instability and makespan. By introducing two novel local search operators into the original Jaya algorithm, this work proposes an improved Jaya algorithm to solve it. It performs experiments on ten different-scale cases of real-life remanufacturing environment. The results show that the improved Jaya is effective and efficient for solving a flexible job shop scheduling and rescheduling problems. It can effectively balance instability and makespan in a rescheduling phase.

INDEX TERMS Jaya algorithm, flexible job shop scheduling, machine recovery, remanufacturing, scheduling and rescheduling.

I. INTRODUCTION

Production scheduling and task scheduling are extremely crucial in industrial manufacturing [43], [46] and data center [47], [48]. In many industrial fields, e.g., wafer fabrication, mechanical manufacturing, and automobile assembly environment, flexible job shop problem (FJSP) is very common [3], and it consists of two parts: machine assignment and operation sequencing [1], [2]. There are some constraints existing in FJSP, which must be well solved in real-life environment [4]. Many researchers have studied various issues in FJSP, e.g., machine breakdowns [5], [6], fuzzy processing time [7]–[9], resource and energy constraints [10], [11], transportation time [12] and new job insertion [13]–[15].

For FJSP with machine breakdown, the studies [16] and [17] propose Genetic algorithm (GA) related algorithms. In [16], a hybrid GA is developed to solve FJSP with random breakdown for robust and stable objectives. In [17], a GA with special chromosome encoding to adapt the machine disruption is proposed. It is compared with a right-shift reschedule and a pre-scheduler to verify its competitiveness. For the same problem and constraint, a rescheduling process is performed in [18] by using route changing and right-shift strategies. It handles right-shift scheduling, route changes and idle time insertion. In [19], game theory is employed to solve FJSP with machine breakdown of its objectives to maximize robustness and stability. The objectives are seen as two sides of the game and ideal Nash Equilibrium (NE) and near NE solution strategies are developed to achieve the optimal solution. Based on this idea, one NE searching algorithm is proposed and benchmarks with machine breakdown are tested to verify the performance. Ahmadi *et al.* [5] employ non-dominated ranking genetic algorithm and non-dominated sorting genetic algorithm II to

The associate editor coordinating the review of this manuscript and approving it for publication was Mengchu Zhou.

solve FJSP with machine breakdown with the objectives to optimize makespan and stability.

In [45], the authors propose a novel machine group-oriented match-up rescheduling approach for dynamic semiconductor manufacturing systems where unexpected events, such as machine breakdown and due date changes could occur. In [42], considering disruptive changes, such as the arrival of new jobs or machine breakdowns, a rescheduling architecture which bases on a predictive-reactive strategy is designed and implemented.

Different strategies, e.g., possible machine break down scheduling by using predictive strategy [20], time-varying scheduling with machine failure rate by using stochastic strategy [21], and machine preventive scheduling for maintenance activities [22] are proposed to deal with dynamic FJSP. They focus on machine failure while machine recovery is rarely considered. When a schedule is executed in a shop, a recovered machine can be selected for the operations that have not been started.

It has been proved that FJSP is an NP-hard problem [23], [24] and recent years have witnessed a new trend, i.e., using and improving meta-heuristics to solve FJSP. Particle swarm optimization [25], genetic algorithm [26], harmony search algorithm [27], artificial bee colony algorithm [28], memetic algorithm [29] and Jaya [13], [30] are among these algorithms. Jaya represents a simple and novel one, which is first developed in [32]. In comparison with existing algorithms, we only need to consider the number of iterations and population size in Jaya and there is not any other parameter. Thus, Jaya is more simple and easier. It has been successfully applied to solving many real-life problems since 2016 [32]–[36].

FJSP exists in many practical engineering fields and it is a topic with significance for theatrical research and value for engineering practices. In this paper, FJSP from pump remanufacturing with machine recovery is studied. In remanufacturing environment, rescheduling is necessary after machine recovery to reduce makespan. Instability is a key metric to measure rescheduling solutions' quality [8], [12]. To solve the concerned problem, Jaya is improved by using our proposed local search strategies. Ten pump remanufacturing cases are scheduled and rescheduled with machine recovery effectively and efficiently.

The rest of the paper is organized as follows. The mathematical model of a flexible job shop scheduling and rescheduling problem with machine recovery is described in the next Section. Then, the improved Jaya (iJaya) algorithm is developed and described in Section III. Section IV shows the comparisons and analysis of experimental results. Finally, the conclusions and some future research directions are given in Section V.

II. PROBLEM MODELING

A. THE MODEL OF FJSP

In the FJSP, a job consists of several operations. One out of multiple candidate machines is selected for one operation. And each operation can be processed on just one machine each time. The following assumptions and notations are used to formulate FJSP.

1. Jobs are denoted as $J = \{J_i\}, 1 \leq i \leq n, n$ is the number of jobs and q_i is the number of operations in job i.
2. Candidate Machines are denoted as $M = \{M_k\}, 1 \leq k \leq m, m$ is the number of machines.
3. Job J_i has a processing sequence with predetermined priority. $O_{i,h}$ indicates operation h of J_i.
4. One operation $O_{i,h}$ is processed on a machine $M(O_{i,h})$ without interruption, i.e., the set of selectable machines for operation $O_{i,h}$. $P_{i,h,k}$ indicates the processing time of $O_{i,h}$ on $M_k \in M(O_{i,h})$.
   Decision variables
   $$x_{i,h,k} = \begin{cases} 0, & \text{machine } M_k \text{is not selected for } O_{i,h} \\ 1, & \text{machine } M_k \text{is selected for } O_{i,h} \end{cases} \quad (1)$$
5. $c_{i,h}$ indicates the $O_{i,h}$'s completion time, and the makespan (the completion time of the last job) is one objective to be optimized in this study,
   $$C_{Max} = \max_{1 \leq i \leq n} c_i, \quad (2)$$

where c_i is the completion time of job J_i.

