IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 22, 2019, accepted February 1, 2019, date of publication February 5, 2019, date of current version February 27, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2897603

Multi-Agent Based Hyper-Heuristics for Multi-Objective Flexible Job Shop Scheduling: A Case Study in an Aero-Engine Blade Manufacturing Plant

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 supported 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 in Beihang University.

ABSTRACT In the paper, a case study focusing on multi-objective flexible job shop scheduling problem (MO-FJSP) in an aero-engine blade manufacturing plant is presented. The problem considered in this paper involves many attributes, including working calendar, due dates, and lot size. Moreover, dynamic events occur frequently in the shop-floor, making the problem more challenging and requiring real-time responses. Therefore, the priority-based methods are more suitable than the computationally intensive search-based methods for the online scheduling. However, developing an effective heuristic for online scheduling problem is a tedious work even for domain experts. Furthermore, the domain knowledge of the practical production scheduling needs to be integrated into the algorithm to guide the search direction, accelerate the convergence of the algorithm, and improve the solution quality. To this end, three multi-agent-based hyper-heuristics (MAHH) integrated with the prior knowledge of the shop floor are proposed to evolve scheduling policies (SPs) for the online scheduling problem. To evaluate the performance of evolved SPs, a 5-fold cross-validation method which is frequently used in machine learning is adopted to avoid the overfitting problem. Both the training and test results demonstrate that the bottleneck-agent-based hyper-heuristic method produces the best result among the three MAHH methods. Furthermore, both the effectiveness and the efficiency of the evolved SPs are verified by comparison with the well-known heuristics and two multi-objective particle swarm optimization (MOPSO) algorithms on the practical case. The proposed method has been embedded in the manufacturing execution system that is built on JAVA and successfully applied in several manufacturing plants.

INDEX TERMS Scheduling, flexible job shop, multi-agent, hyper-heuristics, genetic programming.

NOMENCLATURE

NSGAII	Nondominated sorting genetic algorithm II.
SPEA2	Strength Pareto evolutionary algorithm 2.
2/3/MPGP	Multi-objective cooperative coevolution genetic programming with two/three/multiple sub-populations.
2/3/MTGP	Multi-objective genetic programming with single population that an individual contains two/three/multiple sub-trees.
OMOPSO	Optimized multi-objective particle swarm optimization.
SMPSO	Speed-constrained multi-objective particle swarm optimization.
MAHH	Multi-agent based hyper-heuristics.

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Ching Ying.

I. INTRODUCTION

Driven by the modern technologies such as cyber-physical systems (CPS) [1], internet of things (IoT) [2], big data [3], deep learning [4] and cloud manufacturing [5], traditional

manufacturing is transformed into a digital ecosystem [6]. Benefiting from these developments, numerous sensors, automatic robots and embedded systems are widely used in the shop-floor [7]. Therefore, it is more convenient to collect real time production data. This situation poses a major challenge to the current scheduling system because dynamic changes in the shop-floor require real-time responses [8]. As a matter of fact, it is not necessary to get an optimal solution in real-time scheduling, but to get a satisfactory solution at reasonable computational expenses under multiple constraints. Then, the scheduling solution is revised based on several human-computer interactions according to the judgment of the dispatcher on the actual situation. Finally, a detailed schedule that meets the requirements of the scheduler is formulated and applied to the shop-floor.

Usually, there are many uncertainties happened in the shop floor such as delay arrival of a pre-arranged order, cancellations of already handled jobs and changes in lot size [9]. Due to these uncertainties, the dynamic changes require real time responses in the shop floor. But it is difficult to find out the satisfactory solution using an exact optimization method in a reasonable amount of computation time. In this case, the use of heuristics to produce a satisfactory solution (i. e. feasible solution that is good enough) for solving the production scheduling problem in an acceptable timeframe is a viable option. In a dynamic scheduling environment, empirical dispatching rules made by experts are frequently used by schedulers to make a detail schedule [10]. This is because heuristics are easy to understand and can be used by schedulers to get results very quickly. Although this method has the above advantages, the solution quality must be further improved by machine intelligence. With the change of orders, routes and other elements in the shop-floor, the previously established rules may not be able to adapt to new scheduling scenarios. Therefore, it is necessary to implement hyper-heuristics to further enhance the heuristics made by experts [11].

The industrial application addressed in this study is drawn from a real-world aero-engine blade manufacturing plant in China. The factory produces different kinds of aero-engines for jet fighter, transport aircraft and passenger plane. These aero-engines are varying in diameter from a few millimetres to a few meters because there are many different types of blades. The aero-engine blade manufacturing plant studied in this work produces over 100 different titanium alloy blade parts for different orders. The quantity of each order is usually between 100 and 500 and each order must go through a fixed process route. Each step requires a specific set-up which takes a few hours including time for fixture installation and tool testing. Similarly, each step finishes some specific operations and it usually take a few minutes as processing time, which is much less than the set-up time. Therefore, interruptions should be avoided as much as possible during job processing. There are several copies of critical machines in the shop floor to increase production capacity. Each step in the process route can be processed on one of a specified set of machines. However, some machines are still found to be the bottleneck through the rough-cut capability planning (RCCP) in the MES. Hence, bottleneck machine scheduling is one of the key issues to make a good schedule.

The aero-engine blade manufacturing plant investigated in this study comprises 5 device groups and each device group made up of 2 machines. The FCMC (a 4-axis CNC milling centre) was found as the bottleneck through the RCCP in the MES. Therefore, the prior knowledge of bottleneck resource should be integrated into the algorithm to guide the search direction. Three MAHH methods which consist of 12 algorithms are proposed to evolve scheduling policies (SPs) for the online scheduling problem. By constructing a 5-fold cross-validation, the 12 algorithms are applied to both the training and test scenarios. The multi-objective performance metrics of each algorithm are compared in both the training and test sets, respectively. Both the training and test results demonstrate that the bottleneck-agent based hyper-heuristic produces the best result among the three MAHH methods.

To verify the effectiveness and efficiency of the proposed method, the evolved SPs produced by the MAHH, the 1536 combinations of benchmark SPs and two MOPSOs are applied on the real case for comparison. Comparing with the benchmark SPs, we found that the evolved SPs dominate nearly all the well-known SPs in all aspects of objectives. In addition, the evolved SPs achieve the similar scheduling performance with the two MOPSOs, but the number of fitness evaluation (NFE) of the evolved SPs produced by the MAHH amounted to 0.34% of that of two MOPSOs. Furthermore, the proposed method has been embedded into the MES developed in our laboratory and successfully applied in several manufacturing plants.

The motivation of this study is to integrate domain knowledge into the algorithm to accelerate the search process and improve the solution quality. And the authors aim to make a step towards reducing the gap between theoretical progress and industry practice. At first, three MAHH methods are developed to integrate the prior knowledge into GP-based iterative search. And then, the evolved SPs are investigated to quickly provide an effective scheduling scheme for schedulers in industry practice. Finally, the real information in the shop floor can be synchronized with the cyber information of the scheduling system in real time to form a closed loop control. Compared to the previous studies, our contributions are as follows.

1) Three types of multi-agent based hyper-heuristic methods are proposed to achieve effective machine selection and job sequencing decision making in the MO-FJSP. The proposed system can make a good balance between computation time and solution quality in online scheduling.
2) A new disjunctive graph-based fitness evaluation is proposed to replace the simulation-based evaluation for the evolved SPs in the actual production scheduling.

3) An unsupervised learning framework is achieved to automatically evolve scheduling policies and replace the predesigned rules in the MES, which are successfully applied in several manufacturing enterprises.
4) A new type of online scheduling mode is proposed, and the steps of the new mode are as follows: Firstly, the scheduling scheme is generated rapidly based on the evolved SPs, and the scheduling scheme is transformed into the dispatching instruction through several human-computer interactions. Then, the work instruction is sent to the shop floor for execution. When the job is done, the operator will feed back to the scheduling system in real time. Finally, the scheduler can synchronize the real information in the shop floor with the cyber information of the scheduling system in real time to form a closed loop control of production scheduling.

The remainder of the paper is organized as follows. Section II provides an overview of the related works reported in the literature. Section III describes the problem investigated in this study. Detailed descriptions of the proposed methods are given in section IV. The design of experiments and the results are presented in section V. Section VI further analyzes the evolved SPs and shows the practical application in industry practice. Section VII provides our conclusions and proposals for future works.

II. LITERATURE REVIEW

Recently, many types of approximate optimization methods have been proposed to solve production scheduling problem. They can be classified into four categories, artificial intelligence, heuristics, meta-heuristics and hyper-heuristics [12].

A. ARTIFICAL INTELLIGENCE

Many machine learning approaches have been applied on this subject [13]. These methods include multi-agent [14], gaussian processes [15], imitation learning [16], data mining [17], reinforcement learning [18], artificial neural-networks [19], fuzzy logic [20], game theory [21], ensemble learning [22]. It should be noted that multi-agent approach is frequently used as one of distributed approaches in production scheduling due to its autonomy, flexibility, robustness, modularity, and heterogeneity [23]. Baykaso lu and Gorkemli [24] proposed an agent-based modelling approach to realize part family formation, virtual cell formation and scheduling phases simultaneously. Sahin *et al.* [25] proposed a multi-agent based system to simultaneous scheduling of flexible machine groups and material handling system working under a dynamic environment. Erol *et al.* [26] developed a multi-agent based approach to dynamic scheduling of machines and automated guided vehicles with a manufacturing system. An excellent algorithm not only can quickly respond to various dynamic disturbances in the real production scheduling, but also can obtain high-quality scheduling solution. However, the above two characteristics always contradict each other. If we create a multi-agent scheduling system that combines heuristic rules with iterative search, the system can obtain a good balance between computational time and solution quality [52].

B. HEURISTICS

Dispatching rule is the earliest proposed scheduling method, which assigns priorities to the operation tasks according to the predefined rules at each decision point. Compared with the traditional search-based algorithm, the dispatching rule can produce solution that is good enough in an acceptable timeframe. Hence, it is convenient to use this method for practical application. However, dispatching is a one-pass algorithm without iterations, it is well known that there is no one dispatching rule can beat any other rules in all scenarios and dispatching rules do not guarantee to find global optimal result [27]. According to the computational complexity theory [28], a large-scale complex scheduling problem belongs to NP-Complete problem. It is unable to obtain an optimal solution in a short time. Instead, it has the ability to find reasonably good solution in a short time.

It is well known that the performance of the dispatching method is significantly influenced by the characteristics of shop floor. And the traditional design of dispatching rules is based on manual trial-and-error approaches, which are time-consuming and the solution quality heavily depends on the experience of the experts. Ying *et al.* [29] developed a dynamic dispatching rule and an effective constructive heuristic for solving single-machine scheduling problems with a common due window. Gahm *et al.* [30] extended prior research on the ''decision theory'' approach to scheduling. Pergher and Almeida [31] proposed a multi-attribute, rank-dependent utility model to identify the best dispatching rule for dynamic job shop environments. Amina and El-Bouri [32] proposed a minimax linear programming (LP) model for dispatching rule selection in the presence of multiple criteria.

From these literatures we can find that many studies focus on the selection of existing scheduling rules. On the contrary, the authors focus more on the use of machine intelligence to construct scheduling policies in this study. Furthermore, simple heuristics are easy to implement in realistic scheduling but whose performance are often rather poor. Therefore, the solution quality produced by heuristics needs to be further improved. To this end, many meta-heuristics and hyper-heuristics have been investigated in this domain.

C. META-HEURISTICS

Due to the complex characteristics of the production scheduling problem, many efficient meta-heuristics have been developed to get nearly optimal solutions which satisfy the constraints and minimize or maximize the objective function [33]. We can classify these meta-heuristics into two categories, single-state methods and population methods [34].

The single-state meta-heuristics focus on increasing the exploiting ability of the algorithm. These approaches include many local search algorithms, such as, Simulated Annealing (SA), Tabu Search (TS), Greedy Randomized Adaptive

Search Procedure (GRASP) and Variable Neighborhood Search (VNS). Baykaso lu [35] developed a linguistic based simulated annealing modelling and solution approach for solving the FJSP. Baykaso lu [36] also proposed multiple objective tabu search and grammars to model and solve the MO-FJSP. Jia and Hu [37] solved the MO-FJSP using a novel path-relinking algorithm based on the state-of-the-art tabu search algorithm with back-jump tracking. Li *et al.* [38] proposed a hybrid Pareto-based tabu search algorithm (HPTSA) to solve the MO-FJSP. Baykaso lu and Karaslan [39] recently proposed an event driven dynamic job shop scheduling mechanism under machine capacity constraints to solve comprehensive dynamic job shop scheduling problem by using a GRASP-based approach.

The population methods can be further classified into two categories, evolutionary algorithm (EA) and swarm intelligence (SI). Over the past decades, evolutionary algorithm (EA) have been extensively studied to solve the production scheduling problem [40], [41]. Wang *et al.* [42] proposed an effective Pareto-based estimation of distribution algorithm (P-EDA) to solve the MO-FJSP. Reddy *et al.* [43] proposed a new evolutionary based multi-objective teacher learning-based optimization algorithm (MOTLBO) for solving real time event in MO-FJSP. Shen and Yao [44] developed a multi-objective evolutionary algorithm (MOEA) -based predictive-reactive method to capture the dynamic and multi-objective nature of FJSP. Shen et al. [45] developed a modified multi-objective evolutionary algorithm based on decomposition (m-MOEA/D) for MO-FJSP. Two new scenario-based robustness measures are defined based on statistical tools.

However, EAs differ in the implementation details and the nature of the particular scheduling problem applied. In order to have an effective implementation of EAs for production scheduling, many researchers proposed hybrid EAs to enhance the exploitation and exploration capabilities. As described by Zhou *et al.* [46], a detailed operation scheduling solution based on hybrid genetic algorithm is proposed and integrated with the manufacturing execution system (MES) for multi-objective scheduling. Yuan and Xu [47] proposed new memetic algorithms (MAs) for the MO-FJSP with the objectives to minimize the makespan, total workload, and critical workload. Gong *et al.* [48] proposed a hybrid genetic algorithm (NHGA) to solve the double flexible job-shop scheduling problem (DFJSP), in which both workers and machines are flexible. Li and Gao [49] proposed an effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem.

In recent years, many new swarm intelligence methods have been developed to solve production scheduling problems. These methods include Particle Swarm Optimization (PSO) [50], Ant Colony Optimization (ACO) [51], Artificial Immune Algorithm (AIA) [52], Artificial Bee Colony Algorithm (ABC) [53], Grey Wolf Optimization (GWO) [54], Harmony search algorithm (HS) [55], Shuffled Frog-Leaping Algorithm (SFLA) [56], Firefy Algorithm (FA) [57], Fruit fly optimization (FOA) [58]. Nouiri *et al.* [59] proposed an effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem. Zhang *et al.* [60] combined PSO and Tabu search (TS) technique to deal with the MO-FJSP, where TS was embedded into PSO as a local search. Singh *et al.* [61] developed a particle swarm optimization algorithm embedded with maximum deviation theory for solving MO-FJSP. Zhang and Wong [62] proposed a hybrid (multi-agent system) MAS/ACO approach for flexible job-shop scheduling/rescheduling problems under dynamic environment. Paprocka and Skolud [63] proposed a hybrid multi-objective immune algorithm for predictive and reactive scheduling. Lu *et al.* [64] proposed a new multi-objective discrete virus optimization algorithm (MODVOA) to solve the MO-FJSP with controllable processing times (MOFJSP-CPT). Li *et al.* [65] proposed a hybrid artificial bee colony algorithm with a rescheduling strategy for solving FJSP. Gao *et al.* [66] improved an artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time. Lu *et al.* [67] developed a hybrid multi-objective grey wolf optimizer (HMOGWO) for the multi-objective dynamic welding scheduling problem to narrow the gap between theoretical research and applicable practice. Gao *et al.* [68] proposed a Pareto-based grouping discrete harmony search algorithm (PGDHS) to solve the MO-FJSP. Lei *et al.* [69] Developed a shuffled frog-leaping algorithm (SFLA) based on a three-string coding approach for flexible job shop scheduling with the consideration of energy consumption. Karthikeyan *et al.* [70] proposed a hybrid discrete firefly algorithm to solve the MO-FJSP with limited resource constraints. Liu *et al.* [71] proposed a hybrid discrete firefly algorithm (HDFA) to solve the MO-FJSP.

From the literatures summarized above, we can safely draw the following conclusions. The advantage of heuristic method is that it has the ability to find reasonably good solutions in a short time, but the solution quality needs to be further improved. The meta-heuristic algorithm usually needs more computation time than heuristic method in online scheduling because of the iterative search process, but the solution quality is reasonably better that that of the heuristic method. Therefore, we combine the advantages of meta-heuristics and heuristics in this study to solve the MO-FJSP. In addition, the multi-agent technology is also introduced to embed domain knowledge into the offline learning process to improve the online performance of the evolved SPs.

D. HYPER-HEURISTICS

In recent years, hyper-heuristics have been proposed to combine the advantages of heuristics and meta-heuristics [72]. It operates a high-level heuristic that manages a set of low-level heuristics. The high-level heuristic often has no knowledge of the problem domain, but the low-level heuristic has problem-specific information. A hyper-heuristic algorithm can implement in new problem domains and only have to replace the set of low-level heuristics and the evaluation function. It is cheaper to implement and easier to use than

problem-specific methods. Moreover, it searches for a good problem-solving method rather than for a good solution [73]. The common goals of this method are (1) to improve the applicability of heuristic algorithm; (2) to get still acceptable solution when compared to tailor-made meta-heuristic approaches with simple algorithm design and little computation time cost; (3) to automatically design evolved heuristics for problem solving. In order to achieve these goals, the hyper-heuristics improve the abstraction level of the higher-level heuristics, so as to achieve better generalization ability. Nguyen *et al.* [74] reviewed the application of genetic programming (GP) in production scheduling and gave a unified algorithm design framework.

The hyper-heuristics can be classified into two categories according to the nature of the search space [75]. The first one is the heuristic selection, in which a meta-heuristic is used to select existing heuristics. The other one is the heuristic generation, in which new heuristics are generated from the components of existing ones. These two main types can be further categorised according to whether they are based on constructive or perturbative search. An additional classification of hyper-heuristics considers the source providing feedback during the learning process, which can be either one instance (on-line learning) or many instances of the problem studied (off-line learning).

