IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received October 27, 2018, accepted November 19, 2018, date of publication November 28, 2018,
date of current version January 4, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2883802

Hyper-Heuristic Coevolution of Machine Assignment and Job Sequencing Rules for Multi-Objective Dynamic Flexible Job Shop Scheduling

YONG ZHOU, JIAN-JUN YANG, AND LIAN-YU ZHENG
School of Mechanical Engineering and Automation, Beihang University, Beijing 100191, China

Corresponding author: Jian-Jun Yang (jjyang@buaa.edu.cn)

This work was in part by the Beijing Key Laboratory of Digital Design and Manufacturing, in part by the National High Technology Research and Development Programme (863) of China under Grant 2012AA040907, and in part by the Postgraduate Innovation Practice Base of the Modern Design and Advanced Manufacturing Technology for Complex Products at Beihang University.

ABSTRACT Nowadays, real-time scheduling is one of the key issues in cyber–physical system. In real production, dispatching rules are frequently used to react to disruptions. However, the man-made rules have strong problem relevance, and the quality of results depends on the problem itself. The motivation of this paper is to generate effective scheduling policies (SPs) through off-line learning and to implement the evolved SPs online for fast application. Thus, the dynamic scheduling effectiveness can be achieved, and it will save the cost of expertise and facilitate large-scale applications. Three types of hyper-heuristic methods were proposed in this paper for coevolution of the machine assignment rules and job sequencing rules to solve the multi-objective dynamic flexible job shop scheduling problem, including the multi-objective cooperative coevolution genetic programming with two sub-populations, the multi-objective genetic programming with two sub-trees, and the multi-objective genetic expression programming with two chromosomes. Both the training and testing results demonstrate that the CCGP-NSGAII method is more competitive than other evolutionary approaches. To investigate the generalization performance of the evolved SPs, the non-dominated SPs were applied to both the training and testing scenarios to compare with the 320 types of man-made SPs. The results reveal that the evolved SPs can discover more useful heuristics and behave more competitive than the man-made SPs in more complex scheduling scenarios. It also demonstrates that the evolved SPs have a strong generalization performance to be reused in new unobserved scheduling scenarios.

INDEX TERMS Scheduling, flexible job shop, hyper-heuristic, multi-objective, genetic programming.

NOMENCLATURE

MO-DFJSP	multi-objective dynamic flexible job shop scheduling problem
MAR	machine assignment rule
JSR	job sequencing rule
SP	scheduling policy
GEP	genetic expression programming
CCGP	cooperative coevolution genetic programming with two sub-populations
TTGP	genetic programming with single population that a GP individual contains two sub-trees
NSGAII	nondominated sorting genetic algorithm II
SPEA2	strength Pareto evolutionary algorithm 2

I. INTRODUCTION

With the development of information and communication technologies, many different kinds of sensors, automatic robots and data acquisition systems are installed in the shop floor [1]. Moreover, many new manufacturing concepts have been proposed in recent years including cyber-physical systems (CPS), intelligent manufacturing and cloud manufacturing [2]. These manufacturing concepts aim to increase factory productivity and efficient utilization of resources in real time. The increasing use of sensors, robots, and networked machines has resulted in continuous generation of real time data. This situation poses a major challenge to the current scheduling system because dynamic changes in the

shop floor require real-time responses [3]. Thus, real-time scheduling for dynamic manufacturing environments is one of the key issues in CPS production management [4].

The scheduling problem presented in this study arises in a real-world aeronautical manufacturing plant [5]. In this problem, there are several copies of the critical machines in the shop floor to increase production capacity. Thus, an operation can be processed on more than one machine and each of which has the same function. Therefore, it is similar to the flexible job shop scheduling problem [6].

In real production, scheduler need to react quickly to disruptions, using on-line scheduling by dispatching rules [7]. Such search-based techniques, which are time-consuming and tailored to the specific problem, are not applicable in dynamic or uncertain conditions and cannot react quickly to changing system requirements (i.e. new order arrivals, resource failures, cancellations of already handled jobs or changes in lot size). The dispatching rules which were made by experts are often used in the practical scheduling. This approach does not aim to generate (near) optimal schedules, but to generate those in acceptable quality and in a small amount of time. However, the solution quality generated by the man-made rules need to be further improved and the generalization performance also need to be enhanced to adapt to more complex scenarios.

The dynamic characteristics must be considered in this study because it is the essence of dynamic scheduling. We think that the dynamic characteristic of this problem includes two aspects. On the one hand, the dynamic characteristic of the MO-DFJSP considered in this study is new order arrivals. This is because jobs usually arrive over time and cannot be predicted before their arrivals. On the other hand, the dynamic characteristic of the MO-DFJSP is the dynamic factor during the solution process. The online computation time of the evolved rules generated by hyper-heuristics is similar to that of the heuristic methods. Although the training time for the MO-DFJSP is about a few hours, the training process can be carried out offline. For the practical scheduling problems of different enterprises, training process can be carried out when the workers get off work, and the rules obtained by training can be applied quickly online. This is also the motivation of this study. The authors want to develop a training system that can automatically evolve heuristics and replace the rules designed by experts to achieve better scheduling results. It will save the cost of expertise and facilitate large-scale applications.

To the best of our knowledge, there is no literature reported on the hyper-heuristic coevolution of MAR and JSR for the MO-DFJSP. In addition, the spirit of divide and conquer helps to reduce the complexity of the problem. Therefore, this study divides MO-DFJSP into two sub-problems: job sequencing and machine allocation. The proposed cooperative coevolution method evolves two kinds of rules simultaneously, so as to achieve the goal of co-optimization, search for more problem-related features, and improve the quality of evolved SPs. Therefore, we focus on the hyper-heuristic coevolution of the evolved SPs which can be applied to unseen scenarios by combining various small heuristic components. Four algorithms including multi-objective evolutionary algorithm, cooperative coevolution, genetic programming and genetic expression programming are integrated for heuristic generation to fabricate new SPs [8].

The motivation of this study is to generate effective SPs through off-line learning and to implement the evolved SPs online for fast application. In addition, the authors want to evolve a set of SPs with good generalization performance to be applied in practice, rather than find the optimal solution for the MO-DFJSP. Compared to the previous studies, our contributions are four aspects as follows.

1) Three types of methods (MO-CCGP, MO-TTGP and MO-GEP) are proposed to achieve effective machine selection and job sequencing decision making in the MO-DFJSP.
2) Five collaboration methods are appropriately designed to enhance the search space and quality of solutions.
3) An unsupervised learning framework is achieved to automatically evolve heuristics and replace the rules designed by experts.
4) Three Pareto dominance indicators between the evolved SPs and the 320 combinations of benchmark rule are defined, and the influence of the six experimental factors are investigated to explore the relationship between the parameters and the results.

The remainder of the paper is organized as follows. Section II provides a review of recent studies on automated design of heuristics for different types of production scheduling problems. The problem description of the MO-DFSJP is presented in Section III. In Section IV, the proposed algorithm is illustrated in detail, and the multi-objective performance measures for the algorithm are also provided. Section V presents and discusses the results of the empirical experiments. In Section VI, the generalization performance of the evolved SPs is validated by comparing with the benchmark SPs in new unobserved scenarios. Finally, conclusions and directions for future research are drawn in Section VII.

II. LITERATURE REVIEW

There has been a lot of research on shop scheduling, numerous techniques can be divided into the following types: heuristic, meta-heuristic, hyper-heuristic and artificial intelligence [9]. Heuristics named dispatching rules are frequently used in practice due to their ease of implementation, satisfactory performance, low computational requirement, and flexibility to incorporate domain knowledge [10], [11]. A dispatching rule is used to assign a priority index to each job waiting in the queue, and the job with the highest priority is selected for processing on the machine. In the case of machine selection, a priority index is assigned to each of the suitable machines which are capable of processing the needed operation. The machine with the highest priority is selected for processing the operation. The major drawbacks of dispatching rules include the performance of rules depend

on the state of the system and none of rules is superior to all others for all possible states [12].

Meta-heuristics are able to perform quite well and carry more knowledge of the problem domain, such as the ant colony optimization [13], artificial bee colony [14], artificial immune system [15], evolutionary algorithms [16], particle swarm optimization [17], tabu search [18] and variable neighborhood search [19]. Even if meta-heuristics have significantly attracted a lot of attentions from researchers, they still have two disadvantages: the first is that researchers have to design a problem-specific algorithm for each practical scheduling problem. That means they can only be used for the problem for which they have been developed. The second drawback is that most meta-heuristics are too time-consuming and cannot react quickly to changing system requirements in real-time scheduling [20].

In the context of solving various types of shop scheduling problems, many machine learning approaches have been applied on this subject [21]. These methods include evolutionary learning [22], gaussian processes [23], imitation learning [24], data mining [25], reinforcement learning [26], artificial neural-networks [27], fuzzy logic [28], ensemble learning [29], and artificial immune networks [30]. However, most of them belong to the category of supervised learning. Hence, training cases need to be carefully designed because they have a great impact on the test performance in supervised learning.

In recent years, the concept of hyper-heuristic has been proposed to solve the combinatorial optimization problems [31]. This approach has attracted attentions of many researchers in the field of operation research [32]. It refers to high-level iterative techniques, which guide a subordinate heuristic by using intelligent concepts to explore the search space of heuristics [33]. A hyper-heuristic algorithm strives to find near-optimal heuristics for the problem addressed in the search space of heuristics rather than in the search space of solutions [34]. There are many studies that using GP based hyper-heuristic to evolve dispatching rules for many production environments covering single machine scheduling, parallel machine scheduling, job shop scheduling and flexible job shop scheduling. Nguyen *et al.* [35] has reviewed the application of genetic programming (GP) in production scheduling and gave a unified algorithm design framework.

For solving the single machine scheduling problem, Giger and Uzsoy [36] developed a GP algorithm to automatically discover effective dispatching policies for batch processor scheduling. Nie *et al.* [37] proposed a gene expression programming-based scheduling rules constructor (GEPSRC) to automatically construct effective dispatching rules for single machine scheduling with job release dates. Jakobović and Marasović [38] addressed the problems in single machine and job shop scheduling environments, combined with several real-world properties including job weights, dynamic job arrivals, precedence constraints and sequence dependent setup times.

For solving the parallel machine scheduling problem, Đurasević *et al.* [39] proposed several different GP methods for evolving priority functions, like dimensionally aware GP, GP with iterative dispatching rules and GEP. He [40] also investigated four different ensemble learning approaches to improve the performance of GP for evolving dispatching rules, including simple ensemble combination, BagGP, BoostGP, and cooperative coevolution.

For solving the job shop scheduling problem, Nguyen *et al.* [41] proposed a new hybrid genetic programming algorithm for dynamic job shop scheduling based on a new representation, a new local search heuristic, and efficient fitness evaluators. Results show that the new method is effective regarding the quality of evolved rules which are significantly smaller and contain more relevant attribute. Nguyen *et al.* [42] also proposed a hyper-heuristic method based on genetic programming to solve the multi-objective job shop scheduling problem. Pickardt *et al.* [43] proposed a two-stage approach for the semiconductor factory scheduling problem. GP was used to evolve the job sequencing rules, which would be grouped into candidate collections with standard rules from which good rules would be selected by evolutionary computation (EA) and arranged for corresponding machines. Zhang and Roy [44] proposed a semantics-based dispatching rule selection approach for job shop scheduling.

For solving the flexible job shop scheduling problem, Tay and Ho [45] proposed a GP method to generate dispatching rules to solve the multi-objective FJSP. The disadvantage of this study is that the least waiting time (LWT) is used as the benchmark machine assignment rule to find a suitable machine to process an operation. However, only one machine assignment rule is not enough for the deeply research of MO-DFJSP. Nie *et al.* [46] used gene expression programming (GEP) to generate reactive scheduling policies. The encoded chromosome in GEP consists of two parts for two sub-problems, i.e., the job routing problem and the job sequencing problem. In the GEP-based approach, two decision rules are encoded into a chromosome, which make them evolve simultaneously. However, three measure criteria are separately optimized in this paper. Zhang *et al.* [47] developed an efficient Gene Expression Programming (eGEP) algorithm for generating rules automatically to achieve effective machine selection, job sequencing, and machine off-on decision making. The single objective function is to minimize the total energy consumption. However, only the dispatching rules are evolved in this study and the interaction effect between the dispatching rules and the machine assignment rules is neglected in this research. Yska *et al.* [48] proposed a cooperative coevolution framework to co-evolve the routing and sequencing rules together using for FJSP. Yska *et al.* [49] focus on feature construction to improve the effectiveness and efficiency of GPHH. Zhang *et al.* [50] proposed to evolve routing and sequencing rules based on GP with multi-tree representation. Zhang *et al.* [51] proposed two different kinds of strategies of surrogates for GP to automatically design dispatching rules for DFJSP. However, three objectives were

optimized separately in these four studies, which is a single objective optimization problem. In addition, the fittest selection is used as the collaboration scheme in the CCGP. It should be noted that although each situation is solved optimally, this does not always lead to an optimal overall solution.

To conclude, hyper-heuristic coevolution of MAR and JSR for the MO-DFJSP, which is a more general problem, has not been analysed in prior studies. Therefore, our study serves as the first attempt to solve this problem.

III. PROBLEM DESCRIPTION

A. PROBLEM FORMULATION

The MO-DFJSP involves two subproblems: assign each operation to an appropriate machine (machine assignment); sequence the operations on each machine (operation sequence). Job shop scheduling problem has been proven to be NP-hard [52]. FJSP as an extended problem, is also be a NP-hard problem [53]. The MO-DFJSP with functionally related machines is formulated as follows.

Indices:

i, h: index of jobs, $i, h \in \{0, 1, 2, \ldots, n\}$

j, l: index of operations in a given job, $j, l \in \{0, 1, 2, \ldots, k\}$

k, g: index of machines, $k, g \in \{0, 1, 2, \ldots, m\}$

Parameters:

n: number of jobs

m: number of machines

n_i: total number of operations of job J_i

$o_{i,j}$: the jth operation of job J_i

$p_{i,j,k}$: processing time of operation $o_{i,j}$ on machine m_k

$\overline{p_{i,j}}$: the mean processing time per operation is defined as $\overline{p_{i,j}} = \frac{\sum_{k=1}^{cm(o_{i,j})} p_{i,j,k}}{cm(o_{i,j})}$, where $cm(o_{i,j})$ denotes the number of candidate machines that can process the operation $o_{i,j}$

$M_{i,j}$: set of available machines for the operation $o_{i,j}$ of job J_i. If $M_{i,j} \subset M$ for at least one operation, it is a partial flexibility FJSP (P-FJSP); while $M_{i,j} = M$ for each operation, it is a total flexibility FJSP (T-FJSP). According to Kacem *et al.* [54], for the same number of machines and jobs, a P-FJSP is more difficult to solve than a T-FJSP. Hence, all experiments designed in this study are based on the simulation model of a P-FJSP to test the generalization performance of the evolved SPs.

w_i: the weight of job J_i

r_i: the release time of job J_i

d_i: the due date of J_i is defined as $d_i = r_i + c \cdot \sum_{j=1}^{n_i} \overline{p_{i,j}}$

c: the tightness factor of the due time d_i of job J_i

C_i: the completion time of job J_i

F_i: the flow time of job J_i is defined as $F_i = C_i - r_i$

T_i: the tardiness of job J_i is defined as $T_i = \max\{0, C_i - d_i\}$

U_g: the utilization of shop floor

$\overline{t_{av}}$: the average interval time between the jobs arrived at the shop floor is defined as $\overline{t_{av}} = \frac{\overline{p_{i,j}} \cdot \overline{num(o_{i,j})}}{m \cdot U_g}$, where $\overline{num(o_{i,j})}$ denotes the average operation number of jobs

Decision Variables:

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

$c_{i,j,k}$ = completion time of operation $o_{i,j}$ on machine m_k

Three objective functions are simultaneously minimized:

$$\min WT_{mean} = 1/n \cdot \sum_{i=1}^{n} w_i \cdot T_i \quad (1)$$

$$T_{max} = \max\{T_i |\ i = 1, \ldots, n\} \quad (2)$$

$$F_{mean} = 1/n \cdot \sum_{i=1}^{n} F_i \quad (3)$$

and the constraints are

$$\text{s.t. } x_{i,j,k} \in \{0, 1\}, \quad \forall i, j, k \quad (4)$$

$$c_{i,j,k} > 0, \quad \forall i, j, k \quad (5)$$

$$c_{i,j,k} - c_{i,j-1,g} \geq p_{i,j,k} \cdot x_{i,j,k}, \quad \forall i; j \in \{2, 3, \ldots, n_i\}; \quad \forall k; \ \forall g \quad (6)$$

$$\sum_{k \in M_{i,j}} x_{i,j,k} = 1, \quad \forall i, j, k \quad (7)$$

$$c_{i,j,k} - c_{h,l,k} \geq p_{i,j,k} \cdot x_{i,j,k} \cdot x_{h,l,k}, \quad \forall i, j, k, h, l \quad (8)$$

$$M_{i,j} \subset M, \quad \forall i, j \quad (9)$$

Equations (1)-(3) are used to minimize the mean weighted tardiness (WT_{mean}), maximum tardiness (T_{max}) and mean flow time (F_{mean}), respectively. Equations (4) and (5) are variable restrictions. Equation (6) ensures the operation precedence constraint. Equation (7) states that only one machine could be selected from M_{ij} for one operation. Equation (8) ensures that two operations are not overlapping if both of them are assigned on the same machine. Equation (9) indicates that M_{ij} for each operation comes from the given machine set M.