B. RESCHEDULING AND INSTABILITY FOR MACHINE RECOVERY

Machine recovery exists in a pump remanufacturing environment [37]. For a shop floor that is executing a schedule, when a recovered machine is added in a candidate machine set at time t, the makespan could be reduced by rescheduling the not-yet started operations. The start time of one job in a rescheduling phase is t or the completion time of the operation being currently processed. The available time of one machine in rescheduling stage is t or the completion time of the on-going operation on this machine. The scheduling problem in rescheduling stage becomes an FJSP with different job start time (S_J) and machine available time (S_M). In a rescheduling phase, the start time of machine M_k and job J_i are denoted as follows:

$$S_{M_k} = \begin{cases} t, & \text{No operation is being processed on } M_k \\ c_{i,h}, & \text{Operation } O_{i,h} \text{is being processed on } M_k \text{ at } t \end{cases} \quad (3)$$

$$S_{J_i} = \begin{cases} t, & \text{No operation is being processed at } t \\ c_{i,h}, & \text{Operation } O_{i,h} \text{ is being processed at } t \end{cases} \quad (4)$$

In the above formulas, t is the time a recovered machine becoming available. We show an example in Fig. 1 to explain the problem more clearly. Fig. 1(a) shows one schedule which is being implemented. Makespan is 14. At time 3, one recovered machine M_4 is added to the candidate machine set. Fig. 1(b) presents a rescheduling solution after moving some operations to M_4. As a result, makespan is reduced to 12 even if the processing time of operations $O_{2,2}$ and $O_{1,3}$

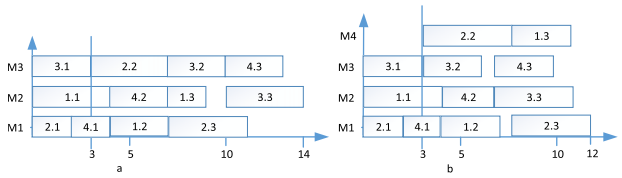

FIGURE 1. An example of rescheduling with a recovered machine.

become larger on M_4 than those on M_3 and M_2. Of course, if the processing time of operations on machine M_4 is very large, it is possible that the makespan cannot be reduced via rescheduling phase.

Instability describes the percentage of changed operations of the existing jobs on their respective processing machines during rescheduling. In practical production, low instability can reduce rescheduling cost, e.g., materials moving and machine setting. Low instability can guarantee seamless connection between current scheduling solution to rescheduling solution. The work [13] defines instability as the percentage of changed operations, which are moved to different machines in rescheduling. In this study, it is revised into:

$$y_{i,h} = \begin{cases} 0, & O_{i,h} \text{ remains on } M_k \\ 1, & O_{i,h} \text{ moves to } M_{k'} \end{cases} \tag{5}$$

$$\Omega_{i,h} = m * \frac{d_{k,k'}}{d_{k,max}} \tag{6}$$

$$F = \frac{\sum_{i=1}^{n} \sum_{h=1}^{q_i} y_{i,h} * \Omega_{i,h}}{\sum_{i=1}^{n} q_i} \tag{7}$$

where M_k and $M_{k'}$ are operation $O_{i,h}$'s processing machines in the executing schedule and rescheduling solution. $\Omega_{i,h}$ is $O_{i,h}$'s transfer coefficient from M_k to $M_{k'}$. The distance from M_k and $M_{k'}$ is $d_{k,k'}$. The maximum distance from M_k to other machines is $d_{k,max}$, including the recovered one. q_i denotes the number of J_i's operations in a rescheduling phase. The goal is to optimize rescheduling solutions' instability. It is clear that fewer operation exchanging and operations moving to a closer machine mean a higher stability or lower instability of a solution in rescheduling. For example, in Fig. 1(b), the result of makespan is 12, and it is less than 14, which is the makespan result in Fig. 1(a). Operations $O_{2,2}$ and $O_{1,3}$ from M_3 and M_2 are moved to M_4. In Fig. 1(b), the rescheduling solution's instability relates to the physical distances from M_4 to M_3 and M_2, respectively.

C. BI-OBJECTIVE FUNCTION

An optimization problem with multiple objectives can be defined as follows:

$$\begin{aligned} &\text{Minimize } F(X) = (f_1(x), f_2(x), \cdots, f_m(x))^T \\ &\text{Subject to } X \in \Omega \end{aligned} \tag{8}$$

where function $F : \Omega \to \wedge$ includes multiple objectives and a decision vector $x = (x_1, x_1, \cdots, x_n)$ belongs to a decision space Ω. In this study, $f_1 = Cmax$ is the makespan presented in Part A of Section II while $f_2 = F$ is the instability of a rescheduling solution.