As for heuristic selection, chromosome length is an important criterion to distinguish the literatures. Korytkowski *et al.* [76] developed evolutionary simulation-based heuristics to construct near-optimal solutions for dispatching rule allocation. Each gene is represented by an integer corresponding to the priority rule for each workstation in the system. Chromosome length is equal to the number of workstations (m) in the system. Huang and Süer [77] proposed a dispatching rule based genetic algorithm with fuzzy satisfaction levels (FRGA) to solve the multi-objective manufacturing scheduling problem. The dispatching rules for each machine at different time periods are encoded in the chromosome. The time interval for each machine is set to 10 min. Therefore, the chromosome length is equal to the number of time periods for each workstation (t)$\times$ the number of workstations (m) in the system. Zhang *et al.* [78] proposed a hybrid probabilistic model-building genetic algorithm (PMBGA) for solving the job shop scheduling problem under the total weighted tardiness criterion. PMBGA is used to search for good combinations of these rules which are then applied in a sequential manner in the simulation based decoding procedure. In PMBGA, each gene is expressed by the serial number of the selected dispatching rule. And the encoding length for each solution is the number of jobs (n)$\times$ the number of workstations (m) in the system. Vázquez-Rodríguez and Petrovic [79] proposed a new hybrid dispatching rule based genetic algorithms (DRGA) which searches simultaneously for the best sequence of dispatching rules and the number of operations to be handled by each dispatching rule. The job sequencing decision points for all machines are divided into multiple decision blocks in time order, and a dispatching rule is selected for each decision block. Six solution representations which encoded in different chromosome sizes are defined and compared in experiments, results show that the hybrid representation maintains a relatively stable good performance regardless of the size value. Li et al. [80] studied a complex scheduling problem where various machine types are considered in a multi-stage hybrid flow shop and an ant colony optimization (ACO)-based hyper-heuristic was proposed for the selection of heuristic rules.

As mentioned above, these researches which used evolutionary algorithms to select a heuristic rule for each machine are differs in chromosome length. However, the selection rules are specified in advance according to human experience, and the optimization ability cannot meet the requirements of complex scheduling problems. Besides, the research on finding the best sequence of dispatching rules for different decision points of the same machine limits the generalization performance of the results. Therefore, we focus more on the heuristic generation in this study.

As for heuristic generation, there are many studies using GP based hyper-heuristic to evolve dispatching rules for different production environments covering single machine scheduling [81], unrelated parallel machine scheduling [82], job shop scheduling [83] and flexible job shop scheduling [84], [85]. However, most existing works have focused on evolving dispatching rules for scheduling jobs on all machines in a shop floor, rather than evolving specific dispatching rules for some specific machines. Besides, experiments in the related literatures usually assume that workshop is balanced and all operations have the same average processing time. Due to this symmetry, all machines have the same expected utilisation and there is no significant difference among them. In fact, some machines are identified as the bottlenecks through RCCP before scheduling in the multi-variety and small batch manufacturing plants. It is hence necessary to evolve machine-specific heuristics for this type of scheduling problem.

In recent years, some efforts have been devoted to evolve machine-specific dispatching rules in production scheduling. Miyashita [86] proposed three approaches (homogeneous, distinct and mixed agent model) to synthesize the dispatching rule for job shop scheduling problem. However, the problem studied in the paper only used the cost as the objective. The actual workshop scheduling problem needs not only to optimize one goal, but also to optimize multiple conflicting goals. Besides, only three benchmark dispatching rules are not enough to verify the superiority of the evolved rules, and more rules are needed to prove the effectiveness of the method. Geiger *et al.* [87] proposed two methods to evolve sequencing rules for a balanced two-machine flow shop to facilitate learning of a decentralized control scheme in multiple-machine environment. The first approach is to learn a centralized sequencing rule for both machines. The second approach is to allow each machine to learn its individual sequencing rule. The results show that the best-learned rules are quite

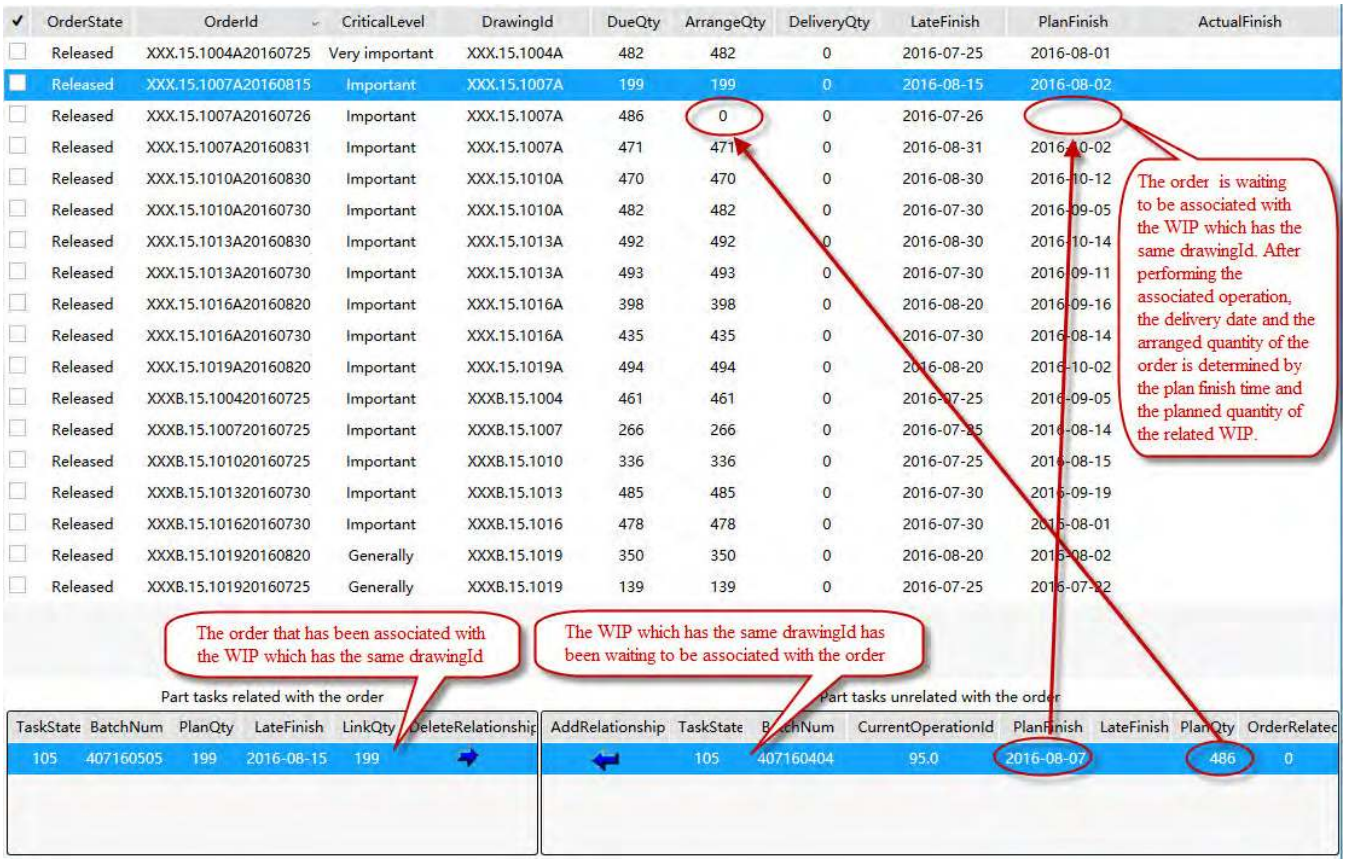


FIGURE 1. The user interface that associating orders with WIPs in the MES.

competitive with Johnson's algorithm which is used as the benchmark rule for this problem. However, the machines have the same expected utilisation in this balanced shop and the method did not consider the bottleneck machine in the system. Besides, a two-machine flow shop is much simpler than a flexible job shop. Pickardt *et al.* [88] proposed a two-stage approach to evolve work-centre-specific rules for semi-conductor manufacturing. GP was used to generate composite dispatching rules in the first stage, further an evolutionary algorithm (EA) was used to select the most suitable rule in the set of candidate rules which made up of the evolved rules and the benchmark rules for each device in the shop. The results show that the two-stage hyper-heuristics outperform the other two single-stage hyper-heuristics. Hunt *et al.* [89] investigated a genetic programming based hyper-heuristic (GPHH) approach to evolving machine-specific dispatching rules for a two-machine job shop in both static and dynamic environments. Results show that the relative performances of these methods are dependent on the test instances, which indicates that the proposed method is not effective for the scheduling problem. It is because that the proposed method did not reflect the characteristic of problem in the algorithm structure.

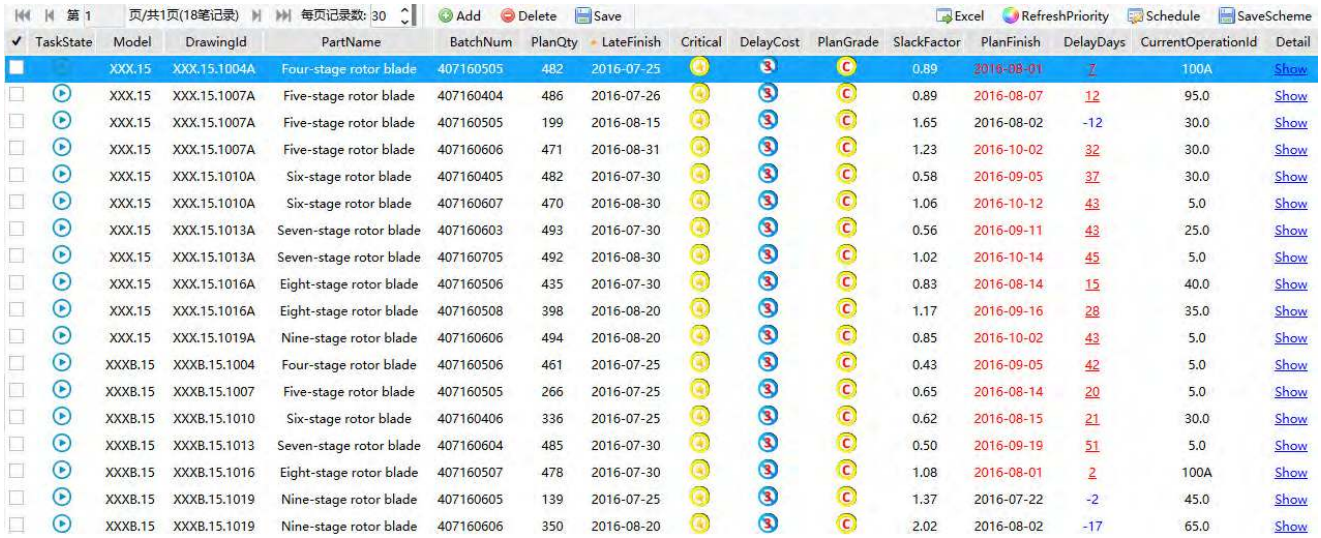
From what has been mentioned above, we can find that using MAHH to evolve machine-specific heuristics for MO-FJSP has not been analysed in previous studies. In addition, the scheduling problems in the related literatures on hyper-heuristics were usually generated by random numbers and the solution was obtained by using discrete event simulation. But the scheduling problem investigated in this study was drawn from practice and the solution was obtained by the disjunctive graph model-based scheduling method in the MES. To the best of our knowledge, our study serves as the first attempt to use the evolved heuristics produced by the MAHH to tackle the actual scheduling problem.

III. PROBLEM DESCRIPTION

A. PRE-TREATMENT OF PRODUCTION INFORMATION

In this research, the problem of detailed operation scheduling applied in an aero-engine blade manufacturing plant has been studied. And the important production information in the MES should be pre-treated.

1) Orders: As shown in Figure 1, each order in the MES is assigned to a batch of part task which is also known as work-in-progress (WIP). For example, the Order 'XXX.15.1007A20160726' is assigned to the WIP which has the same drawing Id and the batch number is '47160404'. The delivery date of the order is determined by the plan finish time '2016-08-07' of the related WIP. Similarly, the arranged quantity of the



第 1 页/共1页(18笔记录) 每页记录数: 30 Add Delete Save Excel RefreshPriority Schedule SaveScheme

TaskState	Model	DrawingId	PartName	BatchNum	PlanQty	LateFinish	Critical	DelayCost	PlanGrade	SlackFactor	PlanFinish	DelayDays	CurrentOperationId	Detail
	XXX.15	XXX.15.1004A	Four-stage rotor blade	407160505	482	2016-07-25	4	3	C	0.89	2016-08-01	7	100A	Show
	XXX.15	XXX.15.1007A	Five-stage rotor blade	407160404	486	2016-07-26	4	3	C	0.89	2016-08-07	12	95.0	Show
	XXX.15	XXX.15.1007A	Five-stage rotor blade	407160505	199	2016-08-15	4	3	C	1.65	2016-08-02	-12	30.0	Show
	XXX.15	XXX.15.1007A	Five-stage rotor blade	407160606	471	2016-08-31	4	3	C	1.23	2016-10-02	32	30.0	Show
	XXX.15	XXX.15.1010A	Six-stage rotor blade	407160405	482	2016-07-30	4	3	C	0.58	2016-09-05	37	30.0	Show
	XXX.15	XXX.15.1010A	Six-stage rotor blade	407160607	470	2016-08-30	4	3	C	1.06	2016-10-12	43	5.0	Show
	XXX.15	XXX.15.1013A	Seven-stage rotor blade	407160603	493	2016-07-30	4	3	C	0.56	2016-09-11	43	25.0	Show
	XXX.15	XXX.15.1013A	Seven-stage rotor blade	407160705	492	2016-08-30	4	3	C	1.02	2016-10-14	45	5.0	Show
	XXX.15	XXX.15.1016A	Eight-stage rotor blade	407160506	435	2016-07-30	4	3	C	0.83	2016-08-14	15	40.0	Show
	XXX.15	XXX.15.1016A	Eight-stage rotor blade	407160508	398	2016-08-20	4	3	C	1.17	2016-09-16	28	35.0	Show
	XXX.15	XXX.15.1019A	Nine-stage rotor blade	407160606	494	2016-08-20	4	3	C	0.85	2016-10-02	43	5.0	Show
	XXXB.15	XXXB.15.1004	Four-stage rotor blade	407160506	461	2016-07-25	4	3	C	0.43	2016-09-05	42	5.0	Show
	XXXB.15	XXXB.15.1007	Five-stage rotor blade	407160505	266	2016-07-25	4	3	C	0.65	2016-08-14	20	5.0	Show
	XXXB.15	XXXB.15.1010	Six-stage rotor blade	407160406	336	2016-07-25	4	3	C	0.62	2016-08-15	21	30.0	Show
	XXXB.15	XXXB.15.1013	Seven-stage rotor blade	407160604	485	2016-07-30	4	3	C	0.50	2016-09-19	51	5.0	Show
	XXXB.15	XXXB.15.1016	Eight-stage rotor blade	407160507	478	2016-07-30	4	3	C	1.08	2016-08-01	2	100A	Show
	XXXB.15	XXXB.15.1019	Nine-stage rotor blade	407160605	139	2016-07-25	4	3	C	1.37	2016-07-22	-2	45.0	Show
	XXXB.15	XXXB.15.1019	Nine-stage rotor blade	407160606	350	2016-08-20	4	3	C	2.02	2016-08-02	-17	65.0	Show

FIGURE 2. The user interface for part task scheduling in the MES.

order is also defined by the related WIP. It should be stressed that the relationship between the orders and the WIPs is not a simple one-to-one correspondence. Due to the complexity of the real production, an order can have multiple related WIPs and each WIP can associated with multiple orders. Moreover, these orders are prioritized because they are provided for different customers. Currently, the priority of the orders in the MES are divided into three levels, including ''very important, important, and generally''. These preference attributes will directly affect the priorities of the part tasks in online scheduling.

2) Part Tasks: Since some of part tasks are produced for predicted orders, there are no delivery date or very loose due date for them. Therefore, it is necessary to assign the delivery date of the orders to the related WIPs. Besides, the WIPs are usually processed in different production stages. In most of the related literature, all part tasks are sequenced from the first step at the beginning of scheduling. However, it is necessary to consider the current progress of each part task in the real production scheduling. As shown in Figure 2, there are 18 WIPs in the MES, and each WIP is associated with an order (shown in Figure 1) to satisfy the delivery date. Each part task contains multiple operation tasks that can be opened from a hyperlink ('Show') for viewing and adding constraints (shown in Figure 3).
3) Operation tasks: Each part task contains some operation tasks according to its process route. Since the manufacturing plant produces hundreds of parts, generating a large-scaled setup time matrix is neither practical nor reliable. As a result, the factory sets different set-up times for each operation task. However, in the practical scheduling, the part tasks with similar preparation content are arranged together manually to reduce the switching cost of production. Since there are several machines which have the same function in the shop floor, an operation can be processed on any one of these machines. This set of machines is denoted as the 'workstation' in this study. Figure 3 illustrates the process route, constraint and scheduling information of an exemplary part task whose drawing Id is XXX.15.1010A and the batch number is 407160607. The number of operation tasks is 35 for this part task, and some of them are processed in a cooperative workshop, which are described as the co-operation tasks. It should be noted that each operation task can own its preferred device that can be opened from a hyperlink ('Preference') for viewing and adding constraints, which will be described in the 'Eligibility constraints'.
4) Co-operation tasks: The areo-engine blade is not only processed at the machining workshop, but also processed at the heat treatment workshop, spraying workshop, inspection room and other co-operation units to complete the whole production. In the related literature, this type of scheduling is known as the distributed scheduling problem. The scheduler needs to distinguish the requestor and the responser between two workshops. The collaborative scheduling process is described as follows: the requestor sends the co-operation tasks to the responser which receive the tasks and incorporate them into the current scheduling scheme, and then the plan start time and the plan finish time are feedback to the requester after the detailed operation scheduling by the responser. If the requester is not satisfied with the result, the collaboration process will be relaunched to request assistance until both are satisfied. Besides, the production time and the set-up time of each operation in the cooperation workshop are set based on the experience of the scheduler, and the transportation time is included in the set-up time. Moreover, the resource

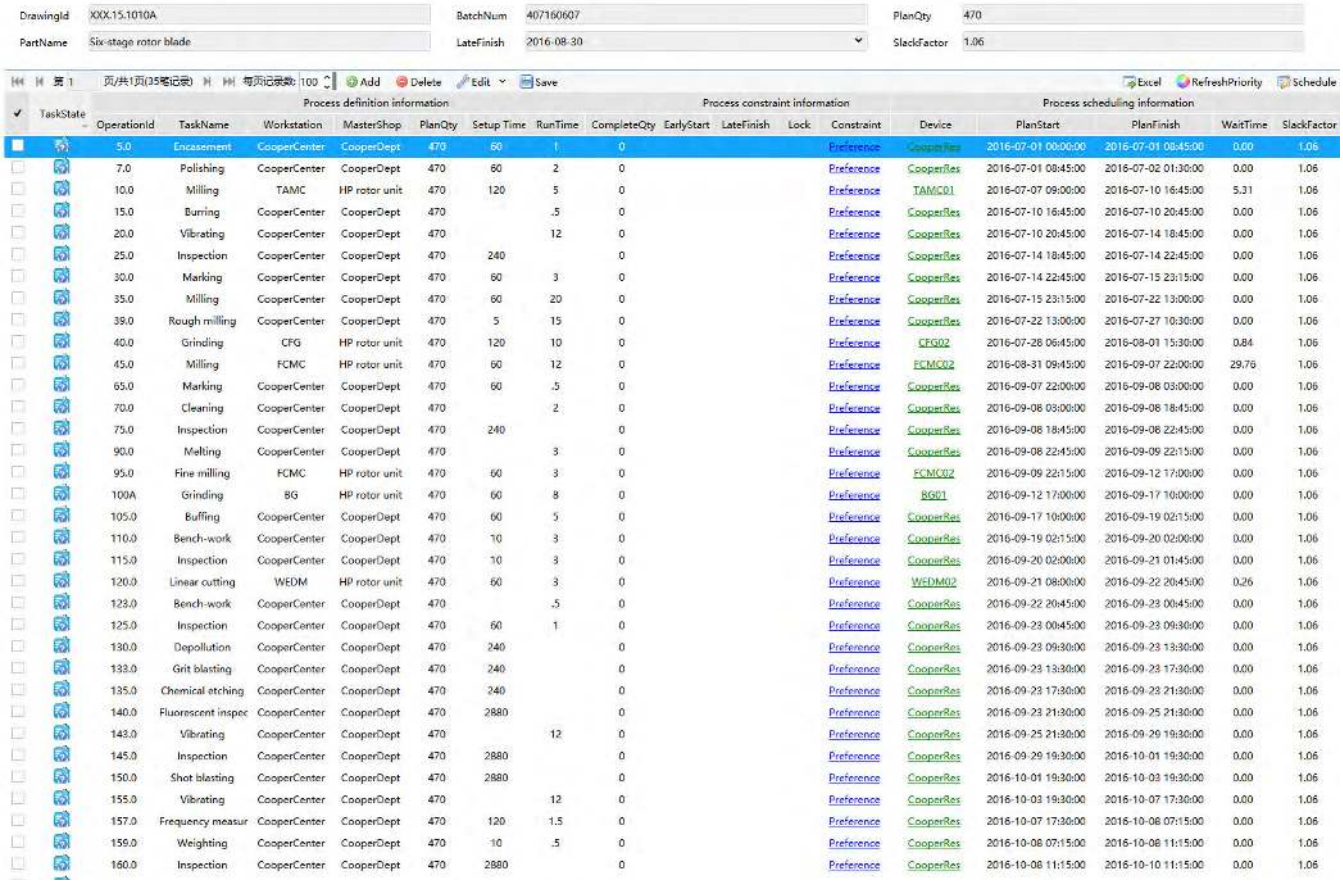


FIGURE 3. The user interface for operation task scheduling in the MES.