For the sake of understanding, a simple instance of MO-DFJSP is showed in Table 1, which is to execute three jobs on three machines. Each cell of the table denotes the processing time of the operation on the corresponding machine. The two numbers (x, y) in column 'Job' mean that the release time of job J_i is 'x', and the due time is 'y'. The symbol "–" means that the machine cannot process the corresponding operation. It should be noted that the release time of each job is different from each other, and the tightness factor of each job is set to 2 in this example.

TABLE 1. An instance of the MO-DFJSP.

Job	Operation	M_1	M_2	M_3
J_1 (0,12)	O_{11}	1	3	-
	O_{12}	3	-	5
J_2 (8,34)	O_{21}	-	3	4
	O_{22}	1	10	-
	O_{23}	3	5	4
J_3 (10,26)	O_{31}	2	6	-
	O_{32}	4	-	-

B. SIMULATION MODEL

In this study, a dynamic flexible job shop simulation model is employed to evaluate the quality of the scheduling policies. Below is the simulation assumption:

- Each machine is continuously available for production, i.e., no machine breakdowns.
- There is no restriction on queue length at any machine, i.e., buffer unlimited.
- There is no travel time between machines. Jobs are available for processing on a machine immediately after completing processing its previous operation.
- The machine setup time for two consecutive jobs is included in the processing time.
- Job arrivals follow Poisson process.
- Weights of jobs are assigned based on the 4:2:1 rule according to Pinedo's study [55], which showed that 20% of the orders are particularly important, 60% are generally average important and 20% are less important.
- For the distribution of the number of operations, the *missing* setting is used to indicate that the number of operations will follow a discrete uniform distribution from 1 to the number of machines. Meanwhile, the *full* setting indicates that each job will have its number of operations equal to the number of machines in the shop.
- The process times of an operation on a machine follow two types of distributions in the testing scenario, one follows discrete uniform distribution U[1, 99] and the other follows normal distribution N(120, 20).
- An operation of a job can be processed on a subset of machines. The *optional device number* setting denotes the number of machines could be selected for each operation, which implies that how many machines are available to process the operation in the shop. Two settings are used in the testing scenario, one setting follows discrete uniform distribution U[1, 2], the other setting follows discrete uniform distribution U[1, 4].

As shown in Table 2, we applied a design of experiments (DOE) approach to design both the training and testing scenarios. The full factorial design is adopted in this case. In the DOE, three factors with two levels are used to construct $2^3 = 8$ scenarios in the training set. Six factors with two levels are used to construct $2^6 = 64$ test scenarios in the test set. The discrete event simulation model of the MO-DFJSP is used to evaluate fitness of the evolved SPs. The simulation runs for a sufficiently long period after the shop reaches the steady state. In each simulation replication, we begin with an empty shop. The interval from the beginning of the simulation until the arrival of the 500th job is considered as the warm-up time, and the statistics from the 500th job to the next completed 2500 jobs will be used to calculate the simulation performance.

TABLE 2. Parameter settings of the training and testing scenarios.

Parameter	Training	Testing
Weights of jobs	4:2:1(20%:60%:20%)	
Simulation Length	2500	
Warmup Length	500	
Machine number	10	10, 20
Operation number distribution	Missing	Missing, Full
Process time distribution	U[1,99]	U[1,99], N(120,20)
Allowance factor	2, 4	1, 3
Utilization	70%, 90%	80%, 95%
Optional device number	1, U[1,3]	U[1,2], U[1,4]

IV. ALGORITHM DESIGN

The scheduling policy is evolved by three methods. The first approach employs the cooperative coevolution genetic programming with two multi-objective approaches (CCGP-NSGAII/ CCGP-SPEA2) to evolve two decision rules in two separate populations. In the second approach, only one population with two multi-objective approaches is used to evolve two decision rules in two sub-trees (TTGP-NSGAII/ TTGP-SPEA2). The third approach employs the multi-objective genetic expression programming with two chromosomes (MO-GEP). Because the experiment found that MO-GEP performs more efficient than MOGP, which means that the running time of MO-GEP is shorter than that of MOGP under the same number of function evaluations (NFEs). To make a relative fair comparison, GEP-NSGAII is divided into two methods: one is the GEP-N-NSGAII which runs for the same NFEs as the GP-based algorithms, the other is GEP-T-NSGAII which runs for the similar computational time as the GP-based algorithms.

A. FRAMEWORK

As shown in Figure 1, the framework of the automated heuristic design approach contains two parts: the hyper-heuristic based policies generation and the simulation-based fitness evaluation. A scheduling policy (SP) which is used in the MO-DFJSP includes two decision rules: a JSR and a MAR. Firstly, when a new job arrival or an operation is completed, the MAR calculates the priority index of each candidate machine and selects the machine with the highest priority to be arranged. And then, when a machine becomes idle, the job sequencing rule calculates the priority index of each waiting operation in its queue and determines the operation with the highest priority to be processed next.

In the evolution stage, GP [56] and GEP [57] is employed as the learning mechanism to evolve SPs for the MO-DFJSP. Three types of methods are employed in this stage. In the evaluation stage, two individuals from two separate populations in MO-CCGP (two sub-trees from an individual in MO-TTGP, two chromosomes from an individual in MO-GEP) are collaborated to formulate a complete SP, which is decoded into the JSR and MAR, and then they are applied to the relevant decision points in the discrete-event simulation model. When the simulation finished, the results are collected and returned to assign fitness to the individual for GP evolution.

B. CHROMOSOME REPRESENTATION

GP is a special kind of evolution algorithm that is characterized by its ability to evolve individuals of variable lengths,

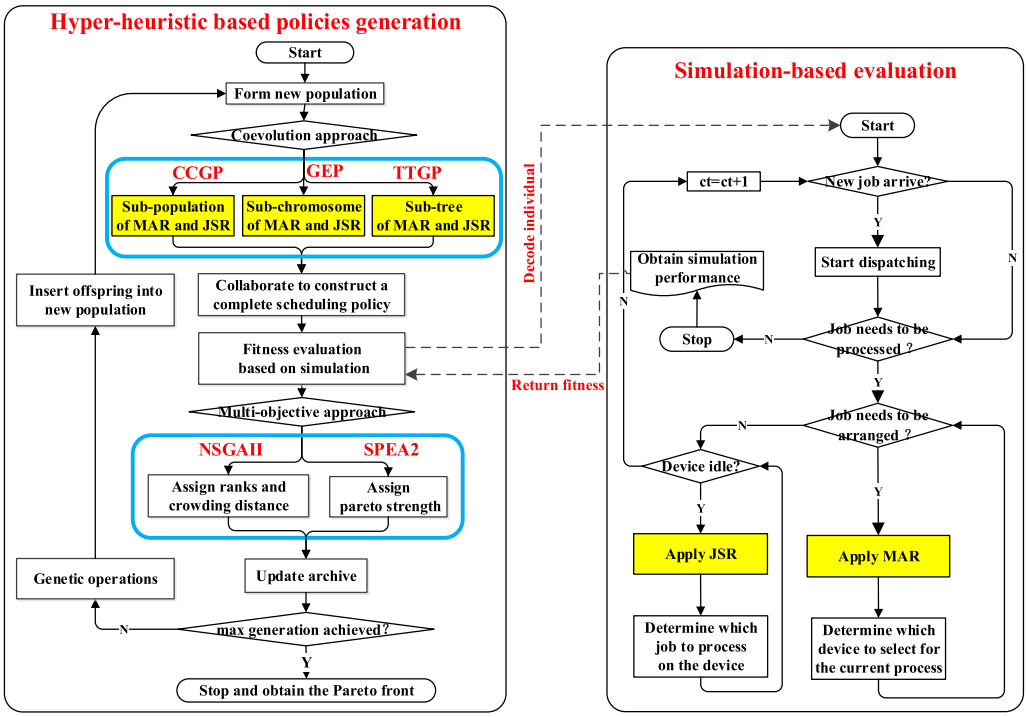


FIGURE 1. Framework of the hyper-heuristic based policies generation and the simulation-based fitness evaluation.

where solution candidates are encoded as tree structures [35]. GEP is a successor of GA and GP. GEP uses fixed length, linear strings of chromosomes to represent expression trees of different shapes and sizes, which makes GEP more versatile than GP [57]. In this study, GP and GEP is employed to mine more efficient SPs by establishing the corresponding relation between the mathematical expressions and the individuals.

1) ENCODING OF INDIVIDUALS

In the MO-CCGP methods, there are two sub-populations which denotes two types of decision rules. Each CCGP individual is represented by a tree form of various lengths, and each individual can only be encoded in one type of rules. There is only one population in MO-TTGP, and each individual contains two types of trees for the two decision rules. In the MO-GEP methods, each GEP individual consists of two chromosomes for the two decision rules, and each chromosome is represented by a fixed length string of genes. Different from other evolution algorithms, typical applications of GP/GEP are the automatic creation of mathematical formulas or computer programs for solving a specific task. Therefore, it is rather obvious to use GP/GEP for the generation of composite SPs.

There are two types of symbol sets used to construct a GP tree: function set and terminal set. The function set consists of basic operators (+, −, ×, /, max, min). The function '/' is the protected division, which returns 1 if the denominator is 0. Based on the literature, we carefully design the terminal set for two decision rules which are presented in Table 3 and Table 4, respectively. The selection of suitable terminals is generally problem-specific.

TABLE 3. Terminal set for JSR.

Terminals	Description
PT	processing time of operation $O_{i,j}$.
OL	number of operations left for job J_i
RPT	work remaining of job J_i
TIS	time spend in system of job J_i (current time- r_i)
TIQ	time spent in the device queue of job J_i
TDD	time to the due date of job (d_i -current time)
SLACK	the difference between job's remaining delivery time and its remaining work time=TDD-RPT
TOD	time to operation's due date
W	weight of job J_i
ERC	Random number generating from (0, 1.0]
NPT	processing time of the next operation $O_{i,j+1}$
WINQ	total work time in the next queue.

Figure 2 shows an example of the encoding of a job sequencing rule(2PT+WINQ+NPT) as a GP tree structure and a GEP chromosome representation. According to

TABLE 4. Terminal set for MAR.

Terminals	Description
WQT	total work time of the job in the device queue
WQS	queue size of the device
WQD	total queueing time of the device
WD	total idle time of the device
WPT	processing time of the job on the device
WTW	total work time of the device
WRL	mean queue size of the device
ERC	Random number generating from (0, 1.0]

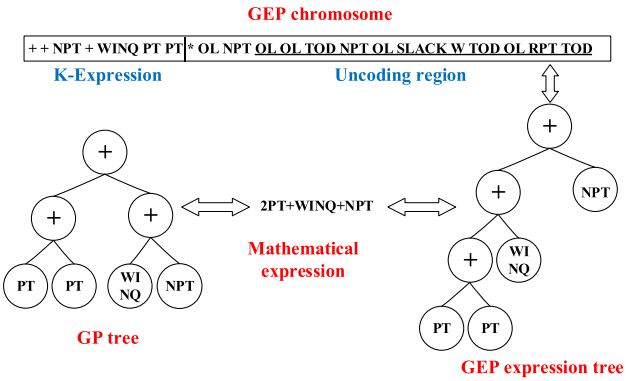


FIGURE 2. GP Tree and GEP chromosome representation of an exemplary JSR.

the Krava language which is designed by Ferreira [57], the lengths of head and tail in Figure 2 are 10 and 11 respectively. It is calculated by the equation t= h(n−1)+1, where h and t are the lengths of the head and the tail of each part respectively, n is the maximum number of arguments for all the operators in functions set. The sequence of symbols is translated into a GEP expression tree. Then, the GP/GEP tree structure is decoded into a mathematical expression, and the priority value of this representation is calculated according to the current system.

2) GENETIC OPERATIONS

After all individuals have been evaluated by the simulation model, the archive will be updated by the two multi-objective approaches (NSGAII [58], SPEA2 [59]) to explore the Pareto front of nondominated SPs. Then, the breeding step is realized using the genetic operations.

GP individuals from the current population are selected using the double tournament selection [60] to hand bloat. This method does a tournament selection based on fitness. But the individuals entering the tournament are not from the general population but from other tournament selection operations which were based on smallness. In this study, the tournament size for fitness t1 is set to 7 and the tournament size for smallness t2 is set to 2 according to [60]. After the double tournament selection process, new individuals are created using genetic operations.

An instance of subtree crossover and mutation in MO-CCGP is shown in Figure 3, the subtree crossover recombines subtrees from two selected parents by randomly picking a node in each individual and swapping over the connecting subtrees, thereby producing two new individuals. The subtree mutation is performed by selecting a node of a chosen parent and replacing the subtree rooted by that node with a newly randomly-generated subtree.

The genetic operations in the proposed MO-TTGP are different from the traditional operations. Because the traditional individual only contains one type of genetic material, but the individual in MO-TTGP contains two types of GP trees. There are three steps of crossover operation in the MO-TTGP. Firstly, the crossover operator randomly selects two individuals R_i and R_j. Then, a sub-tree(S-R_i) which belongs to one type of GP trees in the individual R_i is randomly selected. Finally, S-R_i is swapped with a randomly selected sub-tree in R_j (S-R_j) which belongs to the same type of GP trees. There are also three steps of mutation operation in MO-TTGP. Firstly, the mutation operator randomly chooses one individuals R_i. Then, a sub-tree(S-R_i) which belongs to one type of GP trees in the individual R_i is randomly selected. Finally, S-R_i is replaced by a newly randomly-generated subtree (S_n-R_i) which belongs to the same type of GP trees.