A bi-objective function is:

$$\text{Minimize } F(X) = (f_1(x), f_2(x))^T \tag{9}$$

Pareto domination is a widely used strategy that compares and ranks all the solutions for optimization problems with two objectives. For example, there are two solutions $x = (x_1, x_1, \cdots, x_n)$ and $x' = (x'_1, x'_2, \cdots, x'_n)$. $x \prec x'$ (x dominates x′) if and only if $\forall p \in \{1, 2\}, f_p(x) \leq f_p(x')$ and $\exists q \in \{1, 2\}, f_q(x) < f_q(x')$. In the Pareto set, solution x is an optimal one if no other solution x′ dominates x. The collection of all Pareto optimal solutions is a Pareto optimal set and the Pareto front is the corresponding image in the objective space. There is an archive set (AS) employed for storing non-dominated solutions. In the optimizing process, the dominated solutions are replaced by the new one in AS.

III. IMPROVED JAYA ALGORITHM

A. JAYA

In Sanskrit, the meaning of word "Jaya" is victory. By reaching the best solution, the applied strategy in Jaya always tries to become victorious [32]. Its first step is to initialize a population with its size being N_P. Then, the solutions with the worst and the best results are selected to update the population in the following generation. The worst solution and the best one are updated in every generation. The updating method for new generated solutions is defined in the following formula, for $i \in \{1, 2, \ldots, N_P\}$:

$$X_i(t+1) = X_i(t) + r_1 (X_B(t) - X_i(t)) + r_2 (X_W(t) - X_i(t)) \tag{10}$$

where $X_i(t)$ is the current position and $X_i(t+1)$ is the updated position. $X_B(t)$ and $X_W(t)$ are the best and worst solutions, respectively. t is an iteration index and r_1 and r_2 are uniformly distributed random numbers in range [0, 1]. During its updating, if the objective function value of $X_i(t)$ is worse than that of $X_i(t+1)$, then $X_i(t+1)$ is accepted and replaces $X_i(t)$. At the end of each generation, all the remained solutions and updated solutions are treated as the initial solutions in the following iteration. The procedures of Jaya are shown as follows:

Jaya algorithm
S1. Initialization
S2. Extract the best and worst solutions
S3. Modify solutions by using the best and worst ones
S4. If the new solution is better than the original one, accept it and replace the original one; else, keep the later.
S5. If the stop criterion is satisfied, output the optimal solution and objective function values; else, go to S3.

B. IMPROVEMENT STRATEGIES

In the initialization phase, the encoding and decoding strategies in [13] are employed to represent the solution. In a

solution, one element includes three values, the number of job, the number of operation and the number of the machine processing this operation. This encoding strategy makes it easy to change processing machine in designing local search algorithm. The best solution obtained in a scheduling phase is employed as an initial solution. For this initial solution, makespan and instability in a rescheduling phase are the best result of the scheduling phase and zero, respectively. The non-dominated solutions in rescheduling should have smaller makepsan values but larger instability values. Otherwise, the solutions are dominated by the above initial solution.

Since a flexible job shop scheduling and rescheduling problem is NP-hard, we propose two local search operators to improve the exploitation performance of Jaya, resulting an improved Jaya (iJaya). One is for makespan while another for instability. In a rescheduling phase, they are integrated as an ensemble to improve the convergence speed of Jaya. Two operators are presented as follows.

The local search heuristic for Makespan:
For a solution X with makespan C_{Max}
Select machine M and find the last operation $O_{i,h}$ on
If $O_{i,h}$ can be processed on the recovery machine $M_r, M_r \neq M$

a. Move $O_{i,h}$ to M_r and obtain X'
b. **If** the $C'_{Max} < C_{Max}$,
c. Accept X' and replace X
d. Break the local search operator
e. **Endif**

Else
Do
On machine M, select former operation O', and for job i, the former operation O'' is selected.
If operation $O_{i,h}$'s start time is the same as O''s and/or O'''s completion time
Implement Steps *a to e* for O' or / and O'',
Break this heuristic
Endif
While (existing O′ or / and O″)
Endif

This heuristic starts from the last competed operation and is executed just on the operations in a critical path. The computation time for local search is very short and this operator is stopped once the objective value becomes better. Based on the procedures of the local search for Makespan, the complexity is $O(Log2\ (c))$, where c is the count of operations on Machine M. The next one for instability focuses on the operations having more than one candidate machines. It moves the operations to machines with smaller physical distance to reduce instability. Since all operations on Machine M_k are executed the local search for instability, the complexity is $O(c)$, where c is the count of operations on M_k.

Local search for instability objective:
Select one machine M_k randomly,
For each operation $O_{i,h}$ on M_k
If two or more candidate machines can process $O_{i,h}$
Find a different machine candidate $M_{k'}$
If $d_{k',f} < d_{k,f}$ // M_f is $O_{i,h}$'s former processing machine
Move $O_{i,h}$ to $M_{k'}$
Break the local search operator
Endif
Endif
Endfor

TABLE 1. Makespan results by six algorithms.

Instance	KB ACO	TS PCB	TABC	SEA	IJaya		
					C_M	SD	Agv_t (s)
Kacme1	11	11	11	11	11	0.00	0.41
Kacme 2	14	14	14	14	14	0.00	1.08
Kacme 3	11	11	11	11	11	0.00	1.30
Kacme 4	7	7	7	7	7	0.00	1.42
Kacme 5	11	11	11	11	11	0.00	2.97

C. FRAMEWOR OF iJaya

The framework of iJaya is shown in Fig. 2. Two local search operators are employed to improve the exploitation performance in iJaya. In a scheduling phase, iJaya obtains the best solutions for makespan. When a machine is recovered, the rescheduling phase is activated. The best solution in a scheduling phase is used to calculate the start time of jobs and available times of machines for rescheduling. Then, iJaya optimizes the bi-objective function in (9). Finally, iJaya outputs solutions which are not dominated by others in AS for instability and makespan.