TABLE 1. The resource information and working calendar in the MES.

Resource	Description	Number of machines	Working shift
TAMC	3-axis machining center	2	From Monday to Friday
CFG	creep feed grinder	2	08:00-12:00; 14:00-23:59.
BG	belt grinding machine	2	
WEDM	wire cut electrical discharge machine	2	From Saturday to Sunday
FCMC	4-axis CNC milling center	2	08:00-12:00; 14:00-18:00.

in the cooperative unit is set to unlimited capacity for production scheduling.

5) Working calendar: In the MES, users could define specific calendar for each device, which means that each device could have a unique working calendar. As shown in Table 1, the normal work shifts in the factory are 08:00–12:00 and 14:00–23:59 at weekdays, 08:00–12:00 and 14:00–18:00 at weekends. Moreover, an exception period will be set for the device due to failure and maintenance, and the MES will avoid scheduling tasks for the device during this period.
6) Eligibility constraints: Due to different equipment specifications and manufacturers, the equipment functions in the same workstation are not exactly the same. Thus, some jobs can be only processed on some of the predefined machines. Currently, the priorities of the device for each operation are divided into five levels in MES, including ''must be processed on this machine, preferred to be processed on this machine, generally, as far as possible not be processed on this machine, must not be processed on this machine''. These preference attributes which are specified by the scheduler will directly affect the priorities of machines in online scheduling.
7) Bottleneck identification: The scheduler has experience in identifying which machine is the bottleneck because of the long-term work. And this is also verified by the scheduling results of the RCCP in the MES. The RCCP is used to check whether the critical resource is available to support various production orders. The load ratio of each workstation is calculated as follows:

$$L_s = \frac{\sum_{p=1}^{P} set_{i,j} + q_i \times run_{i,j}}{WC \cdot M_s} \quad (1)$$

Where L_s is the load of workstation s, WC is the total working capability of each machine at workstation s, M_s is the number

Resource Name	Capacity/hour	Setup Time/hour	Run Time/hour	Load/hour	Load Ration
FCMC	1501.00	30.00	1447.05	1477.05	98.40%
BG	1501.00	18.00	988.93	1006.93	67.08%
CFG	1501.00	26.00	913.67	939.67	62.60%
WEDM	1501.00	18.00	370.85	388.85	25.91%
TAMC	1501.00	12.00	222.33	234.33	15.61%

FIGURE 4. The RCCP results in the MES.

of machines at workstation s. $set_{i,j}$ is the set-up time for operation $O_{i,j}$ and $run_{i,j}$ denotes an item processing time for operation $O_{i,j}$, q_i is the quantity of item for operation $O_{i,j}$. P denotes the total number of operation tasks assigned to this workstation. If L_s is overly high, the production orders could not be delivered on time. Therefore, it is important to identify bottleneck resources in advance for scheduling.

As shown in Figure 4, the workstation FCMC is found as the bottleneck (98.40%) according to the RCCP in the MES. The time range is set between 2016/07/01 and 2016/09/01 in the RCCP.

B. PROBLEM STATEMET

Based on the above collection and pre-process of production information from the MES, the MO-FJSP with predefined bottleneck stage is formulated as follows.

1) There is a set of independent part tasks $= \{J_i\}_{1<i<n}$, indexed i be a set of n part tasks to be scheduled.
2) There is a set of device groups $M = \{M_k\}_{1<k<m}$, indexed k be a set of m device groups in the shop floor.
3) Each machine can perform only one operation at a time. Each machine has a certain work shift.
4) Each part task J_i contains a predefined sequence of operations. Let $O_{i,j}$ be operation j of J_i. Each part task J_i consists of q_i identical items.
5) Each operation $O_{i,j}$ is processed without interruption on one machine of the workstation M_k. The processing time of $O_{i,j}$ is defined as

$$p_{i,j} = set_{i,j} + q_i \times run_{i,j} \tag{2}$$

where $set_{i,j}$ is the set-up time for operation $O_{i,j}$ and $run_{i,j}$ denotes an item processing time for operation $O_{i,j}$.
6) Let w_i, d_i, c_i be the weight, the due date and the completion time of job J_i, respectively. The tardiness of job J_i can be calculated by the following formula:

$$T_i = \max\{0, c_i - d_i\} \tag{3}$$

7) Different types of scheduling objectives must be simultaneously considered by a scheduler when creating a detailed schedule. In this study, we aim to minimize three objectives. They are mean weighted tardiness of orders (T_{mean}), max tardiness of orders (T_{max}), and mean wait time of operation tasks (W_{mean}). They can be defined respectively as follows:

$$T_{mean} = \frac{1}{n}\sum_{i=1}^{n} w_i \times T_i \tag{4}$$

$$T_{max} = \max\{T_i | i = 1, \ldots, n\} \tag{5}$$

$$W_{mean} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{no_i}(ps_{i,j} - es_{i,j}) \tag{6}$$

where $ps_{i,j}$ is the plan start time of operation $O_{i,j}$, $es_{i,j}$ is the early start time of operation $O_{i,j}$, no_i is the number of operations in job J_i.

C. DISJUNCTIVE GRAPH MODEL

In the related literature on hyper-heuristics, discrete event simulation (DES) is the main evaluation technique to estimate the performance of scheduling heuristics [84], [88]. However, the DES model is not easy to be applied in practice, because it is difficult to manipulate constraints on the elements in the simulation model. In our previous works [90], the disjunctive graph has been introduced as a useful representation of operation precedence in the MES. Because the disjunctive graph-based scheduling model is convenient to manipulate various constraints through human-computer interactions in the MES, such as batch splitting, changing the working calendar of the specified resource and other manual adjustment operations [91]. In this study, we also use this model to evaluate the fitness of the individuals (evolved SPs). In addition, all of the elements in the MES, such as operations, jobs, machines, calendars, departments and employees, are modelled by the object-oriented technology and programmed in Java.

Figure 5 illustrates a feasible schedule and its corresponding scheduling process. A node denotes an operation in the disjunctive graph model, and the node with the same color indicates that the operation is processed on the same machine. For example, $A1$ represents the first operation of job A,$A1$and$C1$ are arranged on the same workstation $W2$. The conjunctive (solid) arcs represent the technological precedence constraints among operations of the same job. The disjunctive (dotted) arcs represent the operations that can be processed on the same machine. Each node has at most two immediate predecessors and successors (shown as nodes connected by conjunctive and disjunctive arcs in the directed graph). A schedule is feasible only when the corresponding graph does not contain a cycle [92].

The resource capability model is used to allocate operation task to time segments of machines. Because time and

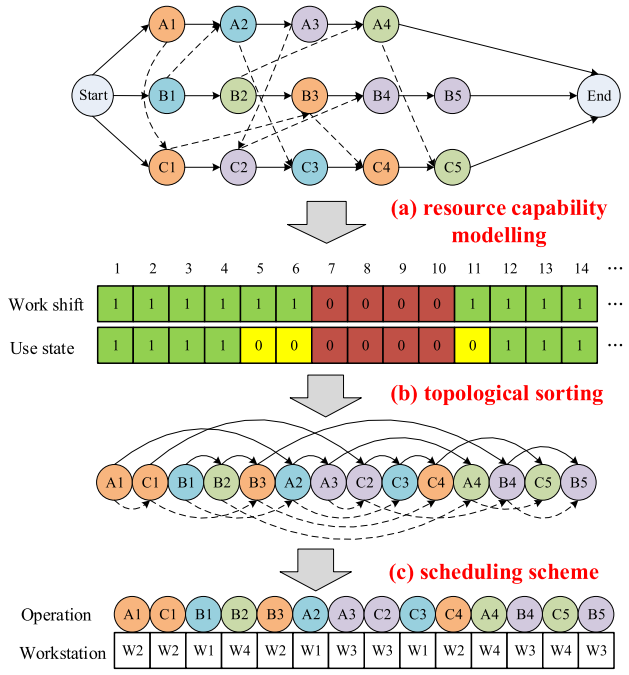


FIGURE 5. The disjunctive graph mode used in the MES.

machines are the main resource elements to be considered in real-world shop scheduling, the scheduler should arrange the operation task on the available machine throughout the manufacturing timeline.

To describe the time resource of machines, the use of a sequence of time-cell states is used to model the work shift (WT) and the use state (UT) [90]. As shown in stage (a) of Figure 5, the span of a time cell is denoted as Δt, which is defined by the scheduler ($\Delta t = 10$ min in this study). In the representation of WT, a value of 0 for a time cell indicates that this time cell is outside of normal working hours for this machine, whereas a value of 1 means working time. In the representation of UT, a value of 0 for a time cell means that the machine is free, whereas a value of 1 means that the machine is occupied. As shown in stage (a) of Figure 5, the work shift corresponds to time cells 1–6 and 11–14, whereas time cells 7–10 correspond to non-working time. The machine is occupied in cells 1–4 and 12–14, free in cells 5–6 and 11.

Once the directions of the disjunctive arcs have been determined, a directed graph is obtained. For a directed acyclic graph, topological sorting based on the evolved JSR and MAR is realized to arrange all nodes in a linear sequence. And then, the linear sequence is decoded into the representation of operation precedence in the MES. Finally, the machine selection and the operation sequence are created for each operation.

As shown in Figure 6, a detailed dispatching process based on a priority-rule-based construction method is presented. A complete SP comprises two types of decision rules: job sequencing rule (JSR) and machine assignment rule (MAR), which are applied to the corresponding decision points (a) and (c) of Figure 6. In this study, the JSR and MAR are trained offline by the MAHH to improve the generalization performance. And then, the selected SPs are used online for fast application.

The construction method based on the disjunctive graph model assumes that all tasks are available at time 0. Therefore, the job sequencing decision point does not occur in the queue of resource, but in the set of operations that can be assigned immediately O_{ready}. The whole evaluation process can be roughly divided into four steps.

1) Firstly, it calculates the priorities of all operation tasks in the scheduling scheme based on the JSR evolved by the MAHH.
2) The second step is to build the set of operations that can be assigned immediately O_{ready}, and then the operation A_i with the highest priority is selected from it. However, it is possible to have the same priority for different operations, so we construct a comparator of JSR (tie-breaker) to further distinguish their priorities. Hence, a lexicographic compactor is adopted to sort the priority, and the order is priority>due date>slack factor>operation number> unique identification of the operation task.
3) At the third stage, a set of alternative machines M_{alt} that can arrange operation A_i is built. The MAR evolved by the MAHH calculates the priority of each alternative machine and selects the machine with the highest priority from M_{alt}. For example, the operation A_i is assigned to machine M_1 in Figure 6. In addition, a lexicographic compactor is adopted to sort the priority of machine, and the order is priority> total occupancy time> unique identification of the machine.
4) Finally, the algorithm updates the resource capability, removes operation A_i from O_{ready} and adds the successor operation A_{i+1} to O_{ready}. These steps iterate until $O_{ready} = \Phi$ which means that all operation tasks have been arranged. When the dispatching process completed, the scheduling results are returned to assign fitness to the corresponding individual for evolution.

IV. PROPOSED METHODS

This section describes three MAHH methods, and each method includes four efficient multi-objective genetic programming algorithms to evolve SPs for the MO-FJSP.

A. ENCODING AND DECODING SCHEME

GP is a special kind of evolution algorithm that is characterized by its ability to evolve individuals of variable depth, where the solution candidates are represented by a tree structure with various depth rather than a fixed length string of genes. It should be noted that the tree depth is defined as the largest number of edges required to reach a leaf node. Typical applications of GP are the automatic creation of mathematical formulas or computer programs for solving a specific task. Therefore, it is rather suitable to use GP for the generation of composite SPs.

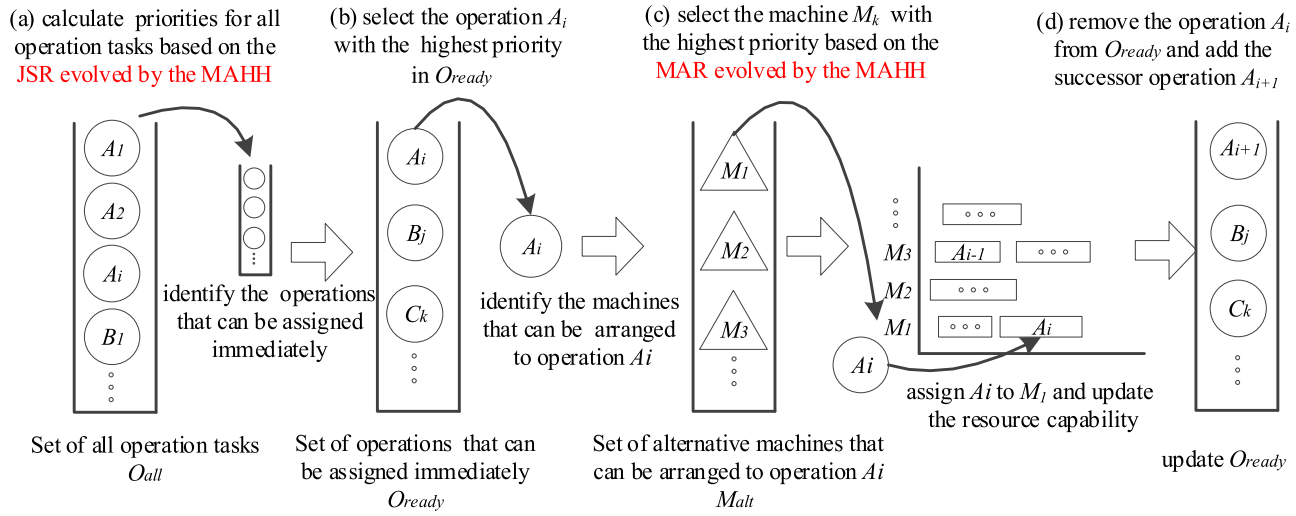


FIGURE 6. The detailed dispatching process based on priority rules.

TABLE 2. The terminal set for JSR.

Terminals	Description
SF	slack factor
TDD	time to job's due date
TOD	time to operation's due date
W	weight of job
PT	processing time of operation.
ERC	random number generating from (0, 1.0]

TABLE 3. The terminal set for MAR.

Terminals	Description
WW	weight (preference) of device
WPC	switching cost of device
WD	delay time of the device cost by the assigned operation
WU	total work hours of the device
ERC	random number generating from (0, 1.0]

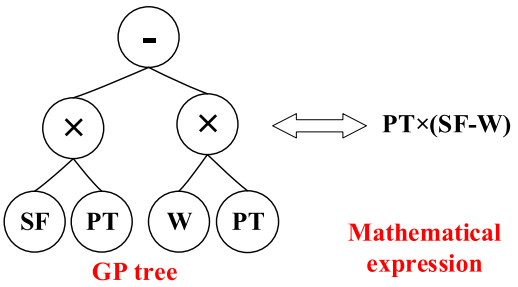


FIGURE 7. GP Tree representation of an exemplary JSR.

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. Since two kinds of rules are evolved in this study, the terminal set for constructing the JSR and the terminal set for constructing MAR are presented in Table 2 and Table 3, respectively. As shown in Table 2, the calculation method of the slack factor is shown in equation (7).

$$SF = \frac{d_i - r_i}{\sum_{j=1}^{n_j} p_{ij}} \tag{7}$$

where d_i, r_i and n_j is the delivery date, the release time and the number of operations of job j_i, respectively.$p_{i,j}$ is the processing time of operation task $o_{i,j}$ which is described in section III-B. SF denotes the ratio of the remaining delivery time to the remaining working hours of the jobj_i.

Figure 7 shows an example of the encoding of a JSR (PT×(SF-W)) as a GP tree structure. Then, the GP tree structure is decoded into a mathematical expression during the fitness evaluation process. As shown in stage (a) of Figure 6, the priority value of this representation is calculated according to the current system and used to sort the operation tasks in O_{ready}.

Individuals from the current population are selected using the double tournament selection [93] to handle bloat in GP. And new individuals are created after the selection process using the standard subtree crossover with probability P_c and the subtree mutation with probability P_m, respectively. 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. It should be stressed that only genetic materials from the same type of selected tree will be exchanged [94].

After all individuals have been evaluated, the archive will be updated by the two multi-objective approaches (NSGAII [95], SPEA2 [96]) to explore the Pareto front of nondominated SPs. Then, the breeding step is realized using the genetic operations (selection, crossover and mutation). New sub-populations are produced and the algorithm begins a new generation if the maximum generation is not achieved.