The MO-GEP method utilizes iteratively the genetic operations including selection, mutation, transposition and recombination. Roulette wheel selection is adopted here to select individuals according to fitness. To enrich the diversification

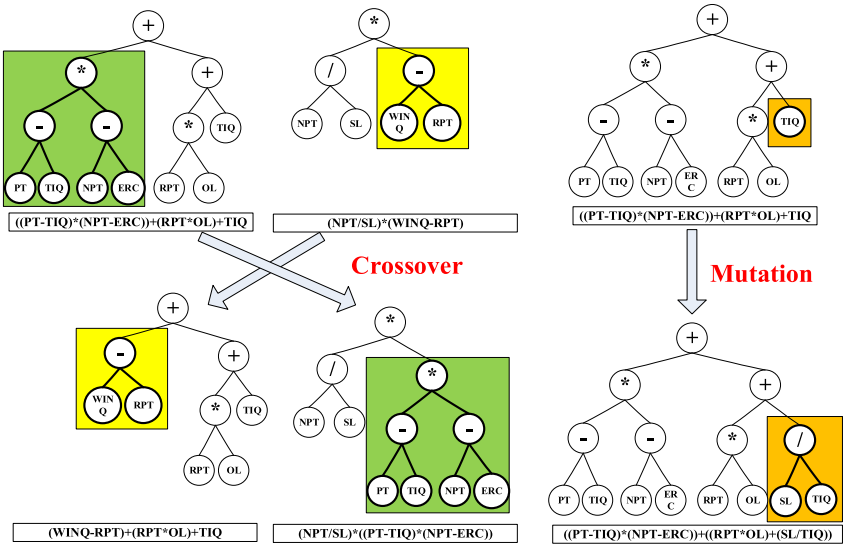


FIGURE 3. An instance of subtree crossover and mutation in MO-CCGP.

of population, mutation tends to produce perturbations on the rules by randomly changing symbols in a chromosome. The principle of transposition is randomly choosing a fragment of a chromosome and insert it in the head. There are two types of transposition operator applied in this study, IS transposition and RIS transposition. Recombination can keep the favourable fragments in the next generation by exchanging some material between two randomly chosen chromosomes. There are two kinds of recombination used in this study: one-point recombination and two-point recombination. More details can refer to [46].

It should be stressed that only genetic materials from the same type of selected trees will be exchanged. For example, a tree representing JSR in a parent will only crossover with a tree representing JSR of the other parent. This is because the terminals used in the sub-population/tree/chromosome of JSR are different from the sub-population/tree/ chromosome of MAR.

C. PROCEDURES OF PROPOSED ALGORITHMS

The proposed MO-CCGP algorithms including CCGP-NSGAII and CCGP-SPEA2 can be described as follows.

Step 1: The initial GP individuals are randomly generated by using ramped half-and-half method (depth 2 to 6). The MO-CCGP method starts with two randomly generated sub-populations, one sub-population Pop for JSR and the other sub-population Pws for MAR.

Step 2: The individual Ri op from the sub-population Pop is paired with the individual Rj ws from the sub-population Pws using random shuffling. Rk rep is the complete SP that is formed by the combination of (Ri op, Rj ws).

Step 3: In the fitness evaluation stage, eight training scenarios O (more details are shown in Table 2) are loaded to evaluate the performance of a complete SP Rk rep. The fitness of (Ri op, Rj ws) is obtained by applying Rk rep to O using one replication [61], and it is measured by the average value of the specific objective across all training scenarios. The simulation results are collected and returned to assign fitness to the individual (Ri op, Rj ws).

Step 4: After all individuals have been evaluated, the archive A will be updated according to the specific multi-objective approach. To explore the Pareto front of nondominated SPs, two multi-objective approaches are employed to assign ranks and crowding distance (NSGAII) or Pareto strength (SPEA2).

Step 5: The multi-objective performances (HVR, IGD, Spacing) of the Pareto front in the current generation are calculated according to the reference Pareto front PF_{ref}. In this study, PF_{ref} is extracted from all Pareto fronts found by the six proposed methods in 30 independent runs.

Step 6: If the maximum generation is not reached, new sub-populations are generated by the double tournament selection, subtree crossover and subtree mutation. After the genetic operations have been done, the algorithm starts a new generation

Step 7: If the maximum generation is reached, fast-nondominated-sort method is applied to the last generation of individuals to obtain the Pareto front PF_{known}.

Algorithm 1 CCGP-NSGAII/ CCGP-SPEA2

Input: simulation model $O \leftarrow \{O_1, \ldots, O_T\}$
Output: the Pareto front of nondominated SPs PF_{known}

(1) initialize population Pop←{R1 op, …, Rn op}, Pws←{R1 ws,…,Rn ws}
(2) generation←0, archive A← {}
(3) **while** generation ≤ max Generation do
(4) pair up the Ri op, Rj ws using random shuffling
(5) **for** all Ri op∈P_{op} do
(6) Rk rep ← collaborate (Ri op, Rj ws)
(7) evaluate f(Rk rep) by applying Rk rep to O using 1 rep
(8) f(Ri op), f(Rj ws) ← f(Rk rep)
(9) **end for**
(10) assign ranks and crowding distance (NSGAII) or Pareto strength (SPEA2) to build archive A
(11) calculate the multi-objective metrics of the Pareto front in the current generation according to the reference Pareto front PF_{ref}
(12) apply genetic operations to archive A to generate new population
(13) generation ← generation + 1
(14) **end while**
(15) apply fast-nondominated-sort to the last generation of individuals to obtain the Pareto front PF_{known}.
(16) **return** PF_{known}

The proposed MO-TTGP methods which includes TTGP-NSGAII and TTGP-SPEA2, are similar to the MO-CCGP methods. The difference is that an individual in MO-TTGP contains two sub-trees for two decision rules when decoding for fitness evaluation. In this case, each individual in MO-TTGP is equivalent to a complete SP.

The representation and the genetic operations of MO-GEP are different from that of GP-based algorithms. However, the overall process of the MO-GEP is similar to MO-TTGP, which is not described in detail here.

D. PARAMETER SETTINGS

There are many methods for choosing representatives with which to collaborate in CCGP [70]. Therefore, we conducted experiments on five collaboration schemes based on the CCGP-NSGAII method. Before introducing the experiment, we need to clarify three questions: how to choose collaborators, how many collaborators to choose, and how to assign fitness when there are multiple collaborations.

The first question can be described as the collaborator selection pressure. It is also the selection strategy of the collaboration scheme. The second question can be called as the collaboration pool size. It defines the number of collaborators per subpopulation to use for a given fitness evaluation.

Algorithm 2 TTGP-NSGAII/ TTGP-SPEA2

Input: simulation model $O \leftarrow \{O_1, \ldots, O_T\}$

Output: the Pareto front of nondominated SPs PF_{known}

(1) initialize population P at random, $P \leftarrow \{R_1, R_2..R_n\}$
(2) generation $\leftarrow$0, archive A $\leftarrow$ {}
(3) **while** generation $\leq$ max Generation do
(4) **for** all $R_i \in P$ do
(5) evaluate f(R_i) by applying R_i (Rop i, Rws i) to O use 1 rep
(6) **end for**
(7) assign ranks and crowding distance (NSGAII) or Pareto strength (SPEA2) to build archive A
(8) calculate the multi-objective metrics of the Pareto front in the current generation according to the reference Pareto front PF_{ref}
(9) apply genetic operations to archive A to generate new population
(10) generation $\leftarrow$ generation + 1
(11) **end while**
(12) apply fast-nondominated-sort to the last generation of individuals to obtain the Pareto front PF_{known}.
(13) **return** PF_{known}

There is no standard answer to the above two questions, so it is necessary to conduct experiments on this problem studied in this study. The third question can be defined as the collaboration credit assignment. A clear answer to this question is given in [70]. It is evident that using an optimistic approach is generally the best mechanism for collaboration credit assignment. Optimistic means assigning an individual a fitness score equal to the value of its best collaboration. Therefore, we use this optimistic approach for collaboration credit assignment.

To make a relative fair comparison, all these collaboration schemes use the same NFEs. The parameter configurations of the collaboration scheme are shown in Table 5. There are two parameter settings, one setting fixed population size and changed evolutionary generations, and the other setting fixed evolutionary generations and changed population size. For example, '100-30' means the population size is fixed to 100 and the generation is set to 30. In addition, 'n' denotes the population size of each sub-population. This is because the CCGP-NSGAII method evaluate both the parent and child individuals in the evaluation process, the number of fitness evaluations per generation is 100×2=200(except for the first generation), and the total number of fitness evaluations is 200×30=6000. This setting will be used as a standard NFEs of the collaboration schemes. Since the experiment was conducted under the same algorithm framework CCGP-NSGAII, and most of the running time was consumed in the evaluation process. Therefore, the NFEs represents the running time, and the running time of each collaboration scheme under the same NFEs is roughly equal. Parameter P1 ensures that the same NFEs can be achieved under the condition of the same population size. Similarity, parameter P2 ensures that the same NFEs can be achieved under the condition of the same generations.

TABLE 5. Parameter settings of the collaboration scheme.

Collaborator selection pressure	pool size	NFEs/ Gen	P1	P2
Complete Selection (CS)	n	$4n^2$	/	/
Random Shuffling (RSH)	1	2n	100-30	100-30
Random Selection (RS)	1	4n	100-15	50-30
Tournament Selection (TS)	1	4n	100-15	50-30
Fittest Selection (FS)	1	4n	100-15	50-30
Fittest Random Selection(FRS)	2	8n	100-8	25-30

Complete selection method performs exhaustive pair-wise evaluations, applying each individual in one population to each in the other population. Such approach can be computationally expensive in multi-population models, since the NFEs used for each assessment of fitness is $4n^2$ (except for the first generation). Therefore, its NFEs is 40,000 which is much more than 6,000, so this study did not use this collaboration method.

The two collaboration methods of random shuffling and random selection randomly select one individual from the other sub-population to pair with the current individual, and then conduct the fitness evaluation of the combined individual. The difference between these two collaboration schemes is that random shuffling method simultaneously assigns fitness to both individuals, but the random selection method only assigns fitness to the current individual but the matched individual is not assessed. Therefore, the random shuffling method needs only to be evaluated $2n$ times per generation, but the random selection method needs to be evaluated $4n$ times per generation. In addition, these two collaboration methods have no relationship with fitness.

There are three collaboration methods which have relationship with fitness. They are tournament selection, fittest selection, fittest and random selection. The tournament selection method chooses the individual by tournament selection from alternative sub-population, the tournament size is set to 2 in this experiment. In the fittest selection, an individual is always combined with the fittest individual from each of the other sub-populations. Similar to random selection, both tournament selection and fittest selection needs to be evaluated $4n$ times per generation. In fittest and random selection, two individuals (best and random) are selected to pair with the individual from the other sub-populations. Therefore, the NFEs for this collaboration method is $8n$ per generation.

All these collaboration methods were applied to the same training set as described in Table 2 in this study. Comparison of multi-objective performance indicators in five collaboration methods under the parameter setting of P1 are shown below. The experimental results under the parameter setting of P2 are similar to P1, so they are not described here. As shown in Figure 4, the pink circle denotes the average result and the red horizontal line represents the median result of the corresponding algorithm. We can observe that the

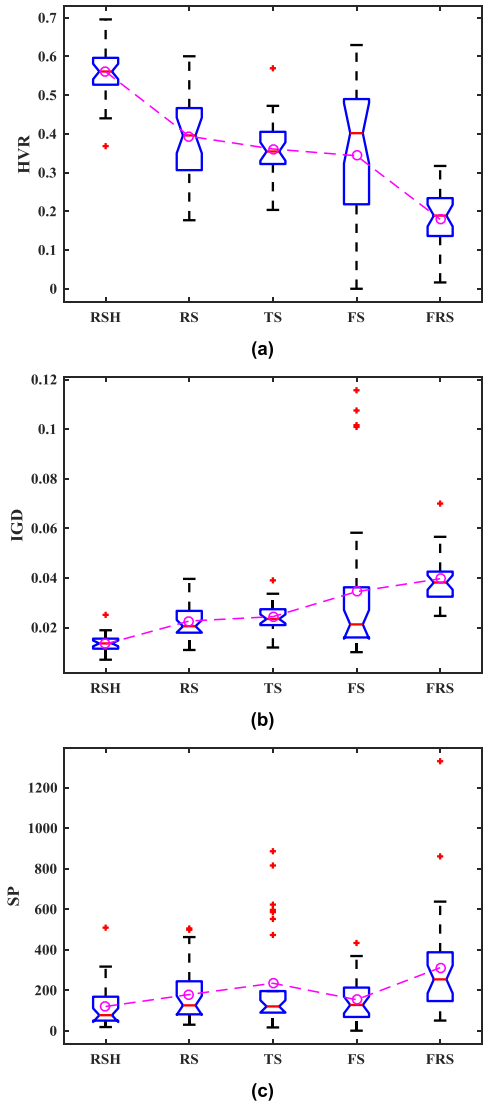


FIGURE 4. Performances of the collaboration methods in training scenarios under the parameter setting of P1 (a) HVR (b) IGD (c) Spacing (HVR to be maximized, Spacing and IGD to be minimized).

metrics of HVR, IGD and Spacing produced by random shuffling method are significantly better than other collaboration methods. For the HVR and IGD metrics, the result is that the RSH>RS>TS>FS>FRS ('>' means 'is significantly better than', '≥' means 'is not bad than'). In term of Spacing, the result is that the RSH>RS≥FS>TS>FRS.

We can observe that each generation of RSH has less NFEs than other collaboration methods. Under the same NFEs conditions, RSH has evolved more generations and thus obtained a better pareto solution set than other four methods. The fittest selection method (FS) evolves each decision rule using the best individual from the other sub-population. This is quite similar to the family of numerical optimization techniques which proceed by optimizing one function variable at a time while holding the other variables constant. It is well known that such procedures work well on functions whose variables are reasonably independent, but have difficulties with functions with interacting variables. In addition, it should be noted that although each situation is solved optimally, this does not always lead to an optimal overall solution [71]. In conclusion, the random shuffling performed best among the five collaboration methods, and it was used in this study.

Table 6 shows the parameter settings of the proposed MO-CCGP and MO-TTGP algorithms. These parameters are adjusted carefully based on our preliminary experiments, and the results show that these parameter settings perform well in both effectiveness and robustness. To make a relative fair comparison, all GP-based methods proposed in this study use the same NFEs. For the NSGAII-based methods, generation is set to 50 and the population size is fixed to 200. Because the NSGAII-based methods evaluate both the parent and child individuals in the evaluation process, the number of fitness evaluations per generation is 200×2=400, and the total number of fitness evaluations is 50×400=20000. For the SPEA2-based methods, generation is set to 100 and the population size is fixed to 200. Therefore, the number of fitness evaluations per generation is 200, and the total number of fitness evaluations is 100×200=20000. These settings are applied to achieve the same NFEs for fair comparison.

TABLE 6. Parameter settings of the MO-CCGP and MO-TTGP methods.

Parameter	CCGP	TTGP
Population Number	2	1
Population Size	200	200
Generation	NSGAII-based: 50; SPEA2-based: 100	
Crossover/mutation rates	90%/10%	
Collaboration method	random shuffling	
Initialization	ramped half-and-half (depth 2~6)	
Selection	tournament selection (size 5)	
Parsimony Pressure	double tournament selection (t1=7, t2=2)	
Maximum depth	8	
Operators	+, −, ×, /, max, min	
Terminals	Shown in Table 3 and Table 4	

In addition, five different archive size settings (20,50,100, 150 and 175) for SPEA2-based methods are used to investigate the impact on the three multi-objective performance metrics. Based on our preliminary experiments, the best setting that achieve high performance is 100.

As shown in Table 7, these parameters are set as recommended in [46]. It should be noted that the termination condition for the GEP-N-NSGAII is set to 50 generations, it is to ensure the same NFEs as the GP-based algorithms. The running time of GEP-T-NSGAII is set to 67.5 minutes, which is the average running time of the GP-based algorithm. It is to ensure the similar computational time as the GP-based algorithms.

E. PERFORMANCE METRICS

Three popular metrics are employed to evaluate the performances of the proposed methods: Hypervolume Ratio (HVR) [62], Inverted Generational Distance (IGD) [63], and Spacing [64]. They can be expressed as follows:

- Hypervolume ratio (HVR): hypervolume is used to measure the size of the objective space dominated by the

TABLE 7. Parameter settings of the MO-GEP methods.