IV. EXPERIMENTS AND DISCUSSIONS

A. EXPERIMENTAL SETUP

To solve a flexible job shop scheduling and rescheduling problem by iJaya, we run computer experiments, analyze and discuss the results. Ten different-scale cases coming from real orders of a pump remanufacturing company are solved. Ten cases' scales are from 5 jobs, 23 operations and 4 machines to 20 jobs, 400 operations and 20 machines. For one instance, there is only one recovered machine. The desensitization data of the 10 cases are shown in the Supplemental File for readers. The recovery time of the first two cases is 10, and that of the third and fourth cases is 15. For other cases, the recovery time is 30.

iJaya is coded in C++ and run on an Intel 3.20 GHz desktop computer with 8 GB memory. First, it is compared to some existing algorithms for five benchmark cases of FJSP. Then, it is employed for solving 10 remanufacturing

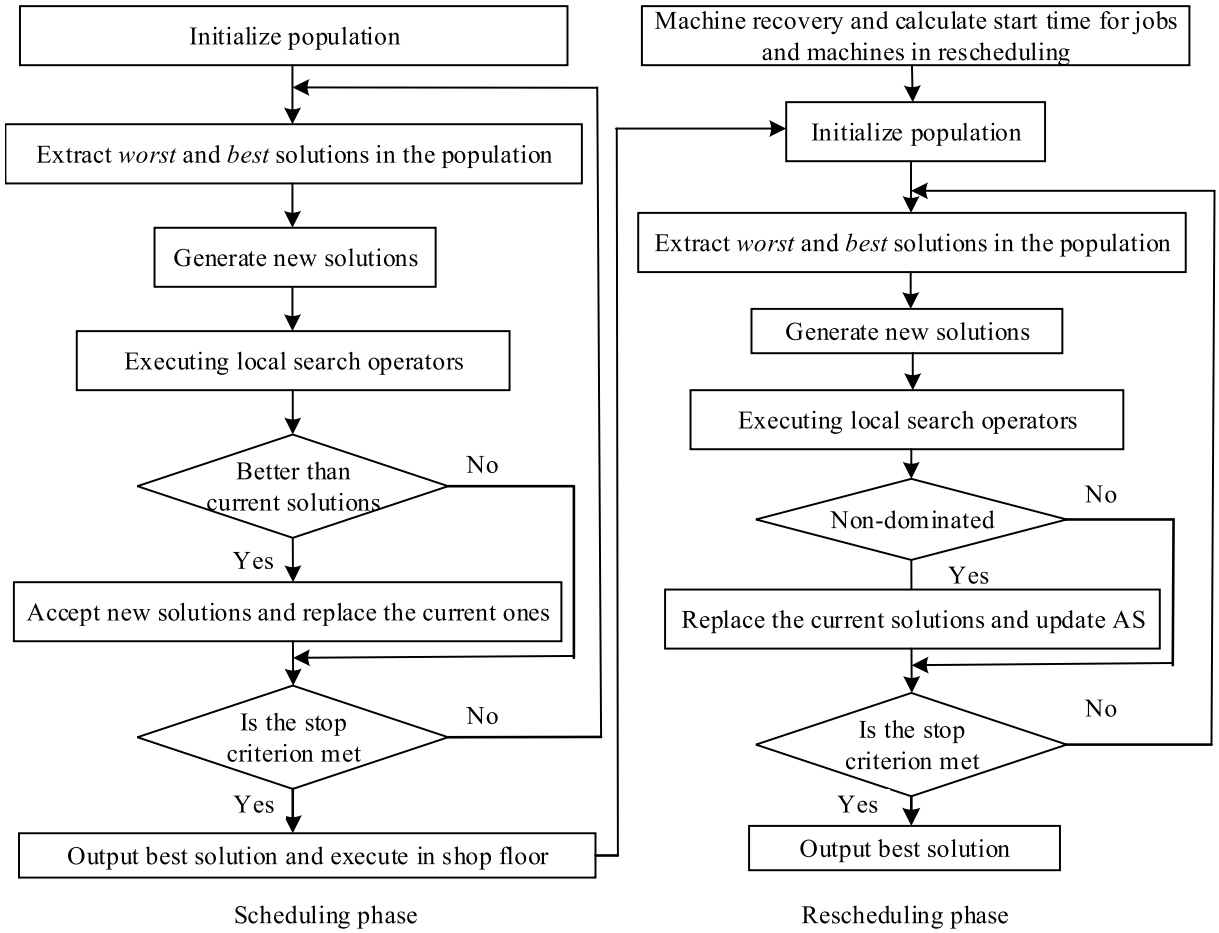

FIGURE 2. iJaya for scheduling and rescheduling flexible job shop with machine recovery.

TABLE 2. Makespan results by Jaya.