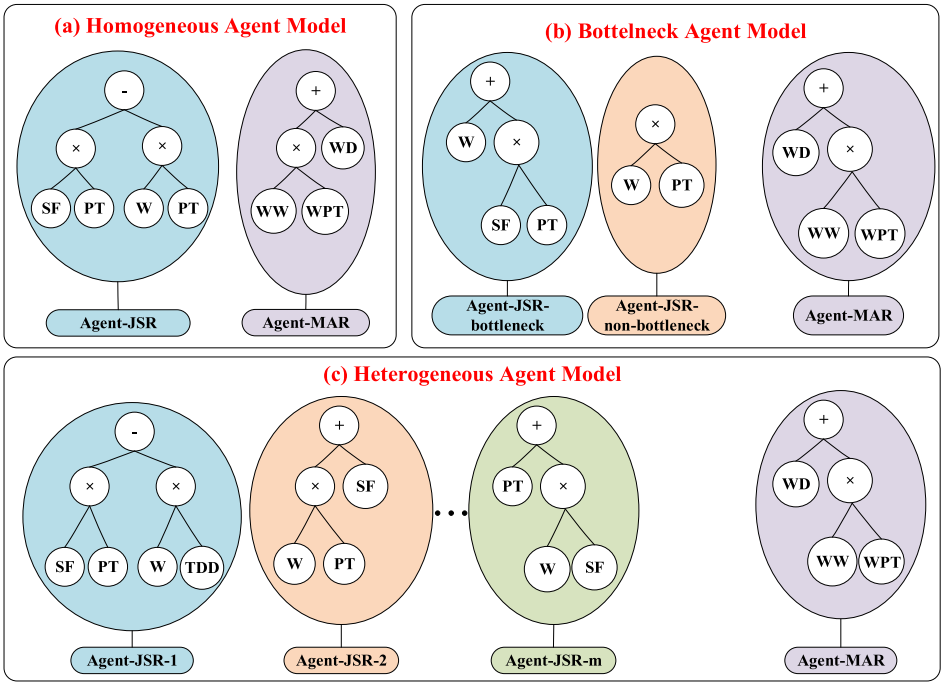


FIGURE 8. Three models of multi-agent based hyper-heuristics. (a) Homogeneous Agent Model. (b) Bottelneck Agent Model. (c) Heterogeneous Agent Model.

B. THREE MODELS OF MULTI-AGENT STRUCTURE

In practical scheduling, each workstation has a unique scheduling strategy due to processing characteristics. Therefore, the three models of multi-agent structure are developed to evolve JSR for different types of workstations. As for MAR, only one agent is used at the machine assignment decision point. In addition, two types of collaborative modes are used in the evolution of genetic programming, namely multi-populations and multi-trees.

Figure 8 demonstrates the three models of multi-agent structure, and Figure 8(a) shows the homogeneous agent model. In this model, each workstation uses the same JSR which is evolved by a single GP population to calculate the priority indexes of operations. Each job also uses the same MAR evolved by the other single GP population to calculate the priority indexes of workstations. Hence, this model utilizes the cooperative coevolution genetic programming with two sub-populations (2PGP) or genetic programming with two sub-trees (2TGP) to formulate a complete SP, including a JSR and a MAR. The benefit of this model is that it is easy to implement in practice and it has good robustness in different environments. However, since all workstations share the same JSR, the rule may become extremely complicated because of the idiosyncrasy of each workstation.

Figure 8(b) shows the bottleneck agent model. In practical scheduling, scheduler pays more attention to the load status of the bottleneck resource. According to the theory of constraints, the bottleneck resources restrict the overall throughput. Therefore, two types of agents including bottleneck agent and non-bottleneck agent are proposed to learn independent JSRs for operation priority evaluation. It should be stressed that there is no immigration and crossover of individuals among populations in different agents.

The bottleneck agent model utilizes the cooperative coevolution genetic programming with three sub-populations (3PGP) or genetic programming with three sub-trees (3TGP) to formulate a complete SP, including two JSRs and a MAR. In the practical scheduling, the bottleneck workstations are not fixed and may change dynamically during production process. The disadvantage of this model is that it needs to be retrained after the bottleneck changed. However, for the case studied in this work, the production process is relatively fixed, and the bottleneck resources have been determined according to the long-term experience of the scheduler. The benefit of this model is that the prior knowledge of bottleneck resource is integrated into the algorithm to guide the search direction, accelerate the convergence of the algorithm and improve the solution quality.

Figure 8(c) shows the heterogeneous agent model. In this model, each workstation acts as a unique agent and learns distinct JSRs evolved by multi-populations to calculate the priority indexes of operations in its queue. It should be stressed that there is no immigration and crossover of individuals among populations in different agents. Hence, this model utilizes the cooperative coevolution genetic programming with multi-populations (MPGP) or genetic programming with multi-trees (MTGP) to formulate a complete SP, including m JSRs and a MAR. The number of populations for JSR equals to the number of workstations. The disadvantage of this model is that it is difficult to apply in practice and it needs to

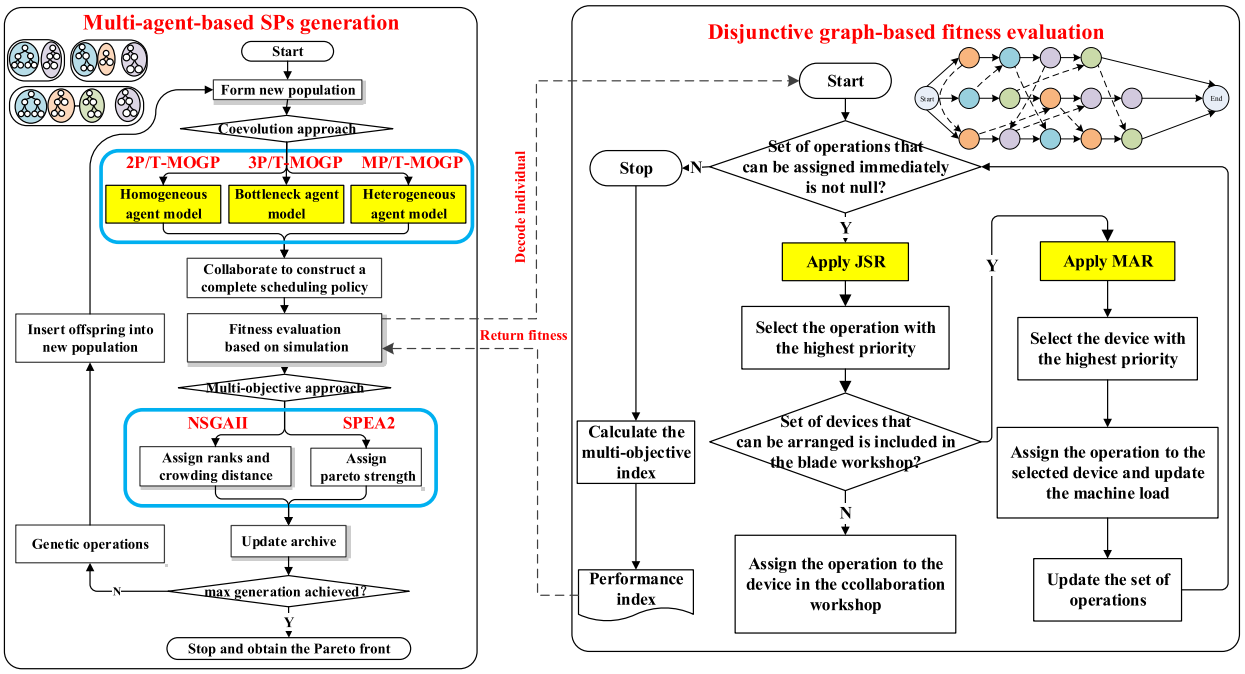


FIGURE 9. Framework of the multi-agent-based SPs generation and the disjunctive graph-based fitness evaluation.

be retrained after workstation changes. However, since each workstation has its own JSR, each agent can improve the quality of its local scheduling by customizing its behaviour to its situated environment and coordinating to optimize the global scheduling results.

Figure 9 illustrates the framework of the proposed MAHH. The left side shows the flow chart of the MAHH for the coevolution of the MAR and JSR, including the 2/3/MPGP and the 2/3/MTGP. In the 2/3/MPGP, each individual in its population cooperates with the individual randomly selected from other sub-populations to formulate a complete SP. In the 2/3/MTGP, each individual includes multi-trees and these sub-trees are collaborated to formulate a complete SP.

According to different models of multi-agent structure, different JSRs are utilized by different types of workstations to evaluate the priority of operations. For the homogeneous agent model, all workstations use the same JSR by considering the generalization performance. For the heterogeneous agent model, different types of workstations use different JSRs to achieve better scheduling result quality. For the bottleneck agent model, the bottleneck and non-bottleneck resources utilize their own JSRs by considering both the effectiveness and generalization performance. However, in order to improve the generalization performance of the proposed methods, only one MAR is evolved for all jobs. This is because machines are relatively fixed in shop floor, whereas part tasks are often changed in real production. These decision rules (JSR and MAR) from a complete SP are applied to the relevant decision points (stage*a* and *c* of Figure 6) in the disjunctive graph model. A detailed flow chart of the fitness evaluation is shown on the right side of Figure 9. When the evaluation process finished, the results are collected and returned to assign fitness to the individuals.

C. PROCEDURES OF THE PROPOSED ALGORTIHMS

There are $3\times2\times2=12$ algorithms proposed in this study, including 2PGP-NSGAII, 3PGP-NSGAII, MPGP-NSGAII, 2TGP-NSGAII, 3TGP-NSGAII, MTGP-NSGAII, 2PGP-SPEA2, 3PGP-SPEA2, MPGP-SPEA2, 2TGP-SPEA2, 3TGP-SPEA2 and MTGP-SPEA2. They are constructed by three models of multi-agent structure, two patterns of cooperative coevolution models and two types of multi-objective algorithms. In this section, the bottleneck agent model (3PGP-NSGAII/SPEA2) is taken as an example to introduce the procedures of the proposed algorithms. The pseudocode is listed as follows.

Step 1: The initial GP individuals are randomly generated by using ramped half-and-half method (depth 2 to 6) [97]. The 3PGP-NSGAII/SPEA2 method starts with three randomly generated sub-populations, one sub-population P_{jsr-b} for JSR of the bottleneck machine, one sub-population P_{jsr-nb} for JSR of the non-bottleneck machine and another sub-population P_{mar} for MAR.

Step 2: The individual R^i_{jsr-b} from the sub-population P_{jsr-b} is paired with the individual R^j_{jsr-nb} from the sub-population P_{jsr-b} and the individual R^k_{mar} from the sub-population P_{mar} using random shuffling. This method has shown to be effective for coevolution of JSR and MAR in our previous work [94]. R^t_{rep} is the complete SP that is formed by the combination of (R^i_{jsr-b},R^j_{jsr-nb}, R^k_{mar}).

Algorithm 1 3PGP-NSGAII/SPEA2

Inputs: k-fold cross-validation scenarios $S \leftarrow \{S_1, S_2, \ldots, S_k\}$

Outputs: the Pareto front of nondominated scheduling policies PF_{known}

(1) initialize population $P_{jsr-b}, P_{jsr-nb}, P_{mar}$ at random, $P_{jsr-b} \leftarrow \{R^1_{jsr-b}, \ldots, R^n_{jsr-b}\}$, $P_{jsr-nb} \leftarrow \{R^1_{jsr-nb}, \ldots, R^n_{jsr-nb}\}$, $P_{mar} \leftarrow \{R^1_{mar}, R^2_{mar} \ldots R^n_{mar}\}$
(2) generation $\leftarrow$0, archive A $\leftarrow$ { }
(3) **while** generation $\leq$max Generation do
(4) pair up the R^i_{jsr-b}, R^j_{jsr-nb}, R^k_{mar} using random shuffling
(5) **for** all $R^k_{mar} \in P_{mar}$ do
(6) $R^t_{rep} \leftarrow$collaborate (R^i_{jsr-b}, R^j_{jsr-nb}, R^k_{mar})
(7) obtain average $f(R^t_{rep})$ by applying R^t_{rep} to each scenario $S_k \in S$ using one replication
(8) $f(R^i_{jsr-b})$, $f(R^j_{jsr-nb})$, $f(R^k_{mar}) \leftarrow f(R^t_{rep})$
(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 produce new populations
(13) generation $\leftarrow$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}

Step 3: In the fitness evaluation stage, k-fold cross-validation scenarios S (more details are shown in section V-B) are loaded to evaluate the performance of a complete SP R^t_{rep}. The fitness of (R^i_{jsr-b},R^j_{jsr-nb}, R^k_{mar}) is obtained by applying R^t_{rep} to S using one replication [97] and it is measured by the average value of the specific objective across all training scenarios. The scheduling results returned to assign fitness to the individual (R^i_{jsr-b}, R^j_{jsr-nb}, R^k_{mar}).

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 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 twelve proposed algorithms in 25 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}.

The proposed 3TGP-NSGAII/SPEA2 methods are similar to the 3PGP-NSGAII/SPEA2 algorithms. The difference is that an individual in 3TGP-NSGAII/SPEA2 contains three sub-trees for three decision rules when decoding for fitness evaluation. In this case, each individual is equivalent to a complete SP. The procedures of the 3TGP-NSGAII and 3TGP-SPEA2 can be described as follows.

Algorithm 2 3TGP-NSGAII/SPEA2

Inputs: k-fold cross-validation scenarios $S \leftarrow \{S_1, S_2, \ldots, S_k\}$

Outputs: the pareto front of nondominated scheduling policies PF_{known}

(1) initialize population P at random, $P \leftarrow \{R_1, R_2 .. R_n\}$
(2) generation $\leftarrow$0čarchive A $\leftarrow$ { }
(3) **while** generation $\leq$max Generation do
(4) **for** all $R_i \in$Pdo
(5) obtain average $f(R_i)$ by applying $R_i(R^i_{jsr-b}, R^i_{jsr-nb}, R^i_{mar})$ to each scenario $S_k \in$S using 1 replication
(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 pareto front PF_{known}
(13) **return** PF_{known}

D. PARAMETER SETTINGS

Table 4 shows the parameter settings of the proposed algorithms. In order to compare the twelve algorithms for fair comparison, the number of fitness evaluations (NFE) for all the algorithms is set to 10000. Each population in all the algorithms has 100 individuals. The homogeneous agent model adopts two sub-populations/trees, the bottleneck agent model adopts three sub-populations/trees, and the heterogeneous agent model adopts seven sub-populations/trees. The reason is that there is a total of five types of workstation in the areo-engine blade workshop investigated in this study, and the workstation in co-operation unit is also defined as one agent. Therefore, there are 6 agents for the evaluation of JSR and an agent for the evaluation of MAR, and the total number of agents is seven in the heterogeneous agent model.

For the NSGAII-based methods, generations are set to 50 and the population size is fixed to 100. Because the NSGAII-based methods evaluate both the parent and child individuals in the evaluation process, the NFE per

TABLE 4. The parameter settings of the proposed algorithms.

Algorithm	(2/3/M)PGP-NSGAII	(2/3/M)TGP-NSGAII	(2/3/M)PGP-SPEA2	(2/3/M)TGP-SPEA2
Sub-population number	2/3/7	1	2/3/7	1
Sub-tree number	1	2/3/7	1	2/3/7
Generations	50	50	100	100
Population size	100			
Computational budget	10000 NFE			
Crossover/mutation rates	90%/10%			
Initialization	ramped half-and-half (depth 2~6)			
Selection	double tournament selection			
Maximum depth	8			
Operators	+, −, ×, /, max, min			
Terminals	Shown in Table 2 and Table 3			

generation is 100×2= 200(except for the first generation), and the total NFE is 50×200=10000. For the SPEA2-based methods, the generation is set to 100 and the population size is also fixed to 100. Therefore, the NFE per generation is 100, and the total NFE is 100×100=10000. These settings are applied to achieve the same NFE for fair comparison.

Similar to other evolutionary algorithms, GP starts its evolution with a randomly generated population. To diversify the initial individuals with various depths, lengths and shapes, GP often employs ramped half-and-half method. In this approach, a maximum depth is determined for each individual to restrict the program tree depth. To reduce the computational time of the GP system and make the evolved SPs easier to analyze, the depth is set to 2∼6 for ramped half-and-half initialization and the maximum depth is set to 8 for crossover and mutation [83]. According to our preliminary experiments, these parameter settings perform well in both effectiveness and robustness.

V. EXPERIMENTAL RESULTS

In this study, twelve proposed algorithms were employed to produce SPs to solve the MO-FJSP. The overall experiment process is described as follows: At first, a k-fold cross validation is designed based on the practical case investigated in this study. And then, the twelve proposed algorithms are applied to the training set to compare the multi-objective performance differences among them. Finally, the twelve Pareto solution sets (non-dominated SPs) generated by the twelve algorithms were applied to the test set to obtain the generalization performance. For each performance metric, a Wilcoxon signed-rank test with the significance level of 0.05 [98] is carried out on the results obtained by 25 independent runs of each method. The experiments were all implemented in Java 8.0 based on the Eclipse platform and run on a computer with Intel Core i5-4590 3.30 GHz, 8 GB RAM.

A. PERFORMANCE METRICS

Three popular metrics were employed to evaluate the performances of the proposed methods: Hypervolume Ratio (HVR) [99], Inverted Generational Distance (IGD) [100], and Spacing [101]. They can be described as follows:

- Hypervolume ratio (HVR): hypervolume is used to measure the size of the objective space dominated by the obtained non-dominated front PF_{konwn}. A higher HV value is desirable and denotes a good dominate performance.

$$HV = volume(\bigcup_{i=1}^{n_{PF}} v_i) \tag{8}$$

  were 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 is the diagonal of the hypercube [95]. HVR is the ratio of the HV of PF_{konwn} and the HV of the reference Pareto front PF_{ref}.

$$HVR = \frac{HV(PF_{konwn})}{HV(PF_{ref})} \tag{9}$$

- 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.

$$IGD = \frac{(\sum_{i=1}^{n} d_i^p)^{1/p}}{n} \tag{10}$$

  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}.

$$Spacing = \sqrt{\frac{1}{n_{PF}-1}\sum_{i=1}^{n_{PF}}(\overline{d}-d_i)^2} \tag{11}$$

  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. However, there are no standard results for reference in this study. Therefore, a reference Pareto front is adopted to calculate these performance metrics. In this study, the evolved SPs from the twelve algorithms in all independent runs are combined into a common pool, and the nondominated sorting technique is used to extract the true Pareto front PF_{ref} from this pool.

B. DESIGN OF EXPERIMENTS

In the related literature about the automated design of production scheduling heuristics, the discrete event system simulation is often used for fitness evaluation [73], [74], [75].