Parameter	GEP-N-NSGAII	GEP-T-NSGAII
Termination condition	Generation:50	Time:67.5 min
Population size	200	
Population number	1	
Chromosome number	2	
Length of head	10	
Selection strategy	Roulette wheel sampling	
Mutation probability	0.05	
IS transposition probability	0.1	
RIS transposition probability	0.1	
One-point recombination probability	0.2	
Two-point recombination probability	0.5	

obtained non-dominated front PF_{konwn}. A higher HV value is desirable and denotes a good dominate performance.

$$HV = volume(\bigcup_{i=1}^{n_{PF}} v_i) \tag{10}$$

Where n_{PF} is the number of members in the obtained non-dominated front PF_{konwn}, v_i is the hypercube constructed with a reference point and the member i as the diagonal of the hypercube [58]. HVR is the ratio of the HV of PF_{konwn} and the HV of the reference Pareto front PF_{ref}.

$$HVR = \frac{HV(PF_{konwn})}{HV(PF_{ref})} \tag{11}$$

- Inverted Generational Distance (IGD): This is a variant of the Generational Distance (GD) and represents a combined or comprehensive indicator. It measures the average distance from the reference Pareto front PF_{ref} to Pareto front PF_{konwn} obtained by the algorithm.

$$IGD = \frac{(\sum_{i=1}^{n} d_i^p)^{1/p}}{n} \tag{12}$$

Where n is the number of all elements in PF_{ref}, p is set to 2 in this study, d_i is the Euclidean distance between the member i in PF_{konwn} and its nearest member in PF_{ref}. Pareto fronts with a lower IGD value are desirable and denote a good convergence performance.

- Spacing: This indicator measures the distance variance of neighboring vectors in PF_{konwn}. A lower Spacing value indicates a good distribution of solutions along PF_{konwn}.

$$Spacing = \sqrt{\frac{1}{n_{PF}-1}\sum_{i=1}^{n_{PF}}(\overline{d}-d_i)^2} \tag{13}$$

Where n_{PF} is the number of members in the obtained Pareto front PF_{konwn}, d_i is the minimum distance between the member and its nearest member in PF_{konwn}, $\overline{d}$ is the average value of all d_i.

PF_{ref} is normally the true Pareto front, which is unknown in advance. Therefore, a reference Pareto front is adopted in the calculation of these performance metrics. In this study, PF_{ref} includes the nondominated SPs extracted from all SPs found by the proposed methods in all independent runs. Each experiment is conducted 30 independent runs for each algorithm. In summary, the evolved SPs from 6 methods × 30 runs = 180 Pareto fronts are combined into a common pool, and the nondominated sorting technique is used to extract the Pareto front from this pool.

V. EXPERIMENTAL RESULTS

A. TRAINING PERFORMANCE

The experiments are all implemented in Java 8.0 and run on a computer with Intel Core i5-4590 3.30 GHz, 8 GB RAM.

For each performance metric, a Wilcoxon signed-rank test with the significance level of 0.05 [65] is carried out on the results obtained by 30 independent runs of each method. Table 8 summarizes the statistical test results of Figure 5. In this table, A denotes GEP-N-NSGAII, B denotes GEP-T-NSGAII, C denotes CCGP-NSGAII, D denotes CCGP-SPEA2, E denotes TTGP-NSGAII, F denotes TTGP- SPEA2. For each performance metric (HVR, IGD and Spacing), the sign of '+', '−', '=' in method A vs. B indicates that according to the metric, approach A is significantly better than B, significantly worse than B, or there is no significant difference between A and B (this case is already marked in bold) based on the Wilcoxon signed rank test with the significance level of 0.05.

TABLE 8. p-values of the statistical test for each metric on training set.

Comparison	HVR	IGD	Spacing
A vs. B	<0.0001-	<0.0001-	0.011-
A vs. C	<0.0001-	<0.0001-	<0.0001-
A vs. D	<0.0001-	<0.0001-	<0.0001-
A vs. E	<0.0001-	<0.0001-	<0.0001-
A vs. F	**0.644=**	**0.658=**	**0.06=**
B vs. C	<0.0001-	<0.0001-	**0.079=**
B vs. D	0.021-	**0.329=**	**0.688=**
B vs. E	**0.629=**	**0.106=**	0.001-
B vs. F	<0.0001+	<0.0001+	**0.28=**
C vs. D	0.02+	0.013+	**0.086=**
C vs. E	<0.0001+	0.004+	**0.491=**
C vs. F	<0.0001+	<0.0001+	0.005+
D vs. E	0.015+	**0.53=**	0.005+
D vs. F	<0.0001+	<0.0001+	0.017+
E vs. F	<0.0001+	<0.0001+	<0.0001+

As shown in Figure 5, for the HVR metric, the result is that CCGP-NSGAII>CCGP-SPEA2>TTGP-NSGAII≥GEP-T-NSGAII>TTGP-SPEA2≥GEP-N-NSGAII. For the IGD metric, the result is that CCGP-NSGAII>CCGP-SPEA2≥TTGP-NSGAII≥GEP-T-NSGAII>TTGP-SPEA2 ≥ GEP-N-NSGAII. In term of Spacing, CCGP-NSGAII≥TTGP-NSGAII>CCGP-SPEA2≥GEP-T-NSGAII ≥ TTGP-SPEA2≥GEP-N-NSGAII. In addition, there is no significant difference between the five algorithms in off-line training time except for the GEP-N-NSGAII. This is because the GEP-N-NSGAII use the same NFEs as the GP-based methods, but the running time of GEP-N-NSGAII is much less than that of GP-based methods. It also shows that the GEP algorithm is more efficient than GP. However, the solution

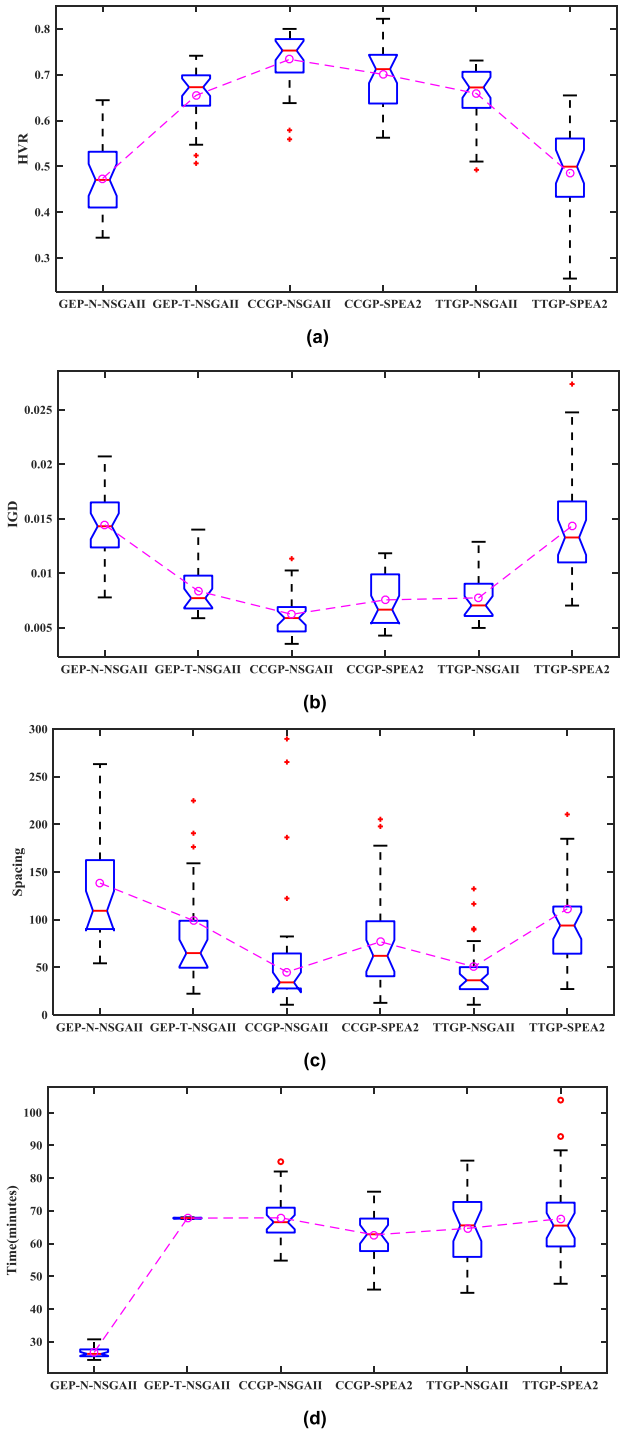


FIGURE 5. Performances of the proposed methods in training scenarios (a) HVR (b) IGD (c) Spacing (d) Running time.

quality of GEP-N-NSGAII is the worst in all the algorithms. It should be stressed that the off-line training time is not particularly important because the training process can be carried out when the workers get off work.

In addition, the solution quality of GEP-T-NSGAII is much better than that of TTGP-SPEA2 and GEP-N-NSGAII, but it is still weaker than that of TTGP-NSGAII and MO-CCGP under the same computational time. Since MO-GEP is inherently unsuited to cooperative coevolution, it was solved using a two chromosomes approach which is similar to the two subtrees model of the MO-TTGP.

From these results, we can observe that the metrics of HVR and IGD produced by MO-CCGP are significantly better than that of MO-TTGP methods and the MO-GEP methods. In terms of Spacing, the values obtained by GP-NSGAII based methods are significantly better than that of GP-SPEA2 based methods and the GEP-based methods. To conclude, the collaborative patterns pose a great impact on the performance metrics of HVR and IGD, and the multi-objective approaches significantly affect the performance of Spacing. The framework of MO-CCGP algorithm is more suitable for solving complex scheduling problems with strong nonlinear correlation.

In addition, there is a close relationship between the interpretation and the simplification of SPs with the program length of the evolved SPs. Hence, we recorded the average program length of a complete SP from the Pareto front PF_{known} in each independent run. As shown in Figure 6(a), the average program length of the MO-CCGP methods is generally higher than that of the MO-TTGP methods. This is because the cooperative co-evolution could find more problem related features than the single population evolution method. Besides, the SPs investigated in this study involve two rules: JSR and MAR. Therefore, the program length of the individual needs to be shown separately for two decision rules. As shown in Figure 6(b), the program length of JSR is smaller than that of MAR in the proposed methods. It indicates that the machine assignment problem is relatively more difficult than the job sequencing problem and MAR will use more problem-related features.

Overall, we found that the CCGP-NSGAII method achieve the best performance on the metrics of the HVR, IGD and Spacing among the six proposed methods. However, the average program length of the evolved SPs generated by the CCGP-NSGAII is higher than other methods. This is because it contains more problem-related features.

Figure 7 shows the average performance metrics of HVR, IGD and Spacing across generations of the GP-based methods from all the 30 independent runs. As shown in Figure 7(a), the HVRs grow quite fast at the early generations and the growing rate is smaller in the latter generations. It is very clear that the MO-CCGP methods achieve higher HVR metric than the MO-TTGP methods. The HVRs obtained by the NSGAII-based methods are significantly better than that of SPEA2-based methods.

The detailed results from Figure 7(b) show that the MO-CCGP methods achieve lower IGD metric much faster than the MO-TTGP methods. Apparently, the MO-CCGP methods only need half of the maximum NFEs to find the best IGD metric that obtained by the MO-TTGP methods. Figure 7(c) shows that the NSGAII-based methods achieve lower Spacing metric than the SPEA2-based methods. These detailed results confirm the above conclusions.

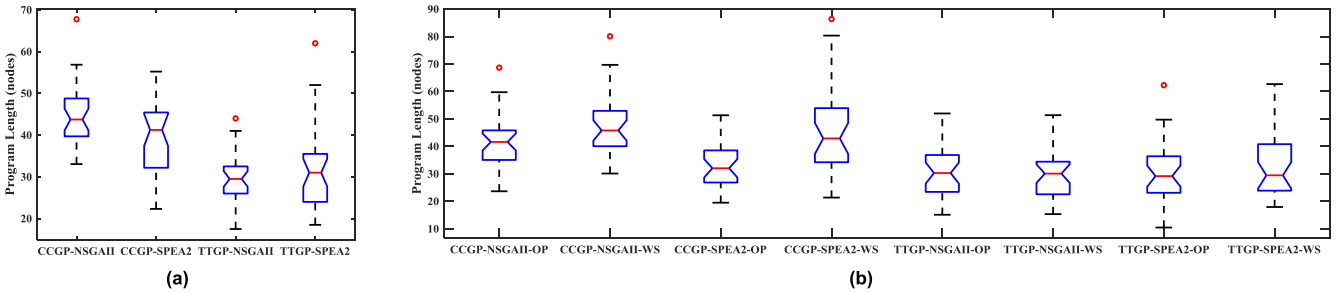


FIGURE 6. Program length of the evolved SP (a) Average program length of the evolved SPs (b) Program length of the JSR and MAR, respectively. ('op' and 'ws' represent the program length of the JSR and MAR, respectively).

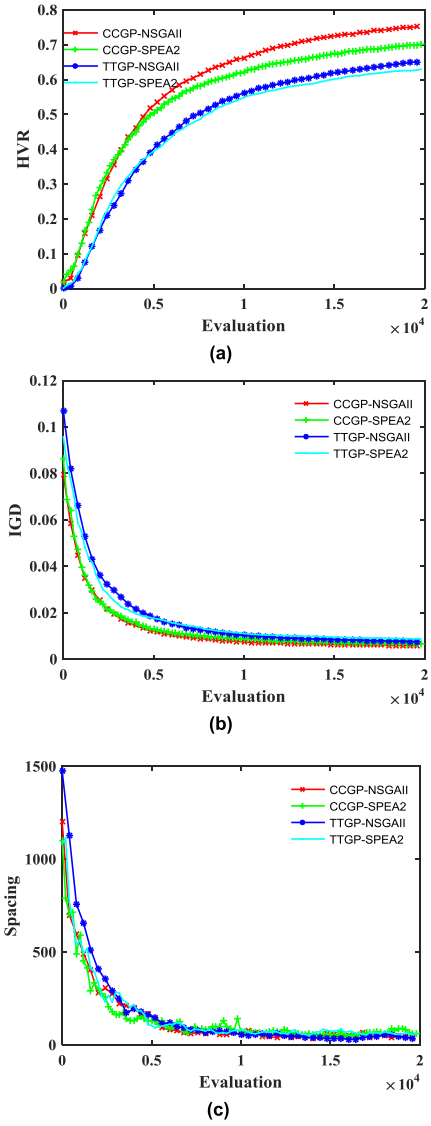


FIGURE 7. Average performance metrics of the proposed methods across generations (a) HVR (b) IGD (c) Spacing.

B. TESTING PERFORMANCE

To evaluate the generalization performance of the evolved SPs, six aggregate Pareto fronts extracted from the six proposed methods in the 30 independent runs are separately applied to the test set. Because the random seed has a significant impact on simulation results, we use the same random number seed for fair comparison of the six proposed methods in each experiment. In summary, we conducted a total of 30 experiments for each algorithm, each of which adopted the same random seed.

Table 9 summarizes the statistical test results of Figure 8. We can observe that the six proposed methods have significant differences in the three indicators. As shown in Figure 8, for the HVR and IGD metrics, the result is that CCGP-NSGAII>CCGP-SPEA2≥GEP-T-NSGAII≥TTGP-NSGAII>TTGP-SPEA2≥GEP-N-NSGAII. In term of Spacing, CCGP-NSGAII≥TTGP-NSGAII≥CCGP-SPEA2 ≥ GEP-T-NSGAII≥TTGP-SPEA2≥GEP-N-NSGAII. These results confirm the above training performance and verify the performance consistency of the proposed algorithms in different environments.

TABLE 9. p-values of the statistical test for each metric on the test set.