Case	Scale	Makespan			
		Min	Ave	Max	CV (%)
Rem1	5×4	25	25.4	27	2.2
Rem2	8×8	38	41.8	46	4.8
Rem3	10×6	57	61.3	71	4.5
Rem4	10×10	47	50.5	60	6.3
Rem5	15×8	84	92.4	102	5.6
Rem6	15×10	74	81.7	91	5.5
Rem7	15×15	56	62.9	74	6.6
Rem8	20×10	107	120.8	137	6.4
Rem9	20×15	91	110.0	123	6.4
Rem10	20×20	79	86.2	99	6.1

cases. Finally, iJaya is employed for rescheduling for machine recovery. For the benchmark FJSP cases, the population size is set to 50 while the maximum generation count is set to 1000. For flexible job shop rescheduling, the size of population and the maximum generation number are 50 and 10000, respectively. All experiments are executed 30 runs.

B. COMAPARING EXISTING ALGORITHMS

To test the performance of iJaya, five benchmark cases for FJSP are solved. iJaya is compared to five competitive metaheuristics, including a simple and effective evolutionary algorithm (SEA), Tabu search with neighborhood local search (TSPCB), two-stage artificial bee colony algorithm (TABC), and knowledge-based ant colony optimization (KBACO) [38]. Table 1 shows the results of makespan of all algorithms. All algorithms can find the best results for five Kacem instances. The second last column of Table 1 shows the standard deviation (SD) of makespan results over 30 repeats of each case by iJaya. Since these cases are FJSP benchmarks, we just show the results of the compared algorithms. For iJaya, both average makespan and running time (Agv_t) over 30 runs are recorded. Based on the comparison results with the existing algorithms, iJaya is an effective and efficient algorithm for scheduling flexible job shop problems. Hence, iJaya is employed for solving scheduling and re-scheduling

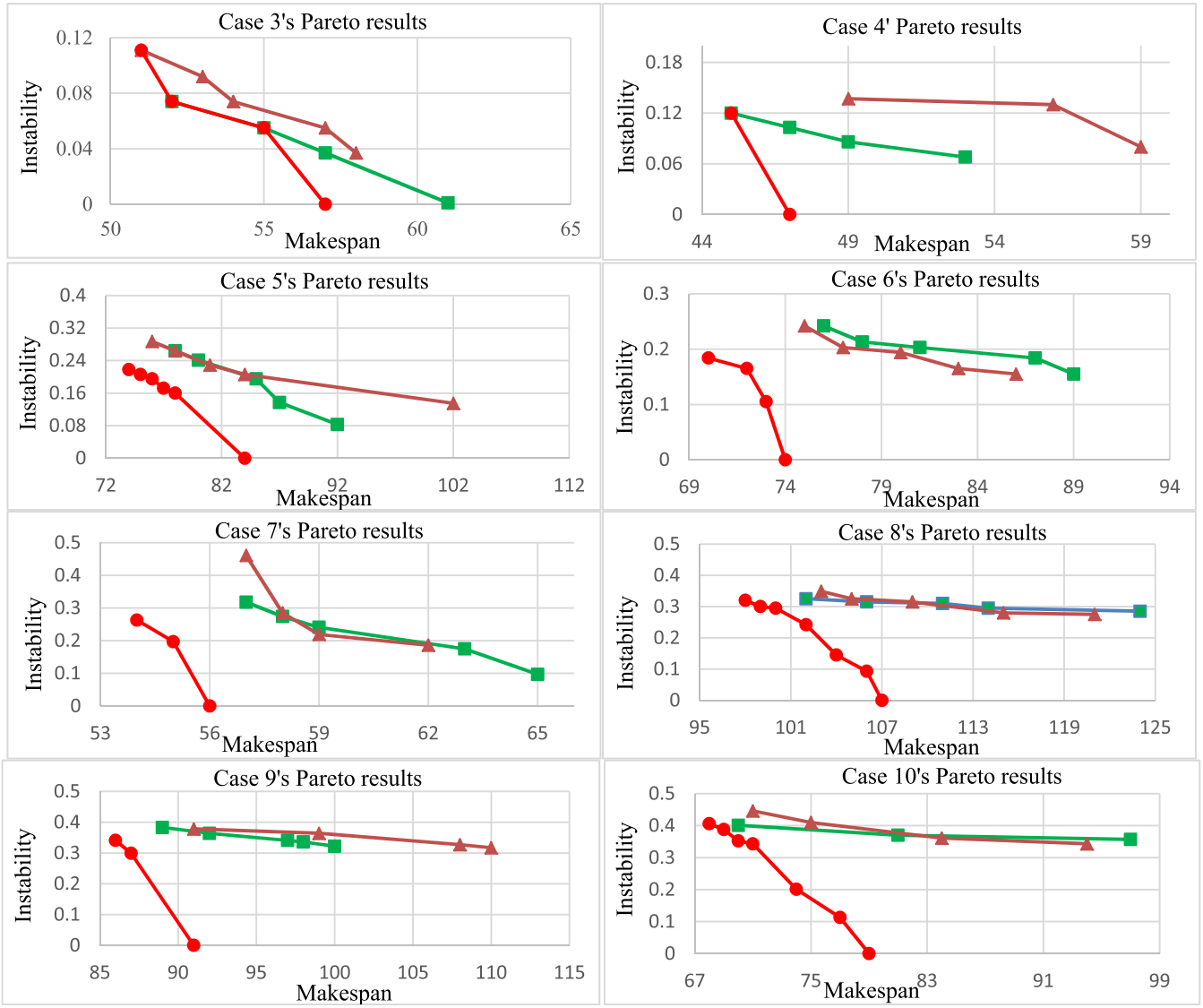

FIGURE 3. Pareto results obtained by three compared algorithms (the results in blue color are obtained by NSGAII, the results in green color are gotten by ISFLA, and the results in red color are received by iJaya, respectively).

flexible job shop problems with machine recovery in the next experiments.