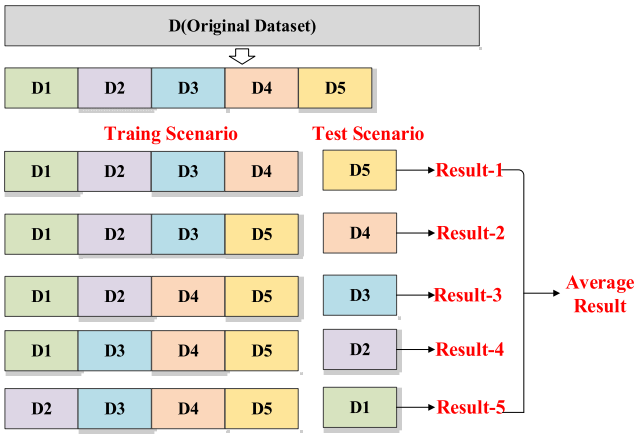


FIGURE 10. The schematic diagram of the 5-fold cross validation.

Therefore, the random number is usually changed per generation to increase the generalization performance of the evolved SPs [97]. However, the problem studied in this work comes from the real case, and the training set cannot be generated randomly. To improve the generalization performance of the evolved SPs, the training set is randomly partitioned into learning data sets via a k-fold cross validation. The training set is divided into k non-overlapping groups. The scheduling policies are trained using the first k–1 groups of training set, and the non-dominated SPs are tested on the kth group. We repeat this procedure until each of the groups is used once as a test set. To reduce the error caused by the different sample partition, the k-fold cross validation is usually repeated p times using different partition methods at random. In this study, we set k =5 and p =5. This is a reasonable compromise considering the computational budget of the case study.

As shown in Figure 10, the principle of the *5*-fold cross validation is to determine the parameters contained in the algorithm by using different training samples. For the scheduling problem studied in this work, the twelve algorithms utilize the *5*-fold cross validation to obtain their learning results (non-dominated SPs), and then calculate the error of each result, and select the model with the smallest error as the final model. It should be noted that in supervised learning, the error is mainly calculated according to the difference between the classification or predicted results and the benchmark results. However, there is no optimal solution as the reference for the unsupervised learning. Besides, the problem investigated in this study is MO-FJSP which cannot be measured according to the difference between single target. Therefore, multi-objective performance metrics are employed in this study to compare their differences between each Pareto set learned by each algorithm, so as to measure the generalization performance of each algorithm.

The case study of the areo-engine blade workshop contains 18 batches of part tasks (shown in Figure 2). To simulate the situation of the workshop under different load status, the sample construction method is carried out by randomly removing some batches of part tasks. The method is described as follows.

1) As shown in Figure 10, the five samples for the *5*-fold cross validation are constructed from the original data set which includes 18 batches of part tasks by using the method of random sampling without replacement. Therefore, the data sets are divided into 5 groups which includes 14, 15, 16, 17 and 18 batches of part tasks. According to the 5-fold cross validation, the SPs are evolved using the first four groups of the samples and tested on the *5*th group. This procedure is repeated until each of the groups is used once as a test set.
2) To reduce the error caused by different sample partition methods, the *5*-fold cross validation is repeated 5 times using different partition methods at random. Hence, a total of 5∗5=25 training sets and 25 corresponding test sets are created in this study. Therefore, 25 independent experiments of the proposed method are performed.
3) In each experiment, twelve Pareto sets are generated from the twelve algorithms after evolving on each training set. The reference Pareto front PF_{ref} is extracted from all the twelve Pareto sets found by the twelve proposed algorithms. Therefore, the multi-objective performances of the twelve Pareto sets are calculated according to the reference Pareto front PF_{ref}. And then, the non-dominated SPs from the Pareto set are applied to the corresponding test set to test the generalization performance of the evolved SPs.
4) Because there are 25 training sets and 25 test sets designed in this study, a total of 25 training results and 25 test results are recorded and compared to verify that the performance of the twelve algorithms is consistent on the two types of sets.

C. TRAINING PERFORMANCE

As mentioned above, 25 independent runs are carried out for each algorithm. In each experiment, twelve pareto sets are generated from the twelve algorithms after evolving on each training set. And then, the reference Pareto front PF_{ref} is extracted from the twelve Pareto sets using non-dominant sorting method. The three multi-objective performance metrics of the twelve Pareto sets, which includes HVR, IGD and Spacing, are calculated according to the reference Pareto front PF_{ref}. The performance of the twelve proposed methods from 25 independent runs on the training scenarios are shown in Figure 11 (better methods have higher HVR and smaller IGD, Spacing). As shown in Figure 11, the pink circle denotes the average result and the red horizontal line represents the median result of the corresponding algorithm.

Table 5 demonstrates the median, interquartile range (IQR), mean and standard deviation (std) of each indicator as performance measures of each algorithm. It can be observed that there is no significant difference between the twelve algorithms on training time. This is because they use the

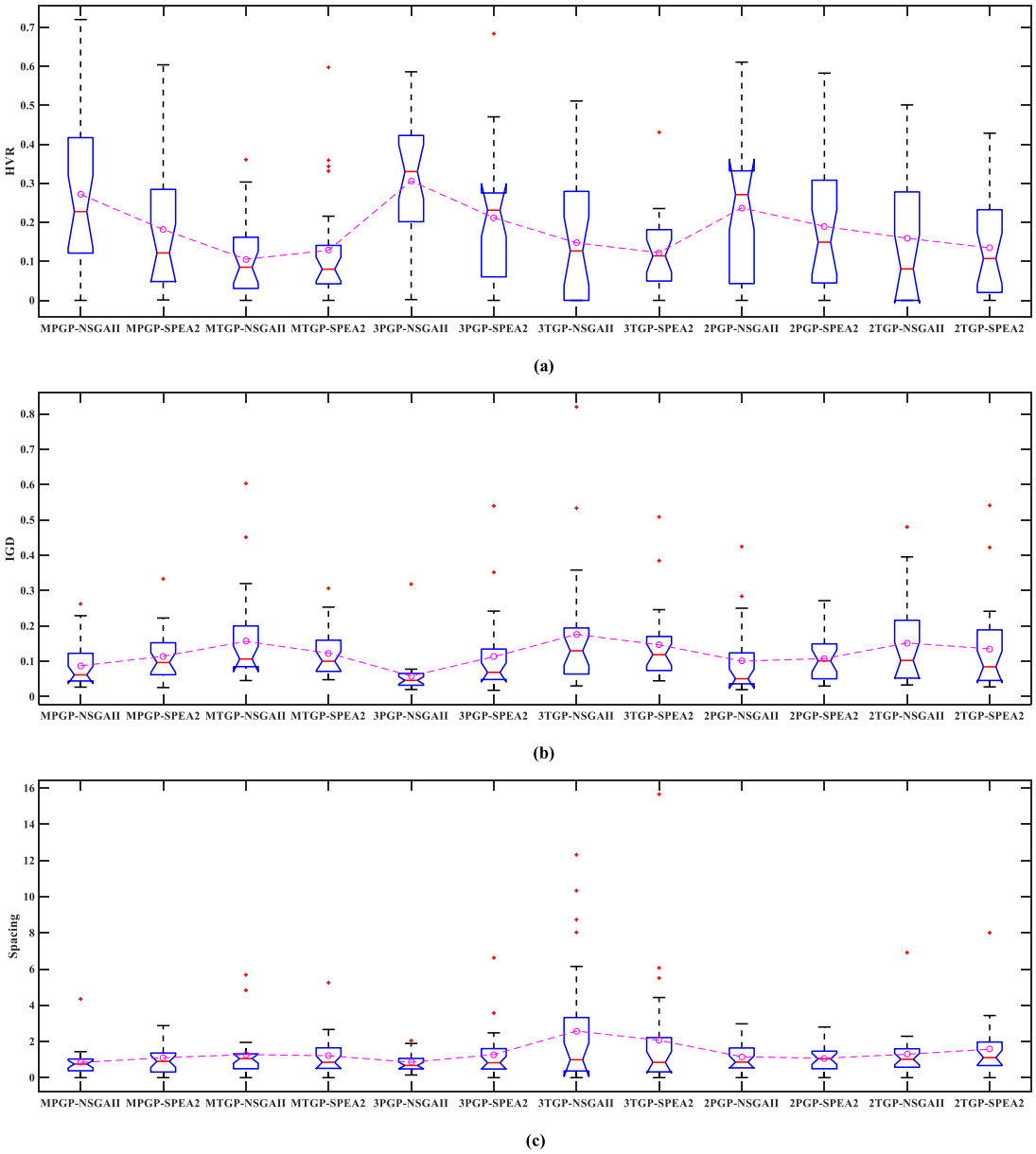


FIGURE 11. Performances of the proposed methods on the training scenarios. (a) HVR. (b) IGD. (c) Spacing.

same NFE and most of the running time spends on the fitness evaluation process. It should be noted that the training time is not particularly important because the training process can be carried out off-line. When the evolved SP is implemented online for fast application, it takes less than a few seconds as much time as a man-made rule.

It can be seen that the HVRs, IGDs and Spacings produced by the multi-population GP algorithms are significantly better than that of multi-tree GP algorithm under the same multi-agent model structure and the same multi-objective method. Similarly, the HVRs, IGDs and Spacings produced by NSGAII are significantly better than that of SPEA2 under the same multi-agent model structure and the same cooperative pattern. Therefore, we pay more attention to the multi-population GP algorithm based on NSGAII (2/3/MPGP-NSGAII).

For each performance metric, a Wilcoxon signed-rank test with the significance level of 0.05 [98] is carried out on the results obtained by 25 independent runs of each method. Table 6 summarizes the statistical training and test results of the multi-population GP algorithm based on NSGAII (2/3/MPGP-NSGAII). For each performance metric, the sign of '+', '−', '=' in method A vs. B indicates that according to the metric, approach A is significantly better than B (this case is already marked in bold), significantly worse than B, or there is no significant difference between A and B based on the Wilcoxon signed rank test with the significance level of 0.05.

TABLE 5. Performance of the proposed methods on the training scenarios.

Model	HVR		IGD		Spacing		Running Time	
	Median (IQR)	Mean±Std	Median (IQR)	Mean±Std	Median (IQR)	Mean±Std	Median	Mean±Std
MPGP-NSGAII	0.228 (0.296)	0.272±0.209	0.062 (0.078)	0.087±0.062	0.750 (0.648)	**0.839**±0.825	223.206	246.403±58.213
MPGP-SPEA2	0.122 (0.236)	0.182±0.164	0.097 (0.090)	0.114±0.071	0.899 (1.048)	1.101±0.872	224.965	247.751±59.990
MTGP-NSGAII	0.085 (0.131)	0.105±0.101	0.107 (0.115)	0.156±0.131	1.052 (0.823)	1.278±1.304	233.780	253.323±54.501
MTGP-SPEA2	0.080 (**0.099**)	0.128±0.141	0.100 (0.088)	0.123±0.068	0.849 (1.133)	1.211±1.121	231.541	252.334±60.188
3PGP-NSGAII	**0.330** (0.221)	**0.306**±0.164	**0.046** (**0.033**)	**0.058±0.057**	**0.674** (**0.581**)	0.851±**0.503**	214.711	236.824±56.347
3PGP-SPEA2	0.231 (0.215)	0.212±0.173	0.069 (0.086)	0.113±0.117	0.825 (1.127)	1.278±1.428	218.257	238.011±57.806
3TGP-NSGAII	0.127 (0.280)	0.147±0.156	0.130 (0.130)	0.176±0.177	0.991 (2.956)	2.577±3.585	223.662	244.338±54.275
3TGP-SPEA2	0.114 (0.132)	0.122±**0.099**	0.119 (0.096)	0.146±0.108	0.844 (1.897)	2.065±3.330	217.673	242.246±60.650
2PGP-NSGAII	0.271 (0.289)	0.237±0.180	0.051 (0.087)	0.101±0.103	0.862 (1.100)	1.156±0.813	222.716	243.322±57.573
2PGP-SPEA2	0.149 (0.263)	0.190±0.160	0.101 (0.099)	0.108±0.070	1.041 (0.973)	1.075±0.744	226.337	245.574±57.928
2TGP-NSGAII	0.081 (0.278)	0.159±0.172	0.102 (0.164)	0.152±0.132	1.019 (1.021)	1.294±1.308	220.932	239.677±56.307
2TGP-SPEA2	0.107 (0.212)	0.135±0.125	0.084 (0.143)	0.135±0.125	1.108 (1.297)	1.575±1.600	220.320	240.323±55.791

TABLE 6. p-values of the statistical test for each metric on training and test set.

Comparison	Training			Test		
	HVR	IGD	Spacing	HVR	IGD	Spacing
3PGP-NSGAII vs. 2PGP-NSGAII	**0.037**+	**0.015**+	0.083=	**<0.0001**+	**<0.0001**+	<0.0001-
3PGP-NSGAII vs. MPGP-NSGAII	0.201=	**0.02**+	0.476=	0.375=	**0.048**+	0.527=
MPGP-NSGAII vs. 2PGP-NSGAII	0.353=	0.819=	0.115=	**<0.0001**+	**<0.0001**+	<0.0001-

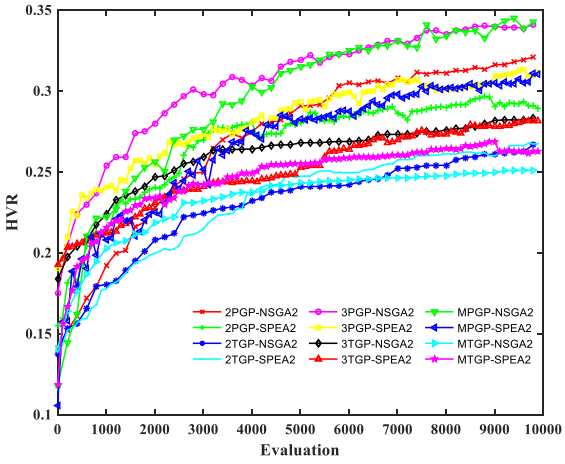
As shown in Table 5 and Table 6, we found that the 3PGP-NSGAII achieve the best performance on metrics of HVR and IGD among the three multi-agent models. Although there is no significant difference between the three MAHH methods on metric of Spacing, we focus more on the other two metrics in this study. It can be concluded that 3PGP-NSGAII performs better than MPGP-NSGAII and 2PGP-NSGAII on the training scenarios.

To demonstrate the detailed evolution process of the twelve proposed algorithms, Figure 12 shows the average performance of HVR, IGD and Spacing from all 25 independent runs across generations on the training set, respectively. The detailed results from Figure 12 (a) and (b) show that the 3PGP-NSGAII method can find good scheduling policies much faster than MPGP-NSGAII and 2PGP-NSGAII. As shown in Figure 12 (c), although the performances of Spacing are not stable across generations, 3PGP-NSGAII still performs better than other algorithms. In summary, the results of the homogeneous agent model are worse than the results of the heterogeneous and the bottleneck agent model. This is because the homogeneous agent model did not integrate the prior knowledge (i.e., which resources are bottleneck) into the algorithm, which resulted in poor performance. In addition, the flexibility of the heterogeneous agent model causes over-fitting to the training set. And the SPs evolved by the heterogeneous agent model are too complicated to implement in practice scheduling. If resources are changed in shop floor, the heterogeneous agent model needs to be retrained. Therefore, the bottleneck agent model makes a good trade-off between the solution quality and the generalization performance among the three agent models.

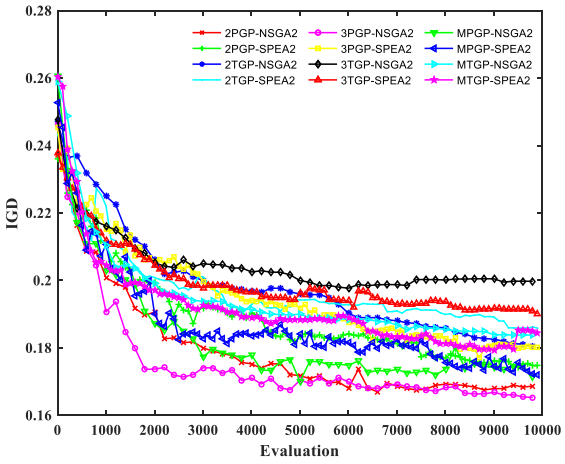
In addition, there is a close relationship between the interpretability and the generalizability of SPs with the program length of the evolutionary SPs. Therefore, we recorded the average program length of a complete SP from the Pareto front PF_{known} in each independent run. According to Occam's Razor, 'Entities should not be multiplied unnecessarily'. Therefore, simple evolutionary SPs are easier to implement in complex scheduling scenarios. As shown in Figure 13, the average program length of the 2/3/MPGP is generally higher than that of 2/3/MTGP. This is because the cooperative multi-population GP algorithm can find more problem related features than the single population GP method.

Because a scheduling policy contains two types of rules, the program length is shown separately for the JSR and the MAR in Figure 14. In the homogeneous agent model, the average length of JSRs is much longer than that of MAR because only one agent is utilized to evolve JSR. In the bottleneck agent model, the JSRs of the bottleneck agent model are divided into two types. The average length of the JSR in the bottleneck agent is greater than that of the non-bottleneck agent, which indicates that the job sequencing problem on bottleneck resource is relatively more difficult than that on non-bottleneck resource. This is because the bottleneck agent contains more problem characteristics. However, both the average length of JSRs are smaller than that of MAR. In the heterogeneous agent model, the JSRs are divided into six categories according to the device type in this study. And the average length of the six types of JSRs is also smaller than that of MAR.

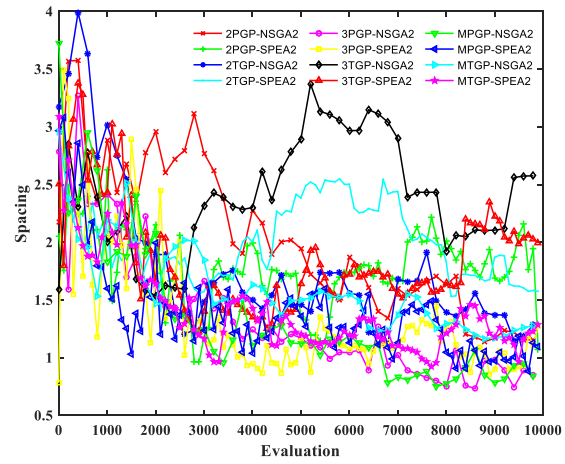
To conclude, the average length of the evolved JSRs decreases significantly with the increase of agents. Moreover, shorter SP will make it easier to interpret, and evaluations of such SPs are also faster. Besides, the generalization performance of the evolved SPs will be deteriorated when the



(a)



(b)



(c)

FIGURE 12. Average performance metrics of the proposed methods across generations. (a) HVR. (b) IGD. (c) Spacing.

number of agents increases. Therefore, the bottleneck agent model succeeded to make a good balance in solution quality and generalization performance among the three agent models on the training scenarios.