Comparison	HVR	IGD	Spacing
A vs. B	<0.0001-	<0.0001-	0.069-
A vs. C	<0.0001-	<0.0001-	0.004-
A vs. D	<0.0001-	<0.0001-	0.002-
A vs. E	<0.0001-	<0.0001-	<0.0001-
A vs. F	**0.23=**	**0.53=**	**0.102=**
B vs. C	0.005-	0.018-	**0.054=**
B vs. D	**0.254=**	**0.658=**	**0.136=**
B vs. E	**0.719=**	**0.318=**	0.028-
B vs. F	<0.0001+	<0.0001+	**0.586=**
C vs. D	0.005+	0.001+	**0.558=**
C vs. E	<0.0001+	<0.0001+	**0.719=**
C vs. F	<0.0001+	<0.0001+	**0.075=**
D vs. E	0.082+	**0.262=**	**0.075=**
D vs. F	<0.0001+	<0.0001+	**0.28=**
E vs. F	<0.0001+	0.002+	0.019+

In addition, the time performance of the evolved SP is similar to that of the man-made SP on each test scenario, which is within a few seconds. Therefore, the evolved SPs are very suitable for online scheduling to cope with dynamic changes.

VI. THE EVOLVED SCHEDULING POLICIES

To evaluate the generalization performance of the evolved SPs, the Pareto front of the evolved SPs will be applied to

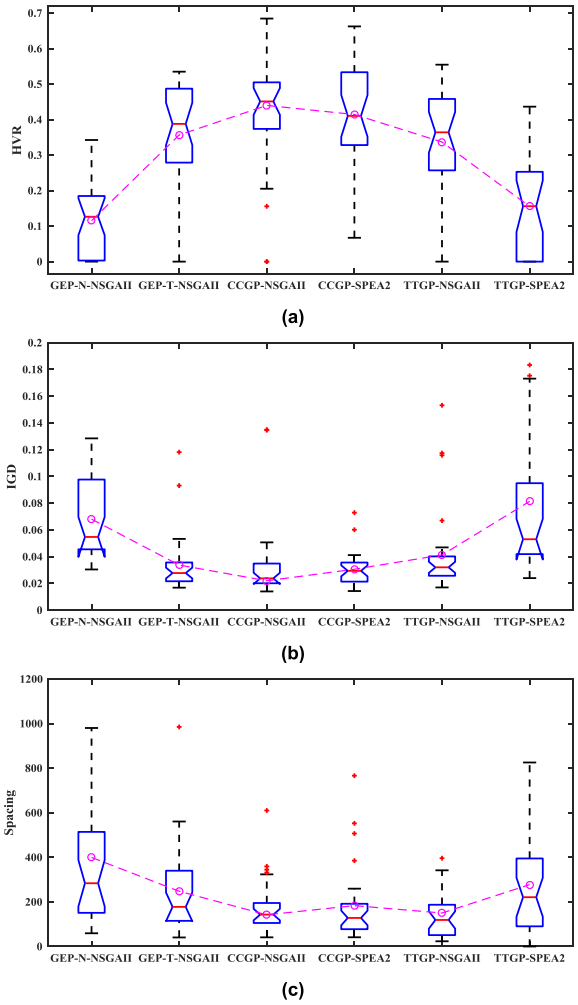


FIGURE 8. Performances of the proposed methods on the test scenarios (a) HVR (b) IGD (c) Spacing.

both the training and testing scenarios to compare with the combinations of JSR and MAR reported in the literature.

A. COMPARISON TO EXISTING SPs ON TRAINNING SET

In this study, 320 combinations of benchmark SPs are used to compare with the evolved SPs [10]–[12]. These benchmark SPs are made up of 10 well-known MARs and 30 JSRs which are shown in Table 10 and Table 11, respectively. Because

TABLE 10. Benchmark machine assignment rules.

No.	MAR	Description
1	WQT	total process time of the job in the device queue
2	WQS	queue size of the device
3	WQD	total waiting time of the job in the device queue
4	WD	total idle time of the device
5	WPT	process time of the job on the device
6	WTW	total work hours of the device
7	WRL	average queue size of the device in one minute
8	WTN	total work numbers of the device
9	WU	device utilization
10	WRD	average queueing time of the device

TABLE 11. Benchmark job sequencing rules.

No.	JSR	Description
1	ATC	apparent tardiness cost
2	CR	critical ratio
3	CR+SPT	critical ratio plus SPT
4	EDD	earliest due date
5	FASFS	first arrival at shop first served
6	FCFS	first come first served
7	LCFS	last come first served
8	LPT	longest processing time
9	LRPT	longest remaining processing time
10	MDD	modified due date
11	MOD	modified operation due date
12	ODD	operational due date
13	RND	random rule
14	SI	a truncated version of SPT
15	SL	negative minimum slack
16	SLK	minimum slack
17	SPT	shortest processing time
18	SRN	shortest remaining operation number
19	SRPT	shortest remaining processing time
20	SRPT/PT	remaining processing time per processing time
21	SRPT/SLK	remaining processing time per slack
22	PT+WINQ	processing time plus WINQ
23	PT+WINQ+Slack	processing time plus WINQ and slack
24	2PT+WINQ+NPT	double processing time plus WINQ and NPT
25	WINQ	work in the next queue
26	NPT	processing time of next operation
27	LW	largest weight
28	WMDD	weighted modified due date
29	WMOD	weighted modified operational due date
30	WSPT	weighted shortest processing time

the ATC rule contains parameter k, three parameter configurations with k equals 1.0, 2.0, 3.0 are used in this study. Therefore, there are 32 JSR$\times$10 MAR=320 combinations of benchmark SPs used for the comparison with the evolved SPs.

To evaluate the effectiveness of the proposed methods, the 320 types of benchmark SPs are applied to 8 training scenarios (see Table 2), and 100 simulation replications are performed for each scenario. Therefore, we perform 8$\times$100=800 simulation replications to test the performance of each SP on the training scenarios. Average value of the specific objective of the 800 replications generated by the benchmark SPs are recorded to compare with the results generated by the nondominated SPs (extracted from the reference Pareto front PF_{ref}) on the training set. As shown in Figure 9, the nondominated evolved SPs dominate nearly all the man-made SPs under any objective.

B. COMPARISON TO EXISTING SPs ON TEST SET

To evaluate the generalization performance of the proposed method, the evolved SPs in the aggregate Pareto front P are compared to the 320 benchmark SPs in the set B in 64 test scenarios. For each test scenario, 100 simulation replications are performed for each SP, and the average value of the specific objective of each SP is recorded

Three performance metrics as shown in Table 12 are used to determine the Pareto dominance between each pair (P_i, B_j) for all $P_i \in P$ and $B_j \in B$. The first metric is the *comparison dominance* which expresses the percentage that the evolved

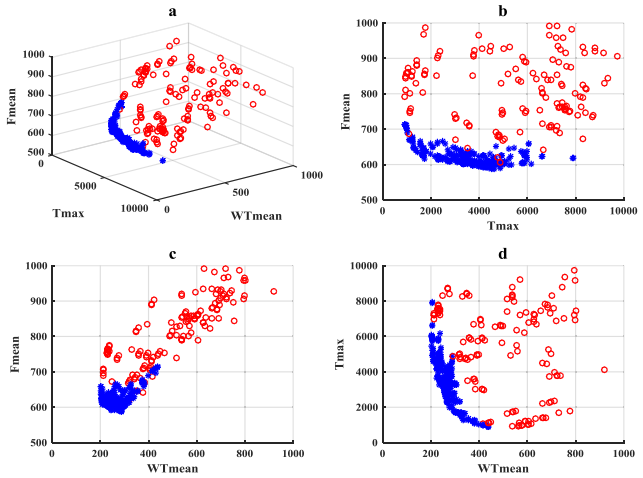


FIGURE 9. The performance of the nondominated evolved SPs and the existing SPs in the training scenarios. ('∗' and 'o' respectively represent the evolved and existing SPs).

SPs dominate the benchmark SPs in $|P| * |B|$ comparisons in each test scenario.

The second metric is the *total dominance* which means the Pareto dominance between the evolved set P and the benchmark set B. After all the comparisons were done, an evolved SP P_i can be classified into three categories [66].

- Dominating: if P_i is not dominated by any $B_j \in B$ and $\exists B_j \in B$ such that P_i dominates B_j. It is denoted by D in Table 12.
- Non-dominated: if there is no dominance between P_i and B_j for $\forall B_j \in B$. It is denoted by ND in Table 12.
- Dominated: if $\exists B_j \in B$ such that P_i is dominated by B_j. It is denoted by DE in Table 12.

The third metric is the *set coverage* [67], the set coverage C(P, B) represents the percentage of solutions in B that are dominated by at least one solution in P. It can be described as follows.

$$C(P, B) = \frac{|\{x \in B | \exists y \in P : y \text{ dominate } x\}|}{|B|} \tag{14}$$

It should be stressed that C(B,P) is not necessarily equal to 1–C(P,B). If C(P,B) is larger than C(B,P), then P is superior to B. The results are shown in Table 12. It is clear that the evolved SPs always perform better than the man-made SPs in the metrics of comparison dominance and set coverage. But the total dominance varies greatly depending on the different experimental configurations. Therefore, we use 2^6 full factorial design to investigate the effect of each factor. The main effects plot of six factors for total dominance are shown in Figure 10. Results with the main effects of six factors are shown in Table 13, and the factors that have a significant impact on the results have been marked in bold and surrounded by boxes.

From these results we can observe that four main factors (*optional device number*, *utilization*, *operation number* and *machine number*) are positively correlated with the result of total dominance. This is because the problem become more complicated as the values of these factors increase. Therefore, the evolved SPs behave more competitive than the man-made rules and can find better heuristic knowledge in the huge heuristic search space. In contrast, it is easy to see that there are two main factors (*process time* and *allowance factor*) are negatively correlated with the result of dominance. The reason is that the degree of dispersion of Normal distribution (120,20) is much lower than the Uniform distribution (1,99), and the delivery date is more relaxed in the shop floor. Hence, the search space is narrowed down and the problem is much easier to solve. Therefore, the man-made rules can also find useful knowledge to solve the problem as compared with the evolved SPs.

As shown in Table 12, we can observe that there are seven test scenarios (9,25,29,41,45,57,61) where the dominating proportion on the total dominance of the evolved SPs equals 0. These scenarios can be described as (X,X,N,X,80,2). Because of the concentrated distribution of the processing time N(120,20), low utilisation (80%) and the low optional device number (2) settings in these scenarios, the problem is more convenient to solve under these configurations. Therefore, man-made rules can also achieve similar performance as compared with the evolved SPs. From what has been mentioned above, we can safely draw a conclusion from these results: the evolved SPs can discover more useful heuristic knowledge and behave more competitive than the man-made rules in more complex scheduling environments. Thus, the generalization performance of the evolved SPs is validated in the new unobserved scenarios.

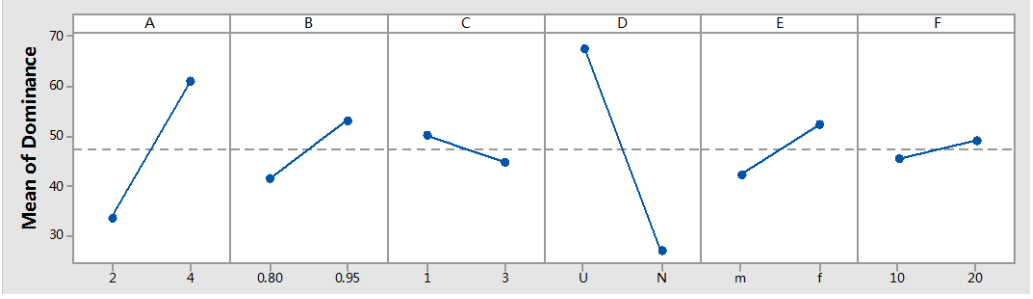


FIGURE 10. The main effects plot of six factors for total dominance.

TABLE 12. Performance of the dominance percentage of the evolved SPs.

No.	Instance	Comparison Dominance			Total Dominance			Set Coverage	
		D(%)	ND(%)	DE(%)	D(%)	ND(%)	DE(%)	C(P,B) (%)	C(B,P) (%)
1	10-m-U-1-80-2	97.23	2.77	0	38.16	61.84	0	100	0
2	10-m-U-1-80-4	99.78	0.22	0	76.32	23.68	0	100	0
3	10-m-U-1-95-2	98.48	1.52	0	51.32	48.68	0	100	0
4	10-m-U-1-95-4	99.87	0.13	0	85.53	14.47	0	100	0
5	10-m-U-3-80-2	97.14	2.86	0	44.74	55.26	0	100	0
6	10-m-U-3-80-4	98.96	1.04	0	21.05	78.95	0	100	0
7	10-m-U-3-95-2	98.91	1.09	0	47.37	52.63	0	100	0
8	10-m-U-3-95-4	99.67	0.33	0	67.11	32.89	0	100	0
9	10-m-N-1-80-2	92.37	7.63	0	**0.00**	100.00	0	100	0
10	10-m-N-1-80-4	97.65	2.35	0	50.00	50.00	0	100	0
11	10-m-N-1-95-2	94.60	5.40	0	19.74	80.26	0	100	0
12	10-m-N-1-95-4	98.08	1.92	0	57.89	42.11	0	100	0
13	10-m-N-3-80-2	93.50	6.50	0	7.89	92.11	0	100	0
14	10-m-N-3-80-4	97.03	2.97	0	53.95	46.05	0	100	0
15	10-m-N-3-95-2	94.16	5.84	0	18.42	81.58	0	100	0
16	10-m-N-3-95-4	97.36	2.64	0	44.74	55.26	0	100	0
17	10-f-U-1-80-2	97.40	2.60	0	42.11	57.89	0	100	0
18	10-f-U-1-80-4	99.97	0.03	0	96.05	3.95	0	100	0
19	10-f-U-1-95-2	98.27	1.73	0	51.32	48.68	0	100	0
20	10-f-U-1-95-4	99.95	0.05	0	90.79	9.21	0	100	0
21	10-f-U-3-80-2	97.41	2.59	0	55.26	44.74	0	100	0
22	10-f-U-3-80-4	99.96	0.04	0	94.74	5.26	0	100	0
23	10-f-U-3-95-2	99.96	0.04	0	94.74	5.26	0	100	0
24	10-f-U-3-95-4	99.91	0.09	0	75.00	25.00	0	100	0
25	10-f-N-1-80-2	88.79	11.21	0	**0.00**	100.00	0	100	0
26	10-f-N-1-80-4	94.24	5.76	0	21.05	78.95	0	100	0
27	10-f-N-1-95-2	92.90	7.10	0	13.16	86.84	0	100	0
28	10-f-N-1-95-4	95.98	4.02	0	55.26	44.74	0	100	0
29	10-f-N-3-80-2	88.82	11.18	0	**0.00**	100.00	0	99.375	0
30	10-f-N-3-80-4	94.31	5.69	0	13.16	86.84	0	100	0
31	10-f-N-3-95-2	93.48	6.52	0	18.42	81.58	0	100	0
32	10-f-N-3-95-4	96.86	3.14	0	50.00	50.00	0	100	0
33	20-m-U-1-80-2	98.56	1.44	0	57.89	42.11	0	100	0
34	20-m-U-1-80-4	99.99	0.01	0	98.68	1.32	0	100	0
35	20-m-U-1-95-2	98.64	1.36	0	52.63	47.37	0	100	0
36	20-m-U-1-95-4	99.98	0.02	0	98.68	1.32	0	100	0
37	20-m-U-3-80-2	95.92	4.08	0	51.32	48.68	0	100	0
38	20-m-U-3-80-4	97.01	2.99	0	18.42	81.58	0	100	0
39	20-m-U-3-95-2	99.53	0.47	0	77.63	22.37	0	100	0
40	20-m-U-3-95-4	98.21	1.79	0	18.42	81.58	0	100	0
41	20-m-N-1-80-2	92.00	8.00	0	**0.00**	100.00	0	100	0
42	20-m-N-1-80-4	97.58	2.42	0	36.84	63.16	0	100	0
43	20-m-N-1-95-2	92.86	7.14	0	1.32	98.68	0	100	0
44	20-m-N-1-95-4	97.46	2.54	0	48.68	51.32	0	100	0
45	20-m-N-3-80-2	90.60	9.40	0	**0.00**	100.00	0	99.0625	0
46	20-m-N-3-80-4	91.24	8.76	0	28.95	71.05	0	100	0
47	20-m-N-3-95-2	94.95	5.05	0	21.05	78.95	0	100	0
48	20-m-N-3-95-4	98.17	1.83	0	53.95	46.05	0	100	0
49	20-f-U-1-80-2	98.82	1.18	0	73.68	26.32	0	100	0
50	20-f-U-1-80-4	99.99	0.01	0	98.68	1.32	0	100	0
51	20-f-U-1-95-2	99.14	0.86	0	68.42	31.58	0	100	0
52	20-f-U-1-95-4	99.98	0.02	0	97.37	2.63	0	100	0
53	20-f-U-3-80-2	97.92	2.08	0	55.26	44.74	0	100	0
54	20-f-U-3-80-4	100.00	0.00	0	100.00	0.00	0	100	0
55	20-f-U-3-95-2	99.60	0.40	0	69.74	30.26	0	100	0
56	20-f-U-3-95-4	99.92	0.08	0	94.74	5.26	0	100	0
57	20-f-N-1-80-2	91.93	8.07	0	**0.00**	100.00	0	100	0
58	20-f-N-1-80-4	96.26	3.74	0	36.84	63.16	0	100	0
59	20-f-N-1-95-2	94.10	5.90	0	15.79	84.21	0	100	0
60	20-f-N-1-95-4	97.27	2.73	0	63.16	36.84	0	100	0
61	20-f-N-3-80-2	89.96	10.04	0	**0.00**	100.00	0	99.0625	0
62	20-f-N-3-80-4	93.43	6.57	0	52.63	47.37	0	100	0
63	20-f-N-3-95-2	95.00	5.00	0	27.63	72.37	0	100	0
64	20-f-N-3-95-4	98.13	1.87	0	50.00	50.00	0	100	0