C. iJaya FOR SCHEDULING AND RESCHEDULING

To solve 10 cases from remanufacturing, Table 2 shows the makespan results by iJaya. For each case, the maximum results (Max), the average results (Ave), and the minimum results (Min) in 30 runs are recorded. In this study, as the ratio of the SD over the average value, the coefficient of variation (CV) is employed to replace of the SD measurement. The aim is to show the robustness and stability of iJaya for scheduling and rescheduling flexible job shop problems. From Table 2, the CV values of makespan are less than 7%. It means that iJaya has high robustness and stability for 10 remanufacturing cases. For each case, the solution with the minimum makespan value is selected to execute on a shop floor. When a recovery machine is added to the candidate machine set, the rescheduling phase is activated, to optimize the bi-objective function based on the presently executed schedules.

In a rescheduling phase, iJaya is compared with two multi-objective optimization algorithms, i.e., a classical multi-objective algorithm, NSGAII [39], and a new published algorithm, improved shuffled frog leaping algorithm (ISFLA) [40]. For the real-life FJRP cases, nobody knows the actual Pareto front (PF) since the high complexity. In our work, its approximation is received by comparing non-dominated solutions obtained by three algorithms. The inverted generational distance (IGD) [12], [41] is a widely used performance indicator, which is employed to verify the competitiveness of non-dominated solutions gotten by three compared algorithms. The lower IGD values mean closer distances to the PF. In PF, the solutions' proportion

TABLE 3. IGD values for Makespan vs. instability.

Case	Scale	NSGAII	ISFLA	iJaya
Rem1	5×4	0.00	0.00	0.00
Rem2	8×8	0.00	0.00	0.00
Rem3	10×6	0.07	0.03	0.00
Rem4	10×10	1.33	0.33	0.00
Rem5	15×8	0.06	0.17	0.00
Rem6	15×10	0.60	0.85	0.00
Rem7	15×15	0.88	0.88	0.00
Rem8	20×10	0.29	0.21	0.00
Rem9	20×15	0.45	0.35	0.00
Rem10	20×20	0.21	0.12	0.00

TABLE 4. Proportion comparisons for Makespan vs instability.

Case	Scale	PF	NSGAII		ISFLA		iJaya	
		θ	θ	Pi (%)	θ	Pi (%)	θ	Pi (%)
Rem1	5×4	1	1	100	1	100	1	100
Rem2	8×8	3	3	100	3	100	3	100
Rem3	10×6	4	5	20	4	50	4	100
Rem4	10×10	2	3	0	4	25	2	100
Rem5	15×8	6	5	0	5	0	6	100
Rem6	15×10	4	5	0	5	0	4	100
Rem7	15×15	3	4	0	5	0	3	100
Rem8	20×10	6	5	0	5	0	6	100
Rem9	20×15	3	4	0	5	0	3	100
Rem10	20×20	5	4	0	3	0	5	100

metric [12], [41] obtained by all algorithms is calculated. The larger proportion means better performance. The aim is to obtain highly competitive non-dominated solutions.

For instability and makespan objectives, the Pareto results gotten by three compared algorithms are counted. The IGD and proportion values of results are recored in TABLEs 3 and 4. According to TABLE 3, the IGD results by three algorithms for first two small-scale cases are zero. It means that three algorithms can find the same non-dominated solutions, which are included in PF. For cases from 3 to 10, the IGD values via iJaya are smallest (0.00), which are included in PF. The results via NSGAII, and ISFLA are non-zero. Hence, iJaya has the best competitiveness among three algorithms for IGD and has best performance for rescheduling the FJSP with instability and makespan objectives.

In TABLE 4, θ is the number of non-dominated solutions in PF or obtained by all compared algorithms. It is obvious that the proportion values of NSGAII are non-zero (100, 100, and 20) for the three two cases and zero for cases 4 to 10. NSGAII can only find Pareto optimal solutions for the first two cases and just partial Pareto optimal solutions for the third case. ISFLA can also find Pareto optimal solutions for the first two cases. ISFLA obtains a part of Pareto optimal solutions for the third and fourth cases and the proportion values of these two cases are 50% and 25%. For all cases, the proportion values of iJaya are 100%. It means that iJaya obtains all Pareto solutions. To show the difference of three algorithms more clearly, the Pareto non-dominated solutions by all compared algorithms for all cases are shown in Fig. 3. It can be seen from Fig.3 that the non-dominated results by iJaya are better than those by NSGAII and ISFLA, especially for the instability objective.

V. CONCLUSION AND FUTURE WORKS

This work studies on scheduling and rescheduling flexible job shop problems with machine recovery. An improved Jaya algorithm is developed. The ensemble of two local search operators can improve the convergence speed of iJaya. One bi-objective function with instability and makespan is minimized. The superior performance of the proposed iJaya is verified through comparing with two existing algorithms.