D. TEST PERFORMANCE

To validate the generalization performance of the non-dominated SPs evolved by the proposed methods, they are applied to the corresponding test set to obtain the test results which can be regarded as the model error. Because there is no reference result to calculate the performance differences, we use the three multi-objective performance indicators (HVR, IGD, Spacing) to compare with each other in the twelve algorithms. The following Figure 15 illustrate the performances of the twelve algorithms from 25 independent runs on the test set.

It can be seen that the test results are basically consistent with the training results. The detailed results from Figure 15 and Table 6 show that the metrics of HVR and IGD produced by 3PGP-NSGAII are significantly better than other methods. Despite the poor performance in terms of Spacing, its comprehensive performance is still the best among the 12 algorithms. Therefore, these detailed results confirm the above conclusions that the bottleneck agent model outperform the other two agent models proposed in this study.

VI. FURTHER ANALYSIS

To evaluate the actual scheduling performance of the evolved SPs, the non-dominated evolved SPs extracted from the test results of the MAHH will be applied to the real case to compare with the heuristics reported in the literature and the well-known meta-heuristics used to solve the MO-FJSP.

A. COMPARISON WITH EXISTING MAN-MADE SPs

To compare the performance differences between the non-dominated evolutionary SPs and the existing man-made SPs, they are simultaneously applied to the real case in the aero-engine blade manufacturing plant. The evolutionary SPs are extracted from the 25 Pareto sets generated by the MAHH on the 25 test scenarios. And the benchmark SPs are made up of 16 well-known JSRs and 6 MARs which are shown in Table 7 and Table 8, respectively. Because the ATC rule contains parameter k, the grid search method is employed to find the most appropriate parameter in this study. The range of k is from 0.1 to 10.0 with the step size 0.1. Besides, the man-made SPs are utilized based on the bottleneck agent model which includes two types of JSRs. Therefore, there is a total of 16 JSR-B $\times$ 16 JSR-NB $\times$ 6 MAR $=$ 1536 combinations of benchmark SPs used for the comparison with the evolved SPs.

As shown in Figure 16, the benchmark SPs can achieve nearly the same performance as the evolutionary SPs under the objective T_{max}. However, the evolutionary SPs outperform all benchmark SPs under two objectives W_{mean} and T_{mean}. The reason is that T_{max} is a maximum objective which is difficult to be optimized, and the average objectives (W_{mean} and T_{mean}) are easier to be optimized for hyper-heuristics. In addition, it can be observed that the three objectives W_{mean}, T_{mean} and T_{max} are conflicting with each other.

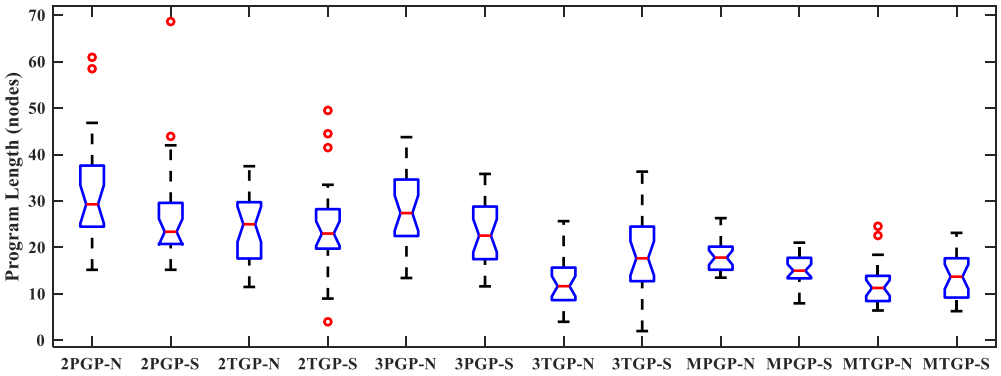


FIGURE 13. Average length of the evolved SPs ('N' and 'S' denote the NSGAII and SPEA2 method, respectively).

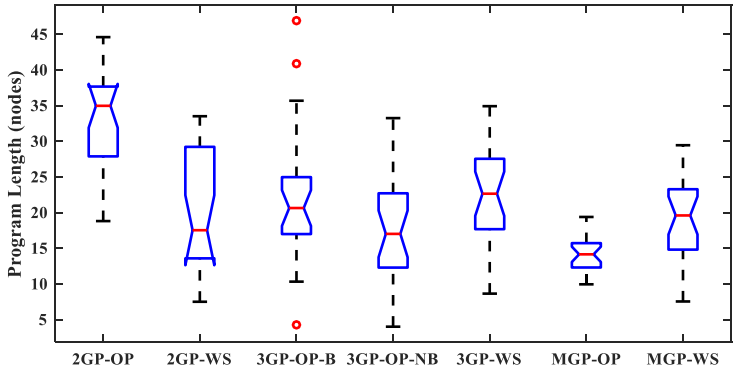


FIGURE 14. Average length of the evolved JSRs and MARs ('op' and 'ws' represent the program length of the JSR and MAR, respectively).

TABLE 7. Benchmark job sequencing rules.

No.	JSR	Description
1	ATC	apparent tardiness cost
2	CR	critical ratio
3	CR+SPT	critical ratio plus processing time
4	EDD	earliest due date
5	LPT	longest processing time
6	LRPT	longest remaining processing time
7	MDD	modified due date
8	MOD	modified operation due date
9	ODD	operation due date
10	RND	random rule
11	SLK	slack
12	SPT	shortest processing time
13	SRN	shortest remaining operation number
14	SRPT	shortest remaining processing time
15	SRPT/PT	remaining processing time per imminent processing time
16	SRPT/SLK	remaining processing time per slack

TABLE 8. Benchmark machine assignment rules.

No.	MAR	Description
1	WW	the weight (preference) of device
2	WPC	the switching cost of the device caused by the same process content
3	WSC	the switching cost of device caused by the same drawing identification
4	WCC	total idle time of the device
5	WD	total delay time of device by the assigned operation
6	WU	total work hours of the device

Figure 16(c) shows that the two objectives W_{mean} and T_{mean} have a positive correlation, while Figure 16(d) shows that the two objectives T_{max} and T_{mean} have a negative correlation. These observations show that handling multi-objective is almost impossible for the manual design of effective SPs. Benefit from the proposed methods, the evolved SPs dominate nearly all the man-made SPs under any objective on the real case.

B. COMPARISON WITH META-HEURISTICS

Although the experiment results show that the proposed MAHH perform better than the well-known heuristics in solving the MO-FJSP, it still needs to be compared with the meta-heuristics to show its effectiveness and efficiency. In recent years, many swarm intelligence algorithms are used to solve the MO-FJSP [59], [60], [61]. Therefore, two well-known multi-objective particle swarm optimization (MOPSO) [102] algorithms including optimized multi-objective particle swarm optimization (OMOPSO) [103] and speed-constrained multi-objective particle swarm optimization (SMPSO) [104] are introduced to compare with the evolved SPs produced by the proposed MAHH method in this study.

Instead of calculating priorities for all operation tasks based on the JSR evolved by the MAHH, OMOPSO and

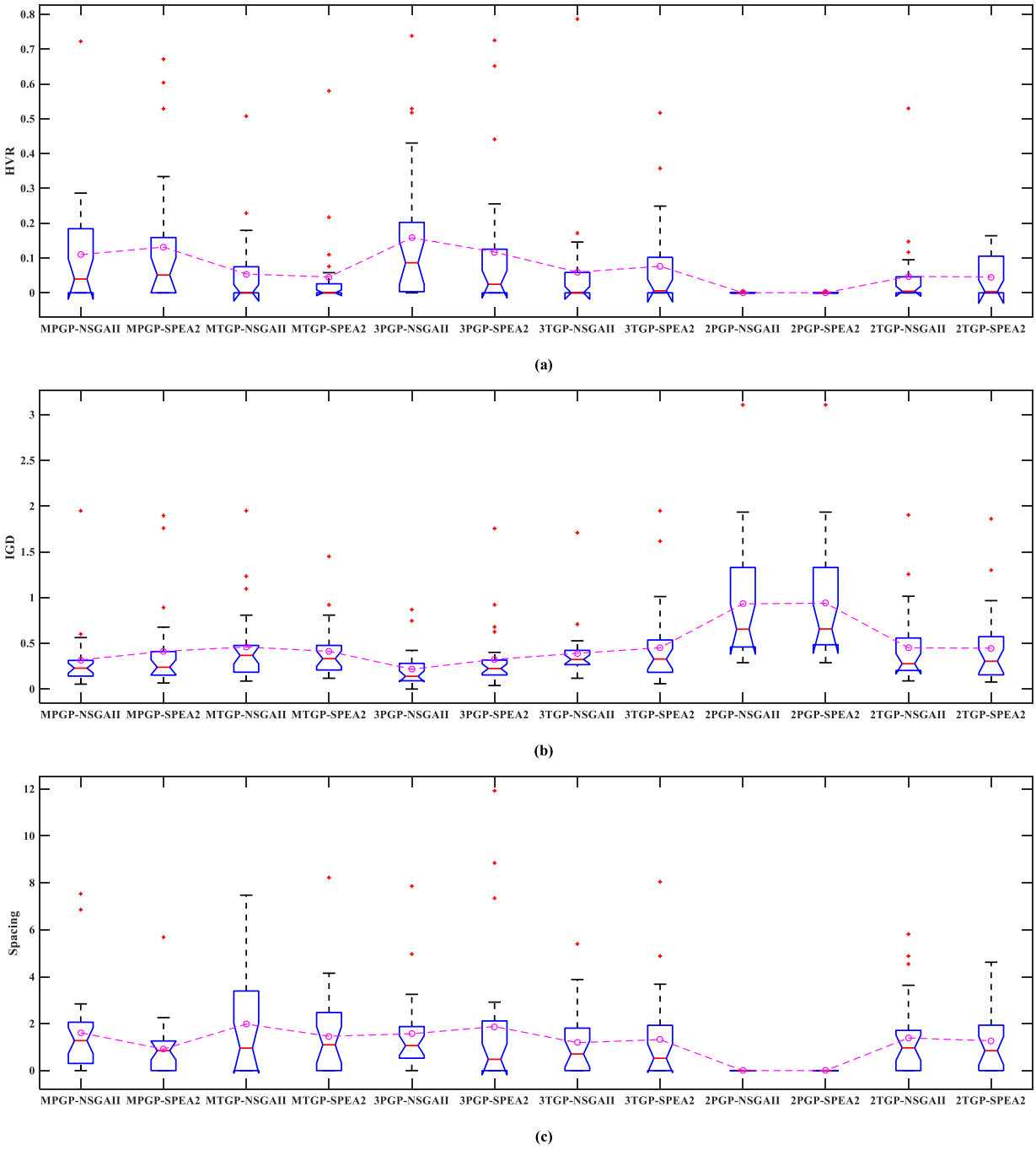


FIGURE 15. Performances of the proposed methods on the test scenarios. (a) HVR. (b) IGD. (c) Spacing.

SMPSO are used to search for appropriate priorities of all operation tasks in stage (a) of Figure 6. PSO maintains a population of n particles,each particle contains a D-dimensional position vector $x_i = (x_{i1}, x_{i2}, ..., x_{iD})$ and a D-dimensional velocity vector $v_i = (v_{i1}, v_{i2}, ..., v_{iD})$. In this study, D is equal to the number of operation tasks in the scheduling scheme. In each iteration, each particle adjusts its speed vector according to its own experience p_{id} and the swarm experience p_{gd}, so as to change their positions and search the D-dimensional space. During the search, each particle updates its velocity and position by the following equations.

$$v_{id}(k+1) = w \cdot v_{id}(k) + c_1 r_1 (p_{id} - x_{id}(k)) + c_2 r_2 (p_{gd} - x_{id}(k)) \quad (12)$$

$$x_{id}(k+1) = x_{id}(k) + v_{id}(k+1) \quad (13)$$

In the above formulas, w is the inertia weight of the particle, r_1 and r_2 are two uniformly distributed random numbers in

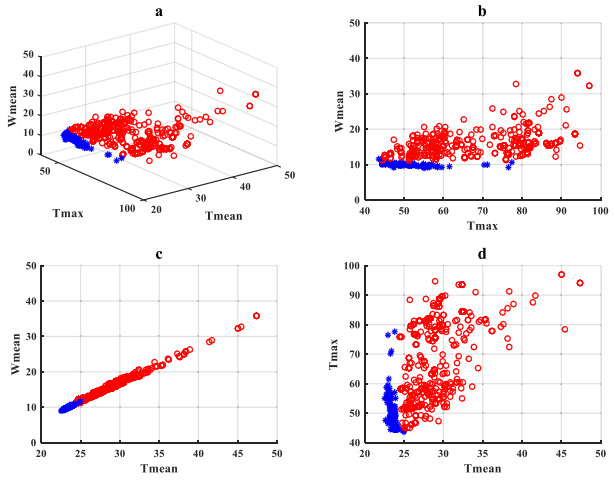


FIGURE 16. The results of the evolved SPs and the existing SPs on the real case. ('∗' and 'o' respectively represent the evolved and existing SPs).

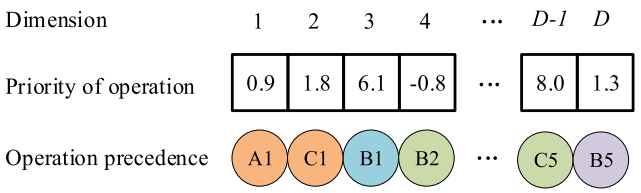


FIGURE 17. Encoding particle scheme.

the range [0, 1], c_1 and c_2 are specific parameters which control the effect of the personal and global best particles. Instead of using upper and lower bound which limit the step size of the velocity in OMOPSO, SMPSO adopt a constriction coefficient x to control the particle's velocity.

$$x = \frac{2}{2 - \varphi - \sqrt{\varphi^2 - 4\varphi}} \tag{14}$$

where φ=$c_1 + c_2$, if$c_1 + c_2 > 4$; φ= 1, if$c_1 + c_2 \leq 4$.

And the accumulated velocity of each variable j (in each particle) is further bounded by means of the following velocity constriction equation.

$$v_{ij}(k) = \begin{cases} delta_j & if\ v_{ij}(k) > delta_j \\ -delta_j & if\ v_{ij}(k) \leq delta_j \\ v_{ij}(k) & \text{otherwise} \end{cases} \tag{15}$$

where $delta_j = (upper\_bound - lower\_bound)/2$. In this study, we limit each variable in each particle to the range $[-100, 100]$. Summarizing the procedure, the velocity of the particle is calculated according to equation (12); the resulting velocity is then multiplied by the constriction factor in equation (14) and the resulting value is constrained by using equation (15).

Figure 17 shows an exemplary encoded particle for the operation precedence based on topological sorting in Figure 5. Each particle contains D variables, and D is the number of operation tasks in the scheduling scheme. Each variable in each particle represents a corresponding priority

TABLE 9. Parameter settings of the MOPSO algorithms.

Algorithm	OMOPSO	SMPSO
Population Size	100	
Archive Size	100	
NFE	10000	
Range	[-100.0, 100]	
r_1, r_2	rand (0, 1)	
w	rand (0.1, 0.5)	
c_1, c_2	rand (1.5,2.0)	rand (1.5,2.5)
Mutation	uniform (30%) +non-uniform (30%) + no mutation (40%)	polynomial mutation (15%) + no mutation (85%)

TABLE 10. p-values of the statistical test for each metric on real case.

Comparison	HVR	IGD	Spacing
MAHH vs. OMOPSO	**0.192**=	**0.459**=	0.009-
MAHH vs. SMPSO	**0.221**=	**0.798**=	0.013-
OMOPSO vs. SMPSO	0.03-	0.476=	0.367=

of an operation task. When we decoding a particle, it assigns the priorities for all the operation tasks in stage (a) of Figure 6. For simplicity, we adopt WD as the MAR, because WD shows dominate performance as shown in Table 11.

Algorithm 3 Pseudocode of a General MOPSO

```
(1)  initialize Swarm
(2)  initialize Leaders Archive
(3)  generation = 0
(4)  while generation < max generations do
(5)    update Velocity
(6)    update Position
(7)    mutation
(8)    evaluation
(9)    update Local Best
(10)   update Leaders Archive
(11)   generation++
(12) end while
(13) return Leaders Archive
```

The pseudo-code of a general MOPSO is shown in Algorithm 3 [103]. It starts by initializing the swarm (Line 1), which includes the position, velocity, and local best of the particles. The leaders archive is initialized with the non-dominated solutions in the swarm (Line 2). Then, the main loop of the algorithm is executed for a maximum number of generations. The velocities and positions of the particles are updated (Lines 5 and 6), and a mutation operator is applied with a given probability (Line 7). The resulting particles are evaluated (Line 8) and both the particle's local best and the leaders archive are updated. When the maximum generation is reached, the algorithm returns the leaders archive as the approximation pareto set (Line 13). According to our preliminary experiments and Coello's recommendation [102], the parameter settings of OMOPSO and SMPSO are shown in Table 9.

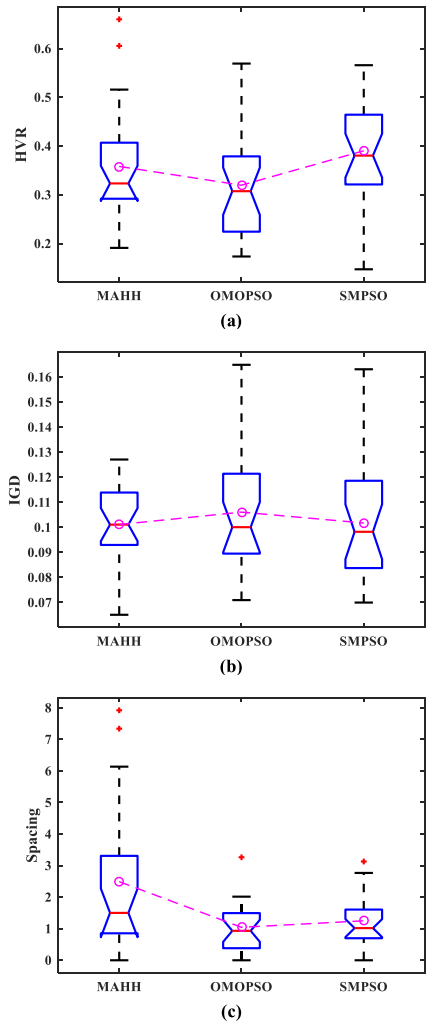


FIGURE 18. Comparison performance of three methods on the real case. (a) HVR. (b) IGD. (c) Spacing.

There are 25 Pareto sets of the evolved SPs generated by the MAHH on the 25 test scenarios in 25 runs. We implemented these 25 heuristic sets on the real case and obtain 25 scheduling result sets. To verify the performance of the evolved SPs produced by the MAHH, two MOPSOs were applied on the same case for 25 independent runs.