C. INSIGHTS INTO THE EVOLVED SPs

We want to further investigate which specific objective of the evolved SPs is dominated by the man-made rules. Therefore, three evolved SPs (#1, #2, #3) as shown in Figure 11 are selected from the aggerate Pareto fronts that perform well on the specific objective of WT_{mean}, T_{max} and F_{mean} to compare with the 320 types of benchmark SPs in 64 test scenarios.

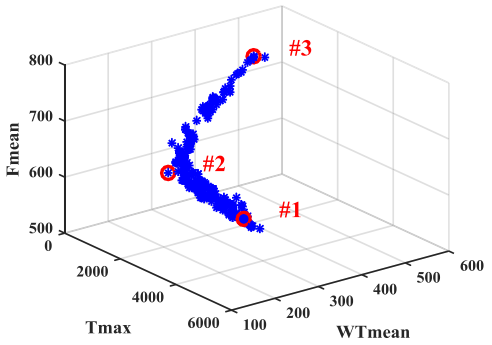


FIGURE 11. Pareto front and the selected SPs. ('∗' denotes the nondominated SPs and 'o' denotes the selected SPs).

TABLE 13. Estimated effects of the total dominance.

Index	Term	Effect	P-Value
A	Optional device number	27.3	**0**
B	Utilization	11.76	**0.003**
C	Allowance factor	-5.35	0.16
D	Process time	-40.71	**0**
E	Operation number	10.2	**0.009**
F	Machine number	3.54	0.35

TABLE 14. Examples of the evolved SPs.

Scheduling Policy #1 (WT_{mean}=188.51.291, T_{max}=5073.07, F_{mean}=623.68)	
OP	(0.14+PT+max(TOD, W)-W*W\(2PT+max(TOD, WINQ))*(0.14-2PT))\(PT+max(TOD, TIS))
WS	(0.67\WTW-2WPT)*max(WQT, WPT)
Scheduling Policy #2 (WT_{mean}=251.68, T_{max}=1338.87, F_{mean}=606.94)	
OP	max(max(OL, min(WINQ, max(max(RPT, 0.2), min(WINQ, OL)))), (max(PT, 0.4)+ W-WINQ))* min(0.15, (max(TOD, TIS)+TIQ-TOD- max(PT, 0.4))\ max(PT, 0.4))-(min(OL, TIQ), max(PT, max(PT, 0.4)))- TOD-max(min(max(PT, 0.4)), min(TOD, 0.6345761154793168))\min(WINQ, max(PT, TIQ)-TIQ-OL)
WS	WQS-WPT*(WPT+ WQT)^2
Scheduling Policy #3 (WT_{mean}=488.73, T_{max}=740.54, F_{mean}=749.72)	
OP	0.97-PT-max(PT, NPT)*2WINQ+max((OL-TOD), max((PT+TIQ)\(W* TIS),max(TIQ, OL)))+ (OL-TOD)
WS	((min(WTW, WPT)*((WPT-WTW)* max(max(WRL, WQT),(WQT+WPT))))-max(WPT, 0.63))*max(max(WRL, WTW), (WQT+WPT)) +max(WPT, (WRL-WTW))-min(WTW, 0.84)

As shown in Table 14, the evolved SPs are complicated to analysis because of the nature of GP algorithm. There are many researches on the subject of the simplification and interpretation of the evolved SPs [41], [68], [69]. However, this work does not focus on this point but on the generalization performance of the evolved SPs.

The three evolutionary SPs and the 320 types of benchmark SPs are simultaneously applied to the 64 test scenarios. Meanwhile, 100 simulation replications are performed in each test scenario. Average value of the specific objective obtained from 100 independent simulation replications is recorded as the performance metric of the test SP in each test scenario.

The relative deviation (RD) is used to compare the evolved SP and the benchmark SP under certain target in each test scenario. And the average relative deviation (ARD) is used for comparison between the evolved SP and the benchmark SP on the average objective function value of the 64 test scenarios. RD is defined as follows:

$$RD(\#j, Ins_i) = \frac{1}{|\mathrm{T}|}\sum_{k=1}^{|\mathrm{T}|}\frac{obj^{j}_{best}(bench, Ins_i) - obj(\#j, Ins_i)}{obj(\#j, Ins_i)} \tag{15}$$

In the formula, $obj(\#j, Ins_i)$ denotes the specific objective function value obtained by applying the evolved SP ($\#j$) to instance Ins_i; $obj^{j}_{best}(bench, Ins_i)$ denotes the best performance among 320 simulation results obtained by applying the 320 benchmark SPs to instance Ins_i under the same target; $|\mathrm{T}|$ is set to 100, which denotes 100 independent runs. $RD(\#j, Ins_i)$ denotes the relative deviation between the performance obtained by the corresponding evolved SP ($\#j$) with the best performance obtained by the 320 types of benchmark SPs on the same instance under the same target.

This method is constructed to verify the generalization performance of the evolved SPs. Obviously, the higher the $RD(\#j, Ins_i)$ value, the better result the evolved SP finds. Because the $obj(\#j, Ins_i)$ could be zero, the $obj^{j}_{best}(bench, Ins_i)$ is usually zero in these cases, we define the $RD(\#j, Ins_i)$ as 0. The exception happens in the test scenario 24, the $obj(\#1, Ins_i)$ is zero, but the $obj^{1}_{best}(bench, Ins_{24})$ is 0.13, in this case we set the $RD(\#1, Ins_{24})$ to 1.

As shown in Table 15, #1, #2 and #3 represent the three evolutionary SP, respectively. Ben.1 denotes the best result of the 320 benchmark SPs obtained in the corresponding scenario under the certain target. RD1 denotes the relative deviation between the average objective value of WT_{mean} obtained by the evolutionary SP (#1) with the best result obtained by the 320 benchmark SPs in each test scenario. Similarly, RD2 denotes the relative deviation between the average objective value of T_{max} obtained by the evolutionary SP (#2) with the best result obtained by the 320 benchmark SPs in each test scenario. RD3 denotes the relative deviation between the average objective value of F_{mean} obtained by the evolutionary SP (#3) with the best result obtained by the 320 benchmark SPs in each test scenario. The negative bold value with borders presents that the benchmark SP performs better than the evolved SP on the specific objective, and vice versa.

From Table 15, we can observe that the ARD of the objective function value WT_{mean}, T_{max} and F_{mean} is 3.86, 2.05 and 0.12, respectively. It demonstrates the evolved SPs generally perform better than the 320 types of man-made SPs on the specific objective for most scenarios. Taking the objective of WT_{mean} for example, there is only one evolved SP (#1) used to compare with the best SP among the 320 benchmark SPs in each scenario. However, there are only four scenarios that the evolved SP (#1) perform worse than the best SP among 320 benchmark SPs on objective of WT_{mean}. This result validates the generalization performance of the evolved SP.

The number of the test scenarios that the evolved SPs perform better than the 320 types of benchmark SPs under

TABLE 15. Comparison of the selected evolutionary scheduling policy and the best benchmark scheduling policy on the test set.