In the future, we plan to

1. Improve Jaya's performance and develop more swarm and evolutionary algorithms for scheduling and rescheduling flexible job shop problems;
2. Consider more scheduling and rescheduling problems with constraints, e.g., environments with disruptions [42], Multiresource Constraints [43].
3. Research on more scheduling and rescheduling problems in some special fields, e.g., semiconductor manufacturing [44], [45] and steel production [46]
4. Develop new local search heuristics based on problem features and ensembles of different local searchers to improve the performance of algorithms.
5. Extend Jaya algorithm to more scheduling and optimization problems, e.g., task scheduling in data center [47].

REFERENCES

[1] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Computing*, vol. 45, no. 4, pp. 369–375, Dec. 1990.
[2] 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.
[3] X. Guo, S. Liu, M. Zhou, and G. Tian, "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.
[4] H. Yuan, J. Bi, W. Tan, M. Zhou, B. H. Li, and J. Li, "TTSA: An effective scheduling approach for delay bounded tasks in hybrid clouds," *IEEE Trans. Cybern.*, vol. 47, no. 11, pp. 3658–3668, Nov. 2017.
[5] 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.
[6] R. Buddala and S. S. Mahapatra, "Two-stage teaching-learning-based optimization method for flexible job-shop scheduling under machine breakdown," *Int. J. Adv. Manuf. Technol.*, vol. 100, nos. 5–8, pp. 1419–1432, Feb. 2019.