Figure 18 illustrates the performances of the evolved SPs and the two MOPSOs on the real case. As shown in Table 10, there is no significant difference between the evolved SPs and the other two MOPSO algorithms in terms of HVR and IGD indicators. However, it should be noted that there is only 852 evolved SPs in 25 Pareto sets produced by the MAHH. It means that the total NFE of 25 runs for the evolved SPs and the OMPSO/SMPSO algorithm is 852 and 25∗10000=250000, respectively. Because most of the running time was consumed in the evaluation process, the NFE also represents the running time. Therefore, the NFE (running time) of the evolved SPs produced by the MAHH amounted to 0.34% of that of OMOPSO/SMPSO when they have the similar performance on HVR and IGD indicators.

In addition, when the scheduling scheme is disrupted in the shop floor, such as the delay arrival of a pre-arranged order, cancellations of already handled jobs and changes in lot size, the meta-heuristics need to start a new iteration in online scheduling. On the contrary, the evolved SPs produced by the MAHH can make a good balance between computation time and quality of solution. This is because the MAHH move the online iteration to the offline training. With the accumulation of knowledge by machine intelligence, similar scheduling results can be obtained by the evolved SPs in much less computing time as compared with the meta-heuristics in online scheduling.

C. INSIGHTS INTO THE EVOLVED SPs

As shown in Table 11, the Pareto solution set of the man-made SPs is obtained by using the non-dominated sorting method. Since many SPs have been obtained from the training experiments, three examples of the non-dominated SPs generated by the three MAHH methods are presented in Table 12. Combined with the two tables, we can observe that the evolved SPs are more complex than the man-made SPs. This is caused by the characteristics of the GP algorithm. However, the evolved SPs can explore the problem characteristics and organize the features together to form a composite SP to defeat the man-made SP in unseen scenario without deteriorating the online computation time. Furthermore, the best values of the three objectives (T_{mean},T_{max} and W_{mean}) in Table 11 are 24.491, 44.49, 11.059, respectively. However, all these values are dominated by the evolve SP#3 in Table 12 (T_{mean} = 23.649, T_{max} =44.46, W_{mean} =10.228). These results validate the effectiveness and the efficiency of the proposed methods.

As shown in Table 12, the evolved SP#1 is generated by the heterogeneous model, where OP-0 is the JSR for the bottleneck resource, and OP-1 to OP-5 represent the JSR of the other five types of non-bottleneck resources (including a cooperative resource). It can be seen that the JSR of the bottleneck agent is more complex than other JSRs of the non-bottleneck agent. The same performance can be seen from the evolved SP#2 that OP-B (JSR of the bottleneck) is more complex than OP-NB (JSR of the non-bottleneck), indicating that the dispatching problem of bottleneck resource is more complicated and requires more features to generate the JSR for the bottleneck resource. The evolved SP#3 is generated by the homogeneous model. We can see that the JSR in this model is more complicated than the JSR in the other two agent models. This is because there is only one agent used to evolve JSR of all the recourses in the problem, resulting in too many feature combinations. Therefore, the bottleneck agent model can balance in both of the generalization performance and solution quality. And it is more convenient to apply in practice than the heterogeneous agent model. Meanwhile, compared with the homogeneous agent model, the solution quality of the bottleneck agent model is more accurate without increasing the complexity of the evolved SPs. This observation also

TABLE 11. Examples of the benchmark SPs based on the bottleneck agent model.

No.	JSR-B	JSR-NB	MAR	T_{mean}	T_{max}	W_{mean}
Man-made SP #1	ATC(7.0)	ATC(7.0)	WD	**24.491**	75.89	11.309
Man-made SP #2	LPT	CR+SPT	WU	24.579	58.2	11.435
Man-made SP #3	ODD	ODD	WD	24.641	51.32	**11.059**
Man-made SP #4	SLK	EDD	WD	24.806	46.59	11.841
Man-made SP #5	LRPT	ODD	WD	24.929	**44.49**	11.263

TABLE 12. Examples of the evolved SPs.

Evolved SP #1 (T_{mean} = 22.549, T_{max} =55.04, W_{mean} =9.016)	
OP-0	SF - (((0.86 * TOD - TDD * W) * 0.624 * TOD* TOD / (SF * SF)) * (TOD / SF * 0.665 / min(SF, 0.044 * TDD)) * (TOD / 0.312 - max(TOD, TDD)))
OP-1	TOD* TDD
OP-2	min(SF, max(min(W, TOD), SF * SF))
OP-3	(0.109 + 2SF - 2PT) / (1.4SF - 1.4TDD- 1.8PT)
OP-4	max(SF + TOD, max(PT, TOD))
OP-5	(((((TOD * TDD / max(0.105, SF)) / max(TOD + TDD, max(0.105, SF))) + TOD) / max(0.105, SF)) + TOD) * (TDD + W)
WS	max(max(min(min(0.507 + WU, min(0.142, WD) / WD), (2WD - 0.818) / (WD / WPC)), max(max(min(WD, WPC), WD + max(WD, 0.874)), 0)), max(min(WD, WPC) / max(WW, WU), max(WD - (WPC / (WPC * WD)), min(0.142, WD))) + ((WD / WPC) / (max(WD, WW) * (WW * WPC))))
Evolved SP #2 (T_{mean} = 22.807, T_{max} =47.49, W_{mean} =9.265)	
OP-B	max((PT * 0.672) + (PT - 0.999), min((min(PT div 0.175, W - TOD) + min(0.843, 0.152 - PT)) * max(W + (PT - 0.999), min(PT, TDD) + SF), W - TOD) + (W - 0.598))
OP-NB	(((W - TOD - PT) / (0.409 - PT)) * SF / TDD) + TOD + PT
WS	min(WU, WW) + WD
Evolved SP #3 (T_{mean} = **23.649**, T_{max} =**44.46**, W_{mean} =**10.228**)	
OP	(max((TOD - SF) / SF, SF * (TOD - (TOD / SF))) * max((TOD* TOD), TOD)) * (SF * (TOD - (SF * (TOD - 2SF)))) * (max(TOD, TOD / SF) * max((TOD / SF) * (TOD / SF), TOD))
WS	2WD + min(WU, 0.454)

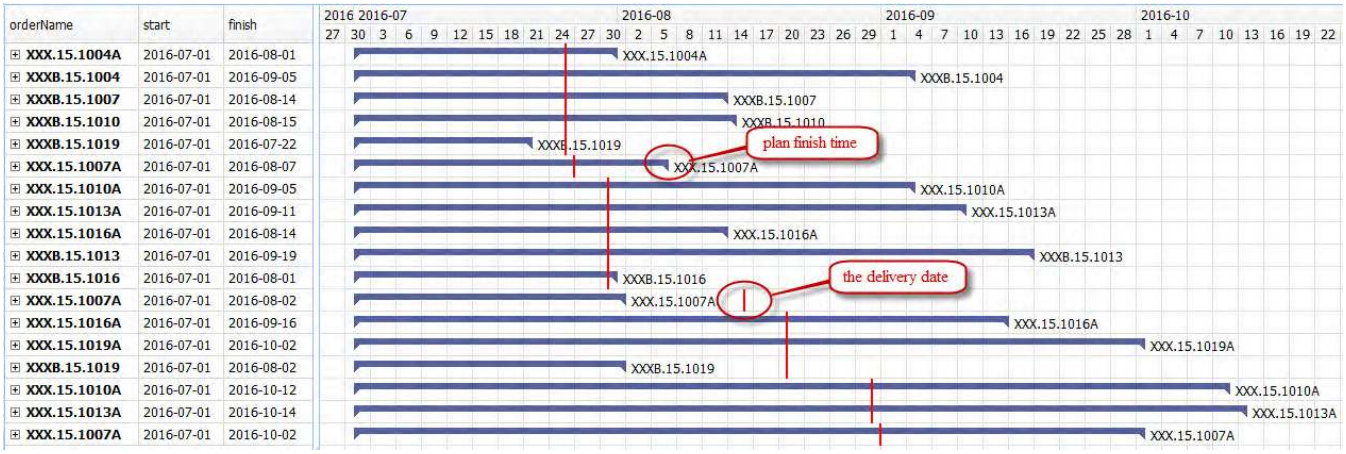


FIGURE 19. Order gantt chart of the solution generated by the evolved SP #2.

shows that the multi-agent model should reflect problem specific characteristic in the structure for better performance.

To visually demonstrate the performance differences between the evolved SPs and the artificial SPs, the man-made SP#3 and the evolved SP#2 are implemented in the MES on the real case for application. Figure 19 shows the order Gantt chart of the solution generated by the evolved SP#2. Figure 20 illustrates the resource Gantt chart generated by the evolved SP#2. The resource Gantt chart produced by the man-made SP#3 is shown in Figure 21.

It can be seen from Figure 21 that the operation tasks are loosely arranged on the FCMC which is the bottleneck resource in this case, and the latest completion time of all tasks on the FCMC is 2016-09-13. As shown in Figure 20, the FCMC workstation has a tight task arrangement, and the latest completion time of all tasks on the FCMC is 2016-09-07 which is about 6 days earlier than that of Figure 21. These detailed results confirm the above conclusions and verify the practical application value of the evolved SPs in industry practice.

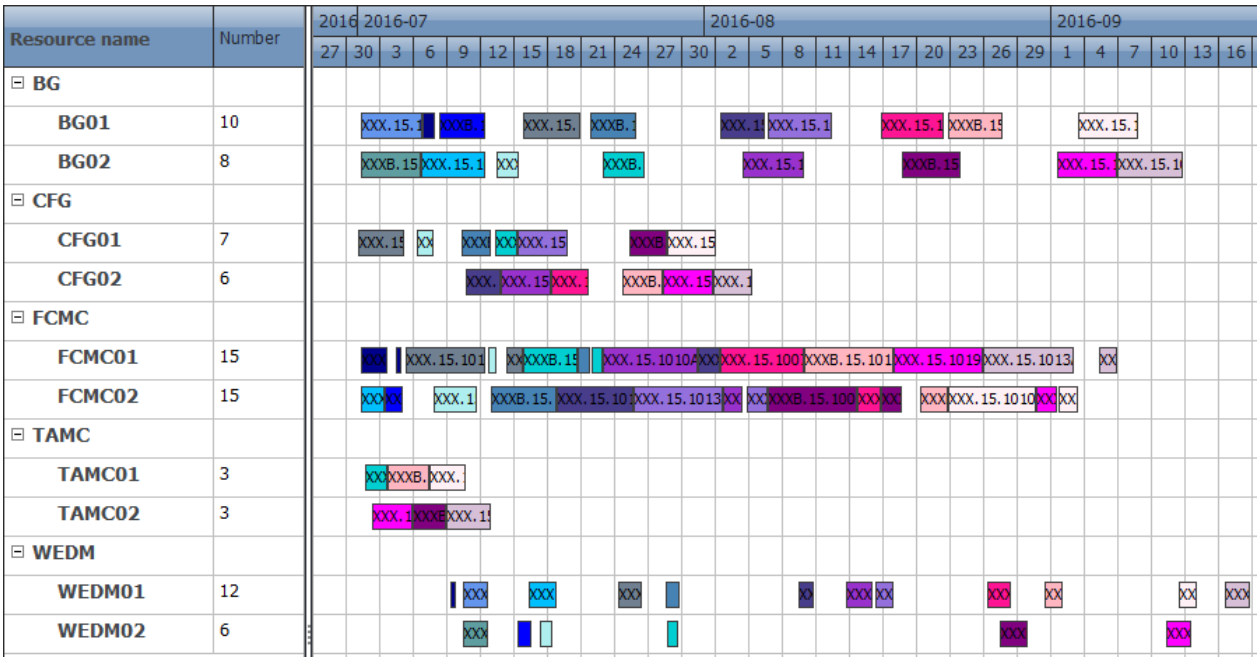


FIGURE 20. Resource Gantt Chart of the solution generated by the Evolved SP #2.

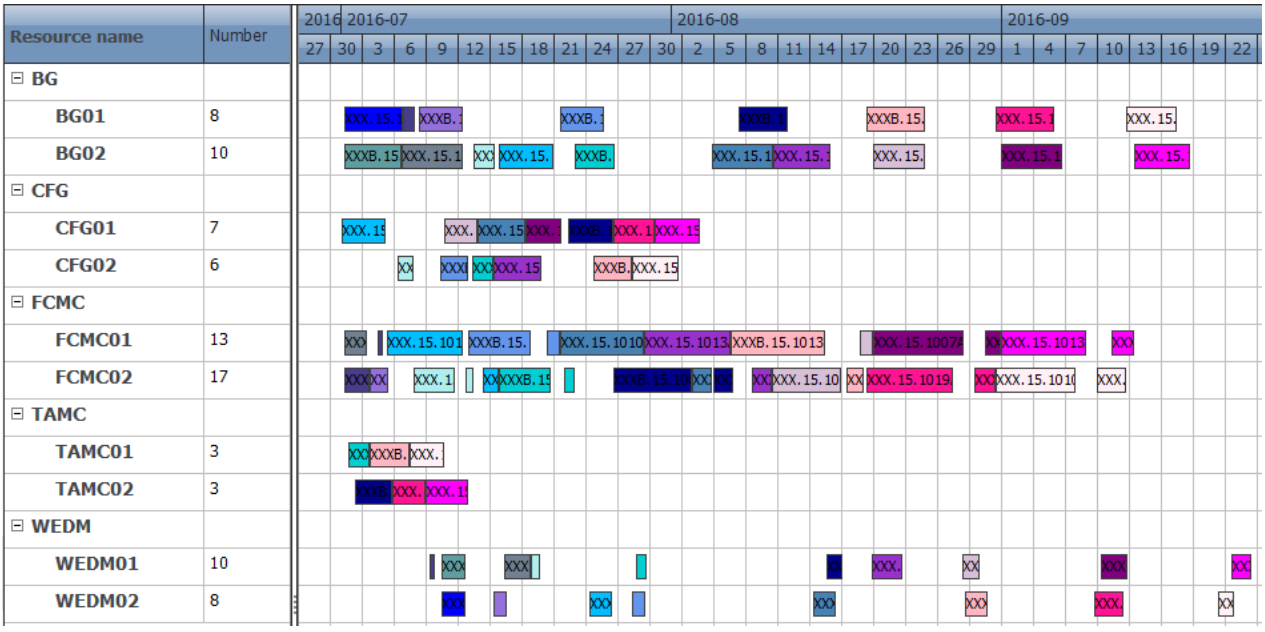


FIGURE 21. Resource gantt chart of the solution generated by the man-made SP #3.

D. PRACTICAL APPLICATION

As shown in Figure 22, the proposed MAHH has been embedded into the MES developed in our laboratory and successfully applied in several manufacturing plants. In these enterprises, industrial tablets which installed with the mobile application of the MES are placed next to each device in the workshop. As shown in the upper left corner of Figure 22, they are called "Device Kanban" which show the operation tasks arranged on the device and can feedback current task status to the MES by the operators. In the daily work, scheduler dispatches tasks to the specific device through the scheduling system in the MES. The device Kanban will display the current task list of the device. When the task is finished, the quantity of the completed task and the progress status will feedback to the MES by the operator. After synchronizing the field information in the MES, the scheduler obtains the latest information about the workshop and performs the new scheduling if needed. In addition, the

FIGURE 22. The Kanban system in the shop floor.

workshop is equipped with four large screens which are shown in the upper right corner of Figure 22 to display the performances of the workshop in real time. It is an intelligent ''Workshop Kanban'' that used to display various instructions and status in the shop floor.

In the past decade, manufacturers have relied heavily on manual feedback in MES. However, with the development of new technologies such as Cyber-physical System (CPS) and Internet of Things (IoT), manufacturers can automatically collect data from multiple data sources through Radio Frequency Identification Devices (RFID), Bar code, Manufacturing Data Collection (MDC) and other data acquisition systems [105]. These technologies can further reduce the workload of workers' feedback tasks and improve work efficiency. After data fusing including filtering, cleaning and processing, the real-time information of the workshop is transmitted into the MES [106]. Then, new scheduling scheme is obtained through the evolutionary scheduling policy trained by the proposed MAHH method. Moreover, the scheduling scheme is revised based on several human-computer interactions according to the schedulers' judgment of the actual situation. As shown in Figure 23, these methods construct a complete closed loop control in the shop floor by connecting the virtual environment with the real world.

The proposed methods are more suitable for short-term scheduling within 1-2 days or even on-line scheduling within 1-2 hours in a discrete manufacturing plant. Since various unpredictable events may happen in the shop floor, the original plan would not be carried out as planned. However, a solution can be achieved by using a sophisticated iterative search algorithm, the cost of computation time will be unacceptable because the dynamic changes require real-time response. If the scheduling scheme does not reflect the current situation in the shop floor, the scheduling result will be meaningless because the original scheduling scheme does not reflect the reality. Compared with the sophisticated algorithm, it is more important to feedback the task progress in time, keep the scheduling scheme synchronized with the task progress in the shop floor, and make scheduling decisions by heuristics in the shortest possible time.

Previous MES implementation experiences and lessons show that it is very important to keep the cyber information of the scheduling system synchronized with the real-time information of the workshop. For example, some enterprises have already implemented MES in the workshop, but workers did not feedback and schedulers did not maintain the scheduling system in time. If things go on like this, the scheduling scheme in the MES would be not consistent with the real information in the shop floor. As a result, this type of

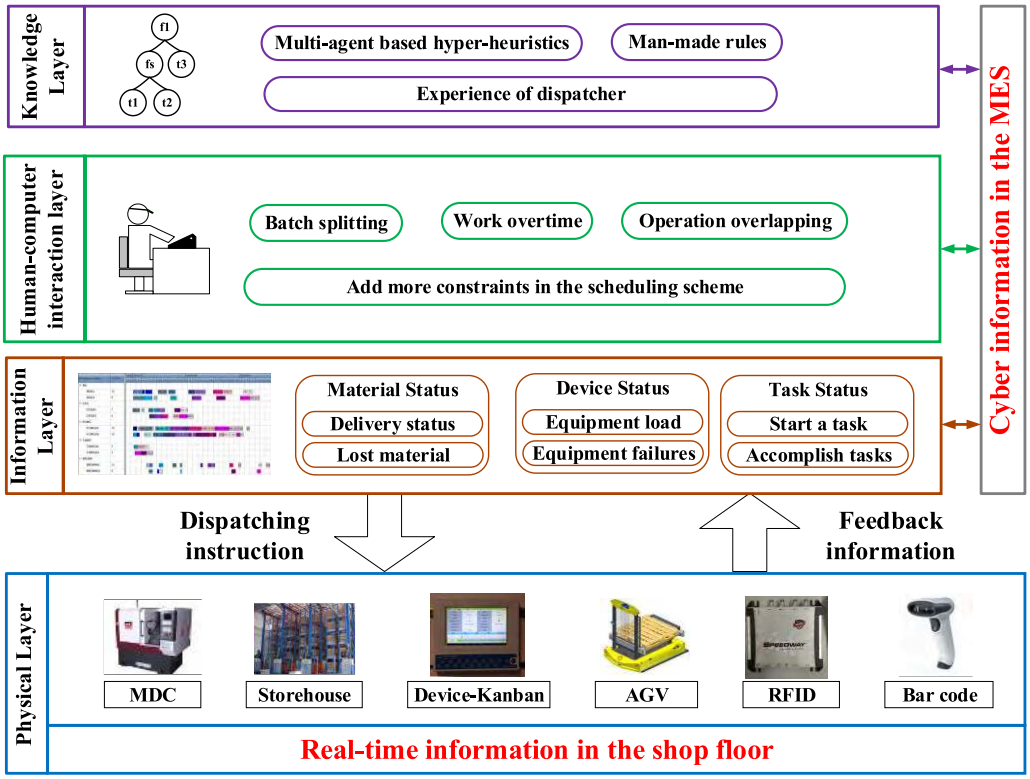


FIGURE 23. A closed loop control of the intelligent shop floor.

scheduling system did not work even with the sophisticated algorithm because the original scheduling scheme did not reflect the reality. Therefore, this type of MES would be abandoned. Furthermore, the manufacturing process is frequently disrupted in the discrete manufacturing plant, so it is more important to obtain an effective solution quickly rather than to spend a lot of time online to get an optimal or near-optimal solution. Therefore, the main contribution of this study is to quickly provide an effective scheduling scheme for schedulers for online scheduling, and the scheduler can synchronize the real information in the shop floor with the cyber information of the scheduling system in real time to form a closed loop control.