No.	instance	#1			#2			#3			Ben.1(WT_{mean})			Ben.1(T_{max})			Ben.1(F_{mean})			RD1	RD2	RD3
		WT_{mean}	T_{max}	F_{mean}	WT_{mean}	T_{max}	F_{mean}	WT_{mean}	T_{max}	F_{mean}	WT_{mean}	T_{max}	F_{mean}	WT_{mean}	T_{max}	F_{mean}	WT_{mean}	T_{max}	F_{mean}	WT_{mean}	T_{max}	F_{mean}
1	10-m-U-1-80-2	**285.80**	1670.7	411.9	341.4	**891.03**	427.8	296.1	1405.2	**414.46**	**367.57**	3944.3	483.5	635.9	**1069.27**	564.2	428.44	2844.17	**467.86**	0.29	0.20	0.13
2	10-m-U-1-80-4	**42.41**	498.6	259.4	49.2	**396.85**	263.6	43.5	483.6	**259.69**	**86.10**	1292.6	298.9	139.7	**492.52**	320.1	100.54	1109.89	**290.76**	1.03	0.24	0.12
3	10-m-U-1-95-2	**483.78**	3112.0	518.0	645.1	**1136.06**	567.4	510.4	2181.4	**525.25**	**637.42**	8582.3	670.4	1241.3	**1690.35**	839.2	780.09	5644.71	**630.06**	0.32	0.49	0.20
4	10-m-U-1-95-4	**77.91**	722.2	288.3	90.8	**510.11**	294.1	80.7	676.1	**289.34**	**148.00**	2143.1	343.2	270.2	**669.66**	388.6	174.41	1763.65	**333.23**	0.90	0.31	0.15
5	10-m-U-3-80-2	**1.01**	236.5	436.1	1.2	**150.33**	443.3	0.6	144.0	**427.41**	**8.33**	975.2	557.7	13.3	**265.30**	561.3	33.77	1973.57	**467.90**	7.25	0.76	0.09
6	10-m-U-3-80-4	**0.10**	40.0	267.8	0.1	**41.44**	274.7	0.1	43.0	**264.51**	**0.10**	38.4	323.9	0.1	**38.41**	323.9	2.10	496.16	**290.76**	0.01	**-0.07**	0.10
7	10-m-U-3-95-2	**9.78**	1377.1	554.4	10.6	**342.07**	558.4	7.8	895.6	**544.38**	**88.76**	5789.2	735.5	210.9	**948.86**	796.8	171.28	4754.08	**630.06**	8.08	1.77	0.16
8	10-m-U-3-95-4	**0.14**	46.5	303.2	0.2	**45.25**	312.5	0.1	46.1	**298.23**	**0.39**	66.7	393.3	0.4	**66.65**	393.3	8.81	1062.95	**333.23**	1.77	0.47	0.12
9	10-m-N-1-80-2	**964.00**	5174.7	1144.8	1126.3	**2096.35**	1171.6	997.8	4023.9	**1146.47**	**1037.44**	7292.5	1244.8	1277.5	**2126.40**	1242.3	1145.72	4147.35	**1182.70**	0.08	0.01	0.03
10	10-m-N-1-80-4	**478.52**	2643.5	888.9	521.3	**1481.19**	893.2	487.6	2350.1	**888.22**	**624.47**	3624.4	965.7	699.8	**1585.15**	978.5	662.02	3171.89	**961.65**	0.30	0.07	0.08
11	10-m-N-1-95-2	**1956.37**	16595.5	1733.7	2804.2	**3490.78**	1932.9	2120.0	7791.9	**1749.85**	**2497.33**	34360.9	2386.7	3708.6	**4127.79**	2341.2	3172.81	22222.16	**2111.64**	0.28	0.18	0.21
12	10-m-N-1-95-4	**1022.73**	8076.0	1189.9	1210.2	**2259.76**	1208.4	1063.6	5092.4	**1188.00**	**1625.96**	23355.6	1595.2	2458.0	**3396.19**	1771.9	1945.05	18277.62	**1546.99**	0.59	0.50	0.30
13	10-m-N-3-80-2	**3.74**	1307.6	1210.7	3.6	**403.79**	1183.3	3.0	849.1	**1184.31**	**5.31**	267.6	1279.5	5.4	**265.60**	1277.3	25.92	705.66	**1181.67**	0.42	**-0.34**	0.00
14	10-m-N-3-80-4	**0.15**	99.0	922.8	0.1	**31.70**	917.1	0.1	45.7	**907.25**	**0.77**	45.6	1015.0	0.8	**45.00**	1014.0	5.42	977.35	**961.65**	4.11	0.42	0.06
15	10-m-N-3-95-2	**202.16**	12223.0	1801.7	315.3	**1567.27**	1773.3	234.7	5744.2	**1806.42**	**649.58**	18244.7	2166.0	806.5	**2329.28**	2042.2	806.53	2329.28	**2042.20**	2.21	0.49	0.13
16	10-m-N-3-95-4	**20.82**	4133.9	1284.0	18.1	**581.18**	1240.2	18.6	2140.0	**1240.88**	**238.44**	12110.2	1602.7	639.3	**1692.63**	1861.2	449.17	15853.47	**1546.99**	10.45	1.91	0.25
17	10-f-U-1-80-2	**507.86**	2021.0	748.4	612.4	**1049.89**	776.5	536.2	1579.8	**756.92**	**653.69**	4833.6	874.2	1067.6	**1388.25**	984.1	759.06	3406.36	**842.63**	0.29	0.32	0.11
18	10-f-U-1-80-4	**58.91**	610.5	471.1	70.1	**472.13**	478.8	62.2	576.1	**472.71**	**131.84**	1700.8	537.9	225.2	**625.20**	579.4	158.40	1482.74	**526.53**	1.24	0.32	0.11
19	10-f-U-1-95-2	**867.64**	3816.0	942.0	1151.6	**1425.34**	1022.7	935.4	2466.7	**961.29**	**1147.23**	11258.2	1214.4	2201.2	**2212.46**	1498.3	1387.21	6907.36	**1130.48**	0.32	0.55	0.18
20	10-f-U-1-95-4	**121.28**	898.4	523.9	144.3	**608.15**	533.7	128.2	802.3	**527.17**	**241.23**	2676.8	619.4	458.4	**868.53**	700.0	289.59	2265.28	**603.21**	0.99	0.43	0.14
21	10-f-U-3-80-2	**0.40**	247.8	806.7	0.5	**135.32**	816.9	0.1	87.7	**783.71**	**7.66**	1625.7	976.0	13.0	**292.73**	1028.5	36.85	2300.40	**842.63**	18.14	1.16	0.08
22	10-f-U-3-80-4	**0.00**	0.0	491.6	0.0	**0.00**	502.3	0.0	0.0	**482.03**	**0.00**	0.0	586.5	0.0	**0.00**	586.5	1.08	412.86	**526.53**	0.00	0.00	0.09
23	10-f-U-3-95-2	**10.94**	1797.5	1026.8	10.9	**385.87**	1028.6	8.1	1024.5	**1000.56**	**120.08**	7334.0	1314.5	196.4	**1069.20**	1206.5	242.64	5810.64	**1130.48**	9.98	1.77	0.13
24	10-f-U-3-95-4	**0.00**	1.7	559.8	0.0	**0.20**	573.7	0.0	0.0	**544.56**	**0.13**	18.1	711.8	0.1	**18.07**	711.8	6.78	1182.58	**603.21**	1.00	89.33	0.11
25	10-f-N-1-80-2	**1733.96**	6263.3	2083.6	2034.9	**2730.70**	2124.2	1833.8	4732.1	**2095.94**	**1863.03**	9504.0	2244.6	2163.6	**2813.31**	2181.6	2070.18	5097.50	**2139.82**	0.07	0.03	0.02
26	10-f-N-1-80-4	**861.01**	3402.5	1619.3	942.1	**1826.84**	1625.7	888.9	2897.8	**1621.73**	**1107.23**	4950.1	1768.6	1218.5	**2097.63**	1753.5	1199.23	3538.77	**1744.85**	0.29	0.15	0.08
27	10-f-N-1-95-2	**3523.94**	20944.4	3189.0	4928.6	**4982.19**	3435.6	3921.3	8747.6	**3217.54**	**4432.24**	47679.5	4238.1	5825.2	**5724.95**	3837.2	5825.20	5724.95	**3837.17**	0.26	0.15	0.19
28	10-f-N-1-95-4	**1855.96**	10807.3	2183.8	2161.1	**3060.09**	2180.8	1966.4	5901.1	**2183.92**	**2897.44**	18117.1	3053.6	3858.5	**4627.06**	2945.9	3681.60	27149.71	**2871.88**	0.56	0.51	0.32
29	10-f-N-3-80-2	**5.20**	1922.1	2235.8	4.3	**447.03**	2171.7	3.8	1070.9	**2172.54**	**4.29**	242.4	2354.0	4.4	**240.39**	2353.2	24.75	776.06	**2136.26**	**-0.17**	**-0.46**	**-0.02**
30	10-f-N-3-80-4	**0.36**	208.2	1701.1	0.2	**36.56**	1686.8	0.1	65.8	**1665.59**	**0.50**	42.5	1867.1	0.5	**42.47**	1867.1	6.61	1127.61	**1744.85**	0.40	0.16	0.05
31	10-f-N-3-95-2	**339.44**	16268.9	3330.6	482.0	**2075.39**	3220.6	391.6	6543.0	**3317.11**	**1080.03**	20113.5	3957.3	1197.0	**2923.91**	3617.2	1197.03	2923.91	**3617.23**	2.18	0.41	0.09
32	10-f-N-3-95-4	**40.35**	6351.6	2394.5	28.6	**739.63**	2306.7	30.4	2718.5	**2300.52**	**395.53**	14248.3	2930.2	580.7	**2234.11**	2945.5	797.40	24740.93	**2875.84**	8.80	2.02	0.25
33	20-m-U-1-80-2	**535.90**	2202.8	785.5	630.3	**1309.67**	811.1	564.9	1736.8	**793.77**	**690.38**	5375.5	916.5	1164.1	**1527.52**	1053.0	801.04	3723.33	**887.82**	0.29	0.17	0.12
34	20-m-U-1-80-4	**50.43**	593.1	495.3	60.3	**486.85**	502.7	52.8	566.0	**496.45**	**128.02**	1681.2	566.4	234.6	**628.74**	614.0	153.53	1412.60	**554.42**	1.54	0.29	0.12
35	20-m-U-1-95-2	**913.01**	3995.8	984.0	1180.8	**1739.13**	1062.3	988.7	2680.8	**1005.06**	**1205.05**	11577.9	1270.5	2386.6	**2404.12**	1607.8	1462.22	6837.27	**1191.53**	0.32	0.38	0.19
36	20-m-U-1-95-4	**112.12**	864.5	549.1	133.6	**651.40**	559.0	118.7	792.6	**552.26**	**245.77**	2826.7	652.9	503.9	**897.76**	748.0	293.93	2240.00	**635.31**	1.19	0.38	0.15
37	20-m-U-3-80-2	**0.25**	73.2	843.2	0.3	**73.77**	845.8	0.2	58.2	**809.23**	**1.85**	313.9	1055.6	3.2	**129.47**	1120.1	26.53	2019.03	**888.30**	6.39	0.75	0.10
38	20-m-U-3-80-4	**0.06**	27.9	514.2	0.1	**30.50**	522.4	0.1	29.8	**501.53**	**0.05**	28.8	617.3	0.0	**28.78**	617.3	0.22	64.05	**554.36**	**-0.22**	**-0.06**	0.11
39	20-m-U-3-95-2	**2.84**	870.2	1075.0	2.3	**288.54**	1061.5	1.5	428.3	**1028.50**	**93.72**	6470.8	1390.1	153.4	**989.33**	1264.8	203.49	5287.90	**1191.53**	32.00	2.43	0.16
40	20-m-U-3-95-4	**0.07**	33.6	582.8	0.1	**35.90**	592.7	0.1	37.5	**561.93**	**0.07**	34.1	757.2	0.1	**34.12**	757.2	5.44	1044.10	**635.31**	0.04	**-0.05**	0.13
41	20-m-N-1-80-2	**1803.51**	6874.2	2163.3	2061.7	**3170.69**	2198.7	1898.0	5111.5	**2172.82**	**1892.94**	9741.5	2303.2	2318.6	**3093.48**	2312.2	2077.95	5273.99	**2205.92**	0.05	**-0.02**	0.02
42	20-m-N-1-80-4	**837.09**	3122.3	1661.1	901.0	**2059.06**	1669.3	856.7	2684.6	**1661.14**	**1069.46**	4499.9	1795.8	1175.6	**2189.99**	1793.7	1147.39	3688.30	**1780.93**	0.28	0.06	0.07
43	20-m-N-1-95-2	**3562.43**	18537.6	3167.7	4814.8	**5121.03**	3447.4	3931.8	9594.0	**3220.48**	**4189.55**	34162.9	4016.9	5711.6	**5656.35**	3849.7	5084.19	14118.56	**3579.39**	0.18	0.10	0.11
44	20-m-N-1-95-4	**1782.91**	8143.4	2158.1	2001.1	**3204.58**	2171.4	1859.7	5480.0	**2156.30**	**2663.30**	18297.2	2681.2	3448.7	**4276.97**	2822.7	3036.28	13805.00	**2642.85**	0.49	0.33	0.23
45	20-m-N-3-80-2	**0.33**	238.1	2264.3	0.2	**94.48**	2204.2	0.1	73.6	**2214.22**	**0.03**	8.9	2390.9	0.0	**6.93**	2388.9	15.58	1541.11	**2205.92**	**-0.90**	**-0.93**	0.00
46	20-m-N-3-80-4	**0.00**	1.9	1694.4	0.0	**3.85**	1690.2	0.0	0.2	**1673.82**	**0.00**	0.2	1838.1	0.0	**0.23**	1838.1	0.50	202.73	**1780.93**	0.00	**-0.94**	0.06
47	20-m-N-3-95-2	**140.21**	10627.3	3295.9	140.8	**1477.10**	3136.8	160.2	5860.4	**3298.51**	**617.59**	2037.3	3470.2	617.6	**2037.27**	3470.2	617.59	2037.27	**3470.23**	3.40	0.38	0.05
48	20-m-N-3-95-4	**5.03**	1689.6	2306.1	1.4	**178.66**	2224.5	1.4	460.8	**2217.55**	**104.63**	836.8	2921.0	104.6	**836.79**	2921.0	289.61	9120.40	**2642.85**	19.80	3.68	0.19
49	20-f-U-1-80-2	**1014.56**	2683.5	1497.0	1182.3	**1533.91**	1537.2	1072.4	2017.9	**1510.81**	**1301.16**	6839.8	1743.2	2094.2	**2117.86**	1951.3	1502.34	4367.61	**1684.50**	0.28	0.38	0.11
50	20-f-U-1-80-4	**70.21**	712.3	943.0	84.8	**574.67**	954.9	73.7	661.9	**943.15**	**205.98**	2163.9	1079.7	412.9	**814.74**	1170.1	250.30	1707.90	**1055.66**	1.93	0.42	0.12
51	20-f-U-1-95-2	**1730.39**	4941.5	1873.4	2187.2	**2098.71**	1994.3	1892.2	3106.8	**1913.96**	**2280.28**	14883.7	2415.2	4482.0	**3469.07**	3034.3	2739.44	8535.31	**2248.29**	0.32	0.65	0.17
52	20-f-U-1-95-4	**184.45**	1084.9	1045.4	217.6	**772.74**	1059.5	194.6	930.0	**1049.02**	**431.50**	3629.8	1241.1	883.6	**1188.81**	1397.3	520.80	2826.24	**1209.33**	1.34	0.54	0.15
53	20-f-U-3-80-2	**0.03**	19.9	1633.8	0.0	**9.54**	1616.7	0.01	7.8	**1542.12**	**1.31**	93.4	2124.4	1.3	**93.42**	2124.4	21.01	2268.55	**1684.50**	42.82	8.79	0.09
54	20-f-U-3-80-4	**0.00**	0.0	985.6	0.0	**0.00**	994.8	0.0	0.0	**952.27**	**0.00**	0.0	1196.4	0.0	**0.00**	1196.4	0.09	64.85	**1055.86**	0.00	0.00	0.11
55	20-f-U-3-95-2	**3.26**	1103.3	2097.9	1.5	**281.54**	2036.8	2.2	486.3	**1964.15**	**119.79**	8324.7	2627.9	181.6	**1061.43**	2363.4	267.72	6438.90	**2248.29**	35.74	2.77	0.14
56	20-f-U-3-95-4	**0.00**	0.0	1122.2	0.0	**0.00**	1129.5	0.0	0.0	**1067.01**	**0.00**	0.0	1426.1	0.0	**0.00**	1426.1	1.73	725.43	**1209.33**	0.00	0.00	0.13
57	20-f-N-1-80-2	**3410.95**	8206.0	4122.0	3901.6	**3965.25**	4173.3	3624.0	5824.8	**4136.17**	**3595.39**	12536.5	4383.0	4243.9	**4258.55**	4328.6	3939.52	6484.44	**4191.55**	0.05	0.07	0.01
58	20-f-N-1-80-4	**1587.73**	3936.8	3167.6	1702.3	**2487.12**	3173.8	1626.6	3204.6	**3163.73**	**2030.26**	5602.5	3414.9	2195.8	**2893.51**	3396.9	2183.71	3913.65	**3391.26**	0.28	0.16	0.07
59	20-f-N-1-95-2	**6764.59**	21975.2	6090.3	8982.3	**7335.27**	6477.7	7621.6	10713.0	**6170.01**	**8096.66**	50263.6	7764.0	10711.1	**8415.61**	7258.9	9929.63	18848.70	**6912.75**	0.20	0.15	0.12
60	20-f-N-1-95-4	**3408.20**	10530.4	4140.9	3746.1	**4181.01**	4103.8	3567.4	6424.0	**4120.02**	**5005.40**	20608.9	5373.4	6218.0	**6141.46**	5216.9	6154.85	6159.04	**5187.80**	0.47	0.47	0.26
61	20-f-N-3-80-2	**0.81**	402.7	4370.4	0.1	**36.77**	4200.1	0.5	117.9	**4212.30**	**0.00**	1.4	4599.3	0.0	**1.37**	4599.3	12.23	1601.46	**4191.55**	**-1.00**	**-0.96**	0.00
62	20-f-N-3-80-4	**0.00**	26.4	3251.8	0.0	**0.00**	3221.0	0.0	0.0	**3189.78**	**0.00**	0.0	3522.0	0.0	**0.00**	3496.3	0.49	99.79	**3391.26**	0.00	0.00	0.06
63	20-f-N-3-95-2	**247.69**	13157.6	6378.5	206.7	**1710.19**	5983.0	293.3	6583.8	**6305.85**	**1070.29**	2756.3	6632.2	1078.5	**2749.75**	6639.7	1070.29	2756.35	**6632.15**	3.32	0.61	0.05
64	20-f-N-3-95-4	**13.37**	2748.3	4500.0	1.8	**174.13**	4277.2	5.5	644.4	**4248.40**	**197.76**	1214.7	5703.4	197.8	**1214.73**	5703.4	510.91	13253.12	**5188.43**	13.79	5.98	0.22
ARD	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	/	**3.86**	**2.05**	**0.12**

each objective of WT_{mean}, T_{max} and F_{mean} is 60, 55 and 63, respectively. There are four scenarios (29, 38, 45, 61) with relatively poor performance of WT_{mean}. We can observe that the result of WT_{mean} is relatively small. The large deviation of the performance difference is caused by the relative difference between the results of these two types of SP, but in absolute terms, both of them are small. In addition, there are nine scenarios (6, 13, 29, 38, 40, 41, 45, 46, 61) that the benchmark SP perform better than the evolved SP (#2) under the objective of T_{max}. Because the T_{max} target is a maximum objective rather than an average objective, it may change drastically with the change of the random number seed in the simulation. Therefore, instead of the mean objective (WT_{mean} and F_{mean}), the extreme objective of T_{max} is hard to minimize. Besides, the evolved SP (#3) perform better than the 320 types of benchmark SPs in 63 scenarios on the performance of F_{mean}.

These scenarios that the evolved SPs perform worse than the 320 types of manual SPs have similar factor levels of experimentations, which can be expressed as (X,X,X,3,80,X). Because of the low utilisation (80%) and the loose delivery time (3) setting in these scenarios, the man-made SPs can also find useful heuristic knowledge to solve the problem as compared with the evolved SPs.

From these results, we can observe that the evolved SPs make a good trade-off when trying to simultaneously minimize the three objectives. Moreover, the results obtained by the evolved SPs are very competitive as compared with the man-made SPs reported in the literature that used to optimize the single objective. This result demonstrates the effectiveness and the scalability of the proposed method. The advantage of the evolved SPs is that they performed well in new unobserved scenarios, which make the evolved SPs more robust when they are employed in a stochastic and dynamic scheduling environment.

VII. CONCLUSIONS AND FUTER WORKS

This study proposes three types of methods to automatically design SPs including JSR and MAR for the MO-DFJSP. The main conclusions are as follows.

1) The results reveal that MO-GEP performs more efficient than MO-CCGP and MO-TTGP, the solution quality of MO-CCGP is significantly better than that of MO-TTGP and MO-GEP under the same computational costs. This is because the MO-DFJSP includes two sub-problems, cooperative coevolution is more suitable for solving such complex problem.
2) The results demonstrate that the random shuffling method performs the best among the five collaboration methods. This is because the two sub-problems have a strong interaction effect. We need to coevolve the two decision rules at low computational costs rather than to collaborate them in a greed way.
3) Both the training and testing performances show that CCGP-NSGAII is the most competitive approach among the proposed methods for evolving efficient non-dominated SPs. Statistical tests indicate that it has the best overall performance in terms of the three performance metrics (HVR, IGD, Spacing).
4) The evolved SPs which were extracted from the aggerate Pareto fronts were compared with the 320 types of benchmark SPs reported in the literature on both the training and testing scenarios. The results reveal that the evolved SPs can discover more useful heuristic knowledge and behave more competitive than the man-made SPs in more complex scheduling environments without increasing the online solution time. It also demonstrates that the evolved SPs can obtain trade-offs among different objectives and have a strong generalization performance to be reused in new unobserved scheduling scenario.

The future works of our study mainly focuses on the three aspects. Firstly, we plan to consider more dynamic characteristic of the problem, such as arrivals of urgent orders, cancellations of already handled jobs and resource failures. And then, we expect to use feature construction method and surrogate model-based evaluation strategy to improve the effectiveness and efficiency of the hyper-heuristic methods. At last, we plan to implement the proposed method which can automatically evolve SPs to replace the man-made SPs designed by experts in our MES (Manufacturing Execution System) software, and further test the generalization performance of the evolved SPs using different industrial cases.