[7] J. Lin, "Backtracking search based hyper-heuristic for the flexible job-shop scheduling problem with fuzzy processing time," *Eng. Appl. Artif. Intell.*, vol. 77, pp. 186–196, Jan. 2019.
[8] L. Sun, L. Lin, M. Gen, and H. Li, "A hybrid cooperative coevolution algorithm for fuzzy flexible job shop scheduling," *IEEE Trans. Fuzzy Syst.*, vol. 27, no. 5, pp. 1008–1022, May 2019, doi: 10.1109/TFUZZ.2019.2895562.
[9] B. Liu, Y. Fan, and Y. Liu, "A fast estimation of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 87, pp. 193–201, Sep. 2015.
[10] 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.
[11] J. Cheng, F. Chu, and M. Zhou, "An improved model for parallel machine scheduling under time-of-use electricity price," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 896–899, Apr. 2018.
[12] J. F. Tang, C. K. Zeng, and Z. D. Pan, "Auction-based cooperation mechanism to parts scheduling for flexible job shop with inter-cells," *Appl. Soft Comput.*, vol. 49, pp. 590–602, Dec. 2016.
[13] K. Z. Gao, F. J. Yang, M. C. Zhou, Q. K. Pan, and P. N. Sugnathan, "Flexible job-shop rescheduling for new job insertion by using discrete Jaya algorithm," *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1944–1955, May 2019.
[14] K. Z. Gao, P. N. Suganthan, T. J. Chua, C. S. Chong, T. X. Cai, and Q. K. Pan, "A two-stage artificial bee colony algorithm scheduling flexible job-shop scheduling problem with new job insertion," *Expert Syst. Appl.*, vol. 42, no. 21, pp. 7652–7663, Nov. 2015.
[15] K. Z. Gao, P. N. Suganthan, Q. K. Pan, M. F. Tasgetiren, and A. Sadollah, "Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion," *Knowl.-Based Syst.*, vol. 109, pp. 1–16, Oct. 2016.
[16] N. Al-Hinai and T. Y. ElMekkawy, "Robust and stable flexible job shop scheduling with random machine breakdowns using a hybrid genetic algorithm," *Int. J. Prod. Econ.*, vol. 132, no. 2, pp. 279–291, Aug. 2011.
[17] Y. M. Wang, H. L. Yin, and K. D. Qin, "A novel genetic algorithm for flexible job shop scheduling problems with machine disruptions," *Int. J. Adv. Manuf. Technol.*, vol. 68, nos. 5–8, pp. 1317–1326, Sep. 2013.
[18] W. He and D.-H. Sun, "Scheduling flexible job shop problem subject to machine breakdown with route changing and right-shift strategies," *Int. J. Adv. Manuf. Technol.*, vol. 66, nos. 1–4, pp. 501–514, Apr. 2013.
[19] D.-H. Sun, W. He, L.-J. Zheng, and X.-Y. Liao, "Scheduling flexible job shop problem subject to machine breakdown with game theory," *Int. J. Prod. Res.*, vol. 52, no. 13, pp. 3858–3876, Jul. 2014.
[20] M. Nouiri, A. Bekrar, A. Jemai, D. Trentesaux, A. C. Ammari, and S. Niar, "Two stage particle swarm optimization to solve the flexible job shop predictive scheduling problem considering possible machine breakdowns," *Comput. Ind. Eng.*, vol. 112, pp. 595–606, Oct. 2017.
[21] H. Mokhtari and M. Dadgar, "Scheduling optimization of a stochastic flexible job-shop system with time-varying machine failure rate," *Comput. Oper. Res.*, vol. 61, pp. 31–45, Sep. 2015.
[22] E. Moradi S. M. T. F. Ghomi, and M. Zandieh, "An efficient architecture for scheduling flexible job-shop with machine availability constraints," *Int. J. Adv. Manuf. Technol.*, vol. 51, pp. 325–339, Apr. 2010.
[23] 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, Mar. 1999.
[24] A. Baykasoglu, "Linguistic-based meta-heuristic optimization model for flexible job shop scheduling," *Int. J. Prod. Res.*, vol. 40, no. 17, pp. 4523–4543, Jan. 2002.
[25] T. Jamrus, C.-F. Chien, M. Gen, and K. Sethanan, "Hybrid particle swarm optimization combined with genetic operators for flexible job-shop scheduling under uncertain processing time for semiconductor manufacturing," *IEEE Trans. Semicond. Manuf.*, vol. 31, no. 1, pp. 32–41, Feb. 2018.
[26] M. T. Jensen, "Generating robust and flexible job shop schedules using genetic algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 3, pp. 275–288, Jun. 2003.
[27] K. Z. Gao, P. N. Suganthan, Q. K. Pan, T. J. Chua, T. X. Cai, and C. S. Chong, "Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives," *J. Intell. Manuf.*, vol. 27, no. 2, pp. 363–374, Apr. 2016.
[28] M. Tao, Q. K. Pan, and H. Y. Sang, "A hybrid artificial bee colony algorithm for a flexible job shop scheduling problem with overlapping in operations," *Int. J. Prod. Res.*, vol. 56, no. 16, pp. 5278–5292, 2018.
[29] Y. Yuan and H. Xu, "Multiobjective flexible job shop scheduling using memetic algorithms," *IEEE Trans. Autom. Sci. Eng.*, vol. 12, no. 1, pp. 336–353, Jan. 2015.
[30] K. J. Yu, B. Y. Qu, C. T. Yue, S. L. Ge, X. Chen, and J. Liang, "A performance-guided JAYA algorithm for parameters identification of photovoltaic cell and module," *Appl. Energy*, vol. 237, pp. 241–257, Mar. 2019.
[31] 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.
[32] R. 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.
[33] C. Pradhan and C. N. Bhende, "Online load frequency control in wind integrated power systems using modified Jaya optimization," *Eng. Appl. Artif. Intell.*, vol. 77, pp. 212–228, Jan. 2019.
[34] S. O. Degertekin, L. Lamberti, and I. B. Ugur, "Sizing, layout and topology design optimization of truss structures using the Jaya algorithm," *Appl. Soft Comput.*, vol. 70, pp. 903–928, Sep. 2018.
[35] C. Huang, L. Wang, R. S.-C. Yeung, Z. Zhang, H. S.-H. Chung, and A. Bensoussan "A prediction model-guided Jaya algorithm for the PV system maximum power point tracking," *IEEE Trans. Sustain. Energy*, vol. 9, no. 1, pp. 45–55, Jan. 2018.
[36] S. P. Singh, T. Prakash, V. P. Singh, and M. G. Babu, "Analytic hierarchy process based automatic generation control of multi-area interconnected power system using Jaya algorithm," *Eng. Appl. Artif. Intell.*, vol. 60, pp. 35–44, Apr. 2017.
[37] R. C. Savaskan, S. Bhattacharya, and L. N. Van Wassenhove, "Closed-loop supply chain models with product remanufacturing," *Manage. Sci.*, vol. 50, no. 2, pp. 239–252, Feb. 2004.
[38] T.-C. Chiang and H.-J. Lin, "A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling," *Int. J. Prod. Economy*, vol. 141, no. 1, pp. 87–98, Jan. 2013.
[39] 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.
[40] J. Luo, Y. Yang, Q. Liu, X. Li, M. Chen, and K. Gao, "A new hybrid memetic multi-objective optimization algorithm for multi-objective optimization," *Inf. Sci.*, vols. 448–449, pp. 164–186, Jun. 2018.
[41] K. Z. Gao, P. N. Suganthan, Q. K. Pan, T. J. Chua, T. X. Cai, and C. S. Chong, "Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling," *Inf. Sci.*, vol. 289, pp. 76–90, Dec. 2014.
[42] P. Valledor, A. Gomez, P. Priore, and J. Puente, "Solving multi-objective rescheduling problems in dynamic permutation flow shop environments with disruptions," *Int. J. Prod. Res.*, vol. 56, no. 19, pp. 6363–6377, Oct. 2018.
[43] X. Guo, M. Zhou, S. Liu, and L. Qi, "Lexicographic multiobjective scatter search for the optimization of sequence-dependent selective disassembly subject to multiresource constraints," *IEEE Trans. Cybern.*, early access, Mar. 27, 2019, doi: 10.1109/TCYB.2019.2901834.
[44] C. Pan, M. Zhou, Y. Qiao, and N. Wu, "Scheduling cluster tools in semiconductor manufacturing: Recent advances and challenges," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 586–601, Apr. 2018.
[45] F. Qiao, Y. Ma, M. Zhou, and Q. Wu, "A novel rescheduling method for dynamic semiconductor manufacturing systems," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 5, pp. 1679–1689, May 2020, doi: 10.1109/TSMC.2017.2782009.
[46] Z. Zhao, S. Liu, M. Zhou, X. Guo, and L. Qi, "Decomposition method for new single-machine scheduling problems from steel production systems," *IEEE Trans. Autom. Sci. Eng.*, early access, Dec. 30, 2019, doi: 10.1109/TASE.2019.2953669.
[47] H. Yuan, J. Bi, M. Zhou, and A. C. Ammari, "Time-aware multi-application task scheduling with guaranteed delay constraints in green data center," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 3, pp. 1138–1151, Jul. 2018.
[48] H. Yuan, J. Bi, M. Zhou, Q. Liu, and A. C. Ammari, "Biobjective task scheduling for distributed green data centers," *IEEE Trans. Autom. Sci. Eng.*, early access, Jan. 7, 2020, doi: 10.1109/TASE.2019.2958979.