REFERENCES

[1] K. Ding, P. Jiang, and S. Su, "RFID-enabled social manufacturing system for inter-enterprise monitoring and dispatching of integrated production and transportation tasks," *Robot. Comput.-Integr. Manuf.*, vol. 49, pp. 120–133, Feb. 2018.
[2] F. Tao, Q. Qi, A. Liu, and A. Kusiak, "Data-driven smart manufacturing," *J. Manuf. Syst.*, vol. 48, pp. 157–169, Jul. 2018.
[3] D. Mourtzis and E. Vlachou, "A cloud-based cyber-physical system for adaptive shop-floor scheduling and condition-based maintenance," *J. Manuf. Syst.*, vol. 47, pp. 179–198, Apr. 2018.
[4] A. Baykasoğlu and F. B. Ozsoydan, "Dynamic scheduling of parallel heat treatment furnaces: A case study at a manufacturing system," *J. Manuf. Syst.*, vol. 46, pp. 152–162, Jan. 2018.
[5] J. Zhang, J. Yang, and Y. Zhou, "Robust scheduling for multi-objective flexible job-shop problems with flexible workdays," *Eng. Optim.*, vol. 48, no. 11, pp. 1973–1989, Jan. 2016.
[6] T. Borreguero-Sanchidrián *et al.*, "Flexible job shop scheduling with operators in aeronautical manufacturing: A case study," *IEEE Access*, vol. 6, pp. 224–233, 2017.
[7] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *J. Scheduling*, vol. 12, no. 4, pp. 417–431, 2009.
[8] J. Branke, T. Hildebrandt, and B. Scholz-Reiter, "Hyper-heuristic evolution of dispatching rules: A comparison of rule representations," *Evol. Comput.*, vol. 23, no. 2, pp. 249–277, Jun. 2015.
[9] I. A. Chaudhry and A. A. Khan, "A research survey: Review of flexible job shop scheduling techniques," *Int. Trans. Oper. Res.*, vol. 23, no. 3, pp. 551–591, May 2016.
[10] V. Sels, N. Gheysen, and M. Vanhoucke, "A comparison of priority rules for the job shop scheduling problem under different flow time- and tardiness-related objective functions," *Int. J. Prod. Res.*, vol. 50, no. 15, pp. 4255–4270, Sep. 2012.
[11] V. Vinod and R. Sridharan, "Simulation modeling and analysis of due-date assignment methods and scheduling decision rules in a dynamic job shop production system," *Int. J. Prod. Econ.*, vol. 129, no. 1, pp. 127–146, Jan. 2011.
[12] T.-C. Chiang, "Enhancing rule-based scheduling in wafer fabrication facilities by evolutionary algorithms: Review and opportunity," *Comput. Ind. Eng.*, vol. 64, no. 1, pp. 524–535, Jan. 2013.
[13] L. Wang, J. Cai, M. Li, and Z. Liu, "Flexible job shop scheduling problem using an improved ant colony optimization," *Sci. Program.*, 2017, Art. no. 9016303, doi: 10.1155/2017/9016303.
[14] X. Li, Z. Peng, B. Du, J. Guo, W. Xu, and K. Zhuang, "Hybrid artificial bee colony algorithm with a rescheduling strategy for solving flexible job shop scheduling problems," *Comput. Ind. Eng.*, vol. 113, pp. 10–26, Nov. 2017.
[15] W. Xiong and D. Fu, "A new immune multi-agent system for the flexible job shop scheduling problem," *J. Intell. Manuf.*, vol. 29, no. 4, pp. 857–873, Apr. 2018.
[16] S. Kemmoé-Tchomté, D. Lamy, and N. Tchernev, "An effective multi-start multi-level evolutionary local search for the flexible job-shop problem," *Eng. Appl. Artif. Intell.*, vol. 62, pp. 80–95, Jun. 2017.
[17] M. Nouiri, A. Bekrar, A. Jemai, S. Niar, and A. C. Ammari, "An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem," *J. Intell. Manuf.*, vol. 29, no. 3, pp. 603–615, Mar. 2018.
[18] X. Li and L. Gao, "An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem," *Int. J. Prod. Econ.*, vol. 174, pp. 93–110, Apr. 2016.
[19] T. F. Abdelmaguid, "A neighborhood search function for flexible job shop scheduling with separable sequence-dependent setup times," *Appl. Math. Comput.*, vol. 260, pp. 188–203, Jun. 2015.
[20] M. Gen and L. Lin, "Multiobjective evolutionary algorithm for manufacturing scheduling problems: State-of-the-art survey," *J. Intell. Manuf.*, vol. 25, no. 5, pp. 849–866, 2014.
[21] B. Çaliş and S. Bulkan, "A research survey: Review of AI solution strategies of job shop scheduling problem," *J. Intell. Manuf.*, vol. 26, no. 5, pp. 961–973, Oct. 2015.

[22] P. Korytkowski, T. Wiśniewski, and S. Rymaszewski, "An evolutionary simulation-based optimization approach for dispatching scheduling," *Simul. Model. Pract. Theory*, vol. 35, no. 6, pp. 69–85, Jun. 2013.

[23] J. Heger, J. Branke, T. Hildebrandt, and B. Scholz-Reiter, "Dynamic adjustment of dispatching rule parameters in flow shops with sequence-dependent set-up times," *Int. J. Prod. Res.*, vol. 54, no. 22, pp. 6812–6824, 2016.

[24] H. Ingimundardottir and T. P. Runarsson, "Discovering dispatching rules from data using imitation learning: A case study for the job-shop problem," *J. Scheduling*, vol. 21, no. 4, pp. 413–428, Aug. 2018.

[25] A. Shahzad and N. Mebarki, "Data mining based job dispatching using hybrid simulation-optimization approach for shop scheduling problem," *Eng. Appl. Artif. Intell.*, vol. 25, no. 6, pp. 1173–1181, Sep. 2012.

[26] Z. Zhang, L. Zheng, F. Hou, and N. Li, "Semiconductor final test scheduling with Sarsa (λ, k) algorithm," *Eur. J. Oper. Res.*, vol. 215, no. 2, pp. 446–458, Dec. 2011.

[27] D. Golmohammadi, "A neural network decision-making model for job-shop scheduling," *Int. J. Prod. Res.*, vol. 51, no. 17, pp. 5142–5157, 2013.

[28] S. Abdullah and M. Abdolrazzagh-Nezhad, "Fuzzy job-shop scheduling problems: A review," *Inf. Sci.*, vol. 278, pp. 380–407, Sep. 2014.

[29] M. Đurasević and D. Jakobović, "Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment," *Genetic Program. Evolvable Mach.*, vol. 19, nos. 1–2, pp. 53–92, Jun. 2018.

[30] W. Xiong and D. Fu, "A new immune multi-agent system for the flexible job shop scheduling problem," *J. Intell. Manuf.*, vol. 29, no. 4, pp. 857–873, Apr. 2018.

[31] E. K. Burke *et al.*, "Hyper-heuristics: A survey of the state of the art," *J. Oper. Res. Soc.*, vol. 64, no. 12, pp. 1695–1724, 2013.

[32] N. R. Sabar, M. Ayob, G. Kendall, and R. Qu, "Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems," *IEEE Trans. Evol. Comput.*, vol. 19, no. 3, pp. 309–325, Jan. 2015.

[33] J. Branke, S. Nguyen, C. W. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Trans. Evol. Comput.*, vol. 20, no. 1, pp. 110–124, Feb. 2016.

[34] H. R. Topcuoglu, A. Ucar, and L. Altin, "A hyper-heuristic based framework for dynamic optimization problems," *Appl. Soft. Comput.*, vol. 19, no. 2, pp. 236–251, Jun. 2014.

[35] S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: A survey with a unified framework," *Complex Intell. Syst.*, vol. 3, no. 1, pp. 41–66, Mar. 2017.

[36] C. D. Geiger and R. Uzsoy, "Learning effective dispatching rules for batch processor scheduling," *Int. J. Prod. Res.*, vol. 46, no. 6, pp. 1431–1454, 2008.

[37] L. Nie, X. Shao, L. Gao, and W. Li, "Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems," *Int. J. Adv. Manuf. Technol.*, vol. 50, nos. 5–8, pp. 729–747, Sep. 2010.

[38] D. Jakobović and K. Marasović, "Evolving priority scheduling heuristics with genetic programming," *Appl. Soft. Comput.*, vol. 12, no. 9, pp. 2781–2789, Sep. 2012.

[39] M. Đurasević, D. Jakobović, and K. Knežević, "Adaptive scheduling on unrelated machines with genetic programming," *Appl. Soft. Comput.*, vol. 48, pp. 419–430, Nov. 2016.

[40] M. Đurasević and D. Jakobović, "Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment," *Genetic Program. Evolvable Mach.*, vol. 19, nos. 1–2, pp. 53–92, Jun. 2018.

[41] S. Nguyen, Y. Mei, B. Xue, and M. Zhang, "A hybrid genetic programming algorithm for automated design of dispatching rules," *Evol. Comput.*, to be published. doi: 10.1162/evco_a_00230.

[42] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 193–208, Apr. 2014.

[43] C. W. Pickardt, T. Hildebrandt, J. Branke, J. Heger, and B. Scholz-Reiter, "Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems," *Int. J. Prod. Econ.*, vol. 145, no. 1, pp. 67–77, Sep. 2013.

[44] H. Zhang and U. Roy, "A semantics-based dispatching rule selection approach for job shop scheduling," *J. Intell. Manuf.*, to be published. doi: 10.1007/s10845-018-1421-z.

[45] J. C. Tay and N. B. Ho, "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems," *Comput. Ind. Eng.*, vol. 54, no. 3, pp. 453–473, Apr. 2008.

[46] L. Nie, L. Gao, P. Li, and X. Li, "A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates," *J. Intell. Manuf.*, vol. 24, no. 4, pp. 763–774, Aug. 2013.

[47] L. Zhang, Q. Tang, Z. Wu, and F. Wang, "Mathematical modeling and evolutionary generation of rule sets for energy-efficient flexible job shops," *Energy*, vol. 138, pp. 210–227, Nov. 2017.

[48] D. Yska and Y. Mei, and M. Zhang, "Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling," in *Proc. EuroGP*, Parma, Italy, 2018, pp. 306–321.

[49] D. Yska, Y. Mei, and M. Zhang, "Feature construction in genetic programming hyper-heuristic for dynamic flexible job shop scheduling," in *Proc. GECCO*, Kyoto, Japan, 2018, pp. 149–150.

[50] F. Zhang, Y. Mei, and M. Zhang, "Genetic programming with multi-tree representation for dynamic flexible job shop scheduling," presented at the 31st Australas. Joint Conf. Artif. Intell., Wellington, New Zealand, Dec. 2018.

[51] F. Zhang, Y. Mei, and M. Zhang, "Surrogate-assisted genetic programming for dynamic flexible job shop scheduling," presented at the 31st Australas. Joint Conf. Artif. Intell., Wellington, New Zealand, Dec. 2018.

[52] 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, May 1976.

[53] Y. Mati and X. Xie, "The complexity of two-job shop problems with multi-purpose unrelated machines," *Eur. J. Oper. Res.*, vol. 152, no. 1, pp. 159–169, Jan. 2004.

[54] 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.

[55] M. Pinedo and M. Singer, "A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop," *Naval Res. Logistics*, vol. 46, no. 1, pp. 1–17, Feb. 1999.

[56] J. R. Koza, "Genetic programming as a means for programming computers by natural selection," *Statist. Comput.*, vol. 4, no. 2, pp. 87–112, Jun. 1994.

[57] C. Ferreira, "Gene expression programming: A new adaptive algorithm for solving problems," *Complex Syst.*, vol. 13, no. 2, pp. 87–129, Mar. 2001.

[58] 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.

[59] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multi-objective optimization," ETH, Zurich, Switzerland, Tech. Rep. TIK-Report 103, May 2001, doi: 10.3929/ethz-a-004284029.

[60] S. Luke and L. Panait, "A comparison of bloat control methods for genetic programming," *Evol. Comput.*, vol. 14, no. 3, pp. 309–344, Sep. 2006.

[61] T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Towards improved dispatching rules for complex shop floor scenarios: A genetic programming approach," in *Proc. GECCO*, Portland, OR, USA, 2010, pp. 257–264.

[62] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm test suites," in *Proc. ACM Symp. Appl. Comput.*, Houston, TX, USA, 1999, pp. 351–357.

[63] C. A. C. Coello and N. C. Cortés, "Solving multiobjective optimization problems using an artificial immune system," *Genetic Program. Evolvable Mach.*, vol. 6, no. 2, pp. 163–190, Jun. 2005.

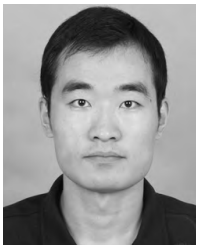
[64] J. R. Schott, "Fault tolerant design using single and multicriteria genetic algorithm optimization," M.S. thesis, Dept. Aeronaut. Astronaut., Massachusetts Inst. Technol., Cambridge, MA, USA, 1995.

[65] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.

[66] S. Nguyen, "Automatic design of dispatching rules for job shop scheduling with genetic programming," Ph.D. dissertation, Dept. Comput. Sci., Victoria Univ., Wellington, New Zealand, 2013.

[67] 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.

[68] Y. Mei, S. Nguyen, B. Xue, and M. Zhang, "An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming," *IEEE Trans. Emerg. Topics Comput. Intell.*, vol. 1, no. 5, pp. 339–353, Oct. 2017.
[69] S. Nguyen, M. Zhang, and K. C. Tan, "Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules," *IEEE Trans. Cybern.*, vol. 47, no. 9, pp. 2951–2965, Sep. 2017.
[70] R. P. Wiegand, W. C. Liles, and K. A. De Jong, "An empirical analysis of collaboration methods in cooperative coevolutionary algorithms," in *Proc. GECCO*, San Francisco, CA, USA, 2001, pp. 1235–1242.
[71] W. B. Powell, M. T. Towns, and A. Marar, "On the value of optimal myopic solutions for dynamic routing and scheduling problems in the presence of user noncompliance," *Transp. Sci.*, vol. 34, no. 1, pp. 67–85, 2000.



YONG ZHOU was born in Suzhou, Anhui, China, in 1987. He received the B.S. and M.S. degrees in manufacturing engineering from Guangxi University, Nanning, China, in 2009 and 2012, respectively. He is currently pursuing the Ph.D. degree with the Laboratory of the Intelligent Manufacturing Technology and Systems, Department of Industrial and Manufacturing Systems Engineering, School of Mechanical Engineering and Automation, Beihang University, Beijing, China.

His research interests include production scheduling, hyper-heuristic algorithm, and intelligent manufacturing system.

JIAN-JUN YANG was born in Fuzhou, Jiangxi, China, in 1960. He received the B.S. and M.S. degrees in manufacturing engineering from Beihang University, Beijing, China, in 1986 and 1989, respectively. From 1993 to 2002, he was an Associate Professor with the Institute of Manufacturing Systems. Since 2003, he has been a Professor with the School of Mechanical Engineering and Automation, Beihang University. He served as the Vice President of the College and the Director of the Department of Industrial and Manufacturing Systems Engineering, Beihang University.

He is the author of one book and more than 100 articles. His research interests include manufacturing resource planning and intelligent optimization, production scheduling and hyper-heuristic algorithm, manufacturing process information integration, and control technology.

LIAN-YU ZHENG was born in Nanchang, Jiangxi, China, in 1967. He received the B.S., M.S., and Ph.D. degrees in mechanical engineering from Beihang University, Beijing, China, in 1989, 1993, and 2001, respectively. He is currently a Professor and the Head of the Department of Industrial and Manufacturing Systems Engineering, School of Mechanical Engineering and Automation, Beihang University.

His current research interests include digital and intelligent manufacturing, reconfigurable flexible manufacturing, and manufacturing systems modeling and simulation.