VII. CONCLUSIONS AND FUTURE WORKS

This study proposes three types of multi-agent based hyper-heuristics to automatically design scheduling policies for the multi-objective flexible job shop problem. In addition, this study has made a step towards reducing the gap between theoretical progress and industry practice. The main conclusions are as follows.

1) The training results demonstrate that the proposed 3PGP-NSGAII algorithm shows competitive performance with other evolutionary approaches in solving the MO-FJSP. The results show that the bottleneck agent model is more favourable than the other two agent models to solve the scheduling problem with the prior knowledge of the bottleneck resource. The test results are also consistent with the training results, which verify the generalization performance of the bottleneck agent model. Therefore, the bottleneck agent model succeeded to make a good trade-off between the solution quality and the generalization performance among the three agent models.
2) Besides, as the number of agents increases, the average length of JSRs evolved by each algorithm decreases significantly. However, too many agents will lead to the reduction of generalization performance, and it is not convenient for practical application. It can be seen from the representation of the evolved SPs that the length of the JSR of bottleneck resource is higher than that of non-bottleneck resource, indicating that the job sequencing problem of bottleneck resource is more complicated and requires more features to generate the JSR.
3) Furthermore, the evolved SPs produced by the MAHH are compared with the 1536 combinations of benchmark SPs and two MOPSO algorithms on the real case. The results reveal that the evolved SPs dominate nearly all the man-made SPs under all objectives, and the evolved SPs take very little computing time to achieve the equivalent performance similar to the two MOPSOs in online scheduling. In addition, the selected man-made SPs and the evolved SPs are implemented on the real case for fast application. The detailed results confirm the above conclusions and demonstrate the application value of the evolved SPs in industry practice.

4) The proposed MAHH method has been embedded into the MES developed in our laboratory, and it has two application modes in practice. The first application mode is to obtain effective solution quickly in online scheduling. The advantage of this method is that the MAHH move the online iteration to the offline training. Besides, schedulers can add various constraints to the scheduling scheme and carry out experiments quickly to verify their scheduling plans based on the evolved SPs. After several human-computer interactions, a scheduling scheme is constructed for fast application. In the second application mode, the solution generated by the evolved SPs can be used as the initial solution of the meta-heuristic algorithm in a relatively stable production environment. In this way, the solution quality can be improved and the number of iterative searches can be reduced.

Our future works will mainly focus on two aspects, the first of which is to continue improving the proposed MAHH method. Because the job sequencing problem of the same resource usually has multiple decision points, but they cannot be changed during the entire scheduling process. Therefore, we plan to use machine learning technology to select different scheduling policies according to the system state, and even to build different scheduling behaviors for the same resource at different decision points.

On the other hand, we will focus on real-time data collection in the shop floor. The current data collection methods include feedback from workers, bar code scanning and so on. However, the real-time performance is still insufficient. In our future works, various data acquisition methods such as MDC (Manufacturing Data Collection and Control) and RFID (Radio Frequency Identification) will be considered to automatically collect the information of machine tools and materials. The manufacturing information will feed back to the MES in real time so as to keep the scheduling system in sync with the field information. In this way, we can respond more quickly and accurately to disturbances in the workshop.

ACKNOWLEDGEMENT

The authors would like to thank Qing-miao Liao and Peng-cheng Fang for their support and contributions during the development of this work.

REFERENCES

[1] L. Wang, M. Törngren, and M. Onori, "Current status and advancement of cyber-physical systems in manufacturing," *J. Manuf. Syst.*, vol. 37, pp. 517–527, Oct. 2015.
[2] 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.
[3] W. Ji and L. H. Wang, "Big data analytics based fault prediction for shop floor scheduling," *J. Manuf. Syst.*, vol. 43, pp. 187–194, Apr. 2017.
[4] J. Wang, Y. Ma, L. Zhang, R. X. Gao, and D. Wu, "Deep learning for smart manufacturing: Methods and applications," *J. Manuf. Syst.*, vol. 48, pp. 144–156, Jul. 2018.
[5] X. Xu, "From cloud computing to cloud manufacturing," *Robot. Comput.-Integr. Manuf.*, vol. 28, pp. 75–86, Feb. 2012.
[6] L. Ren, L. Zhang, L. Wang, F. Tao, and X. D. Chai, "Cloud manufacturing: Key characteristics and applications," *Int. J. Comput. Integr. Manuf.*, vol. 30, no. 6, pp. 501–515, 2017.
[7] F. Tao, Q. Qi, A. Liu, and A. Kusiak, "Data-driven smart manufacturing," *J. Manuf. Syst.*, vol. 48, pp. 157–169, Jul. 2018.
[8] 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.
[9] A. Baykaso lu 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.
[10] A. Baykaso lu and L. Özbakir, "Analyzing the effect of dispatching rules on the scheduling performance through grammar based flexible scheduling system," *Int. J. Prod. Econ.*, vol. 124, pp. 369–381, Apr. 2010.
[11] D. Li, R. Zhan, D. Zheng, M. Li, and I. Kaku, "A hybrid evolutionary hyper-heuristic approach for intercell scheduling considering transportation capacity," *IEEE Trans. Autom. Sci. Eng.*, vol. 13, no. 2, pp. 1072–1089, Apr. 2016.
[12] 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.
[13] B. Çali 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.
[14] V. Kaplano lu *et al.*, "A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles (AGV) in manufacturing systems by considering AGV breakdowns," *Int. J. Eng. Res. Innov.*, vol. 7, no. 2, pp. 32–38, 2015.
[15] 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.
[16] 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.
[17] 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.
[18] 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.
[19] D. Golmohammadi, "A neural network decision-making model for job-shop scheduling," *Int. J. Prod. Res.*, vol. 51, no. 17, pp. 5142–5157, Sep. 2013.
[20] S. Abdullah and M. Abdolrazzagh-Nezhad, "Fuzzy job-shop scheduling problems: A review," *Inf. Sci.*, vol. 278, pp. 380–407, Sep. 2014.
[21] Y. Zhang, J. Wang, and Y. Liu, "Game theory based real-time multi-objective flexible job shop scheduling considering environmental impact," *J. Clean. Prod.*, vol. 167, pp. 665–679, Nov. 2017.
[22] 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.
[23] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *J. Scheduling*, vol. 12, no. 4, pp. 417–431, Aug. 2009.
[24] A. Baykasoglu and L. Gorkemli, "Dynamic virtual cellular manufacturing through agent-based modelling," *Int. J. Comput. Integr. Manuf.*, vol. 30, no. 6, pp. 564–579, 2017.
[25] C. Sahin, M. Demirtas, R. Erol, A. Baykaso lu, and V. Kaplano lu, "A multi-agent based approach to dynamic scheduling with flexible processing capabilities," *J. Intell. Manuf.*, vol. 28, pp. 1827–1845, Dec. 2017.
[26] R. Erol, C. Sahin, A. Baykasoglu, and V. Kaplanoglu, "A multi-agent based approach to dynamic scheduling of machines and automated guided vehicles in manufacturing systems," *Appl. Soft Comput.*, vol. 12, pp. 1720–1732, Jun. 2012.
[27] M. Đurasević and D. Jakobović, "A survey of dispatching rules for the dynamic unrelated machines environment," *Expert Syst. Appl.*, vol. 113, pp. 555–569, Dec. 2018.
[28] D. H. Wolper and W. G. Macready, "No free lunch theorems for optimization," *IEEE Trans. Evol. Comput.*, vol. 1, no. 1, pp. 67–82, Apr. 1997.
[29] K.-C. Ying, S.-W. Lin, and C.-C. Lu, "Effective dynamic dispatching rule and constructive heuristic for solving single-machine scheduling problems with a common due window," *Int. J. Prod. Res.*, vol. 55, no. 6, pp. 1707–1719, 2017.

[30] C. Gahm, J. J. Kanet, and A. Tuma, "On the flexibility of a decision theory-based heuristic for single machine scheduling," *Comput. Oper. Res.*, vol. 101, pp. 103–115, Jan. 2019.

[31] I. Pergher and A. T. de Almeida, "A multi-attribute, rank-dependent utility model for selecting dispatching rules," *J. Manuf. Syst.*, vol. 46, pp. 264–271, Jan. 2018.

[32] G. R. Amin and A. El-Bouri, "A minimax linear programming model for dispatching rule selection," *Comput. Ind. Eng.*, vol. 121, pp. 27–35, Jul. 2018.

[33] S.-W. Lin, Z.-J. Lee, K.-C. Ying, and R.-H. Lin, "Meta-heuristic algorithms for wafer sorting scheduling problems," *J. Oper. Res. Soc.*, vol. 62, pp. 165–174, Jan. 2011.

[34] S. Luke, Lulu. (2013). *Essentials of Metaheuristics*. [Online]. Available: http://cs.gmu.edu/~sean/book/metaheuristics

[35] A. Baykasoglu, "Linguistic-based meta-heuristic optimization model for flexible job shop scheduling," *Int. J. Prod. Res.*, vol. 40, pp. 4523–4543, Nov. 2002.

[36] A. Baykaso lu, L. Özbakir, and A. I. Sönmez, "Using multiple objective tabu search and grammars to model and solve multi-objective flexible job shop scheduling problems," *J. Intell. Manuf.*, vol. 15, pp. 777–785, Dec. 2004.

[37] S. Jia and Z.-H. Hu, "Path-relinking Tabu search for the multi-objective flexible job shop scheduling problem," *Comput. Oper. Res.*, vol. 47, pp. 11–26, Jul. 2014.

[38] J.-Q. Li, P. Duan, J. Cao, X.-P. Lin, and Y.-Y. Han, "A hybrid Pareto-based tabu search for the distributed flexible job shop scheduling problem with E/T criteria," *IEEE Access*, vol. 6, pp. 58883–58897, 2018.

[39] A. Baykaso lu and F. S. Karaslan, "Solving comprehensive dynamic job shop scheduling problem by using a GRASP-based approach," *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3308–3325, 2017.

[40] 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, Oct. 2014.

[41] L. Lin and M. Gen, "Hybrid evolutionary optimisation with learning for production scheduling: State-of-the-art survey on algorithms and applications," *Int. J. Prod. Res.*, vol. 56, pp. 193–223, Feb. 2018.

[42] L. Wang, S. Wang, and M. Liu, "A Pareto-based estimation of distribution algorithm for the multi-objective flexible job-shop scheduling problem," *Int. J. Prod. Res.*, vol. 51, pp. 3574–3592, Jun. 2013.

[43] M. B. S. S. Reddy, C. Ratnam, G. Rajyalakshmi, and V. K. Manupati, "An effective hybrid multi objective evolutionary algorithm for solving real time event in flexible job shop scheduling problem," *Measurement*, vol. 114, pp. 78–90, Jan. 2018.

[44] X.-N. Shen and X. Yao, "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems," *Inf. Sci.*, vol. 298, pp. 198–224, Mar. 2015.

[45] X.-N. Shen, Y. Han, and J.-Z. Fu, "Robustness measures and robust scheduling for multi-objective stochastic flexible job shop scheduling problems," *Soft Comput.*, vol. 21, pp. 6531–6554, Nov. 2017.

[46] L. Zhou, Z. Chen, and S. Chen, "An effective detailed operation scheduling in MES based on hybrid genetic algorithm," *J. Intell. Manuf.*, vol. 29, pp. 135–153, Jan. 2018.

[47] 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.

[48] G. Gong, Q. Deng, X. Gong, W. Liu, and Q. Ren, "A new double flexible job-shop scheduling problem integrating processing time, green production, and human factor indicators," *J. Cleaner Prod.*, vol. 174, pp. 560–576, Feb. 2018.

[49] 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.

[50] R. Wu, Y. B. Li, S. Guo, and X. Li, "An efficient meta-heuristic for multi-objective flexible job shop inverse scheduling problem," *IEEE Access*, vol. 6, pp. 59515–59527, 2018.

[51] B. Zhao, J. Gao, K. Chen, and K. Guo, "Two-generation Pareto ant colony algorithm for multi-objective job shop scheduling problem with alternative process plans and unrelated parallel machines," *J. Intell. Manf.*, vol. 29, pp. 93–108, Jan. 2018.

[52] 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.

[53] T. Meng, Q.-K. Pan, and H.-Y. Sang, "A hybrid artificial bee colony algorithm for a flexible job shop scheduling problem with overlapping in operations," *Int. J. Prod. Res.*, vol. 56, no. 16, pp. 5278–5292, 2018.

[54] T. Jiang and C. Zhang, "Application of grey wolf optimization for solving combinatorial problems: Job shop and flexible job shop scheduling cases," *IEEE Access*, vol. 6, pp. 26231–26240, 2018.

[55] K. Z. Gao, P. N. Suganthan, Q. K. Pan, T. J. Chua, T. X. Cai, and C. S. Chong, "Discrete harmony search algorithm for flexible job shop scheduling problem with multiple objectives," *J. Intell. Manf.*, vol. 27, pp. 363–0374, Apr. 2016.

[56] J. Li, Q. Pan, and S. Xie, "An effective shuffled frog-leaping algorithm for multi-objective flexible job shop scheduling problems," *Appl. Math. Comput.*, vol. 218, pp. 9353–9371, May 2012.

[57] S. Karthikeyan, P. Asokan, S. Nickolas, and T. Page, "A hybrid discrete firefly algorithm for solving multi-objective flexible job shop scheduling problems," *Int. J. Bio-Inspired Comput.*, vol. 7, pp. 386–401, Nov. 2015.

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

[59] 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. Manf.*, vol. 29, pp. 603–615, Mar. 2018.

[60] G. Zhang, X. Shao, P. Li, and L. Gao, "An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem," *Comput. Ind. Eng.*, vol. 56, pp. 1309–1318, May 2009.

[61] M. R. Singh, M. Singh, S. S. Mahapatra, and N. Jagadev, "Particle swarm optimization algorithm embedded with maximum deviation theory for solving multi-objective flexible job shop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 85, pp. 2353–2366, Aug. 2016.

[62] S. Zhang and T. N. Wong, "Flexible job-shop scheduling/rescheduling in dynamic environment: a hybrid MAS/ACO approach," *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3173–3196, 2017.

[63] I. Paprocka and B. Skołud, "A hybrid multi-objective immune algorithm for predictive and reactive scheduling," *J. Scheduling*, vol. 20, pp. 165–182, Apr. 2017.

[64] C. Lu, X. Li, L. Gao, W. Liao, and J. Yi, "An effective multi-objective discrete virus optimization algorithm for flexible job-shop scheduling problem with controllable processing times," *Comput. Ind. Eng.*, vol. 104, pp. 156–174, Feb. 2017.

[65] 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.

[66] K. Z. Gao, P. N. Suganthan, Q. K. Pan, T. J. Chua, C. S. Chong, and T. X. Cai, "An improved artificial bee colony algorithm for flexible job-shop scheduling problem with fuzzy processing time," *Expert Syst. Appl.*, vol. 65, pp. 52–67, Dec. 2016.

[67] C. Lu, L. Gao, X. Li, and S. Xiao, "A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry," *Eng. Appl. Artif. Intel.*, vol. 57, pp. 61–79, Jan. 2017.

[68] K.-Z. Gao, P. N. Suganthan, Q.-K. Pan, T. J. Chua, T. X. Cai, and C.-S. Chong, "Pareto-based grouping discrete harmony search algorithm for multi-objective flexible job shop scheduling," *Inf. Sci.*, vol. 289, pp. 76–90, Dec. 2014.

[69] D. Lei, Y. Zheng, and X. Guo, "A shuffled frog-leaping algorithm for flexible job shop scheduling with the consideration of energy consumption," *Int. J. Prod. Res.*, vol. 55, no. 11, pp. 3126–3140, 2017.

[70] S. Karthikeyan, P. Asokan, and S. Nickolas, "A hybrid discrete firefly algorithm for multi-objective flexible job shop scheduling problem with limited resource constraints," *Int. J. Adv. Manuf. Technol.*, vol. 72, nos. 9–12, pp. 1567–1579, Jun. 2014.

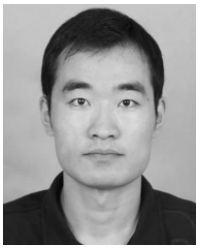
[71] Q. Liu, M. Zhan, F. O. Chekem, X. Shao, B. Ying, and J. W. Sutherland, "A hybrid fruit fly algorithm for solving flexible job-shop scheduling to reduce manufacturing carbon footprint," *J. Cleaner Prod.*, vol. 168, pp. 668–678, Dec. 2017.

[72] 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.

[73] 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.

[74] 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.

[75] 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.
[76] 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.
[77] J. Huang and G. A. Süer, "A dispatching rule-based genetic algorithm for multi-objective job shop scheduling using fuzzy satisfaction levels," *Comput. Ind. Eng.*, vol. 86, pp. 29–42, Aug. 2015.
[78] R. Zhang, S. Song, and C. Wu, "A dispatching rule-based hybrid genetic algorithm focusing on non-delay schedules for the job shop scheduling problem," *Int. J. Adv. Manuf. Technol.*, vol. 67, pp. 5–17, Jul. 2013.
[79] J. A. Vázquez-Rodríguez and S. Petrovic, "A new dispatching rule based genetic algorithm for the multi-objective job shop problem," *J. Heuristics*, vol. 16, no. 6, pp. 771–793, Dec. 2010.
[80] D. Li, M. Li, X. Meng, and Y. Tian, "A hyperheuristic approach for intercell scheduling with single processing machines and batch processing machines," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 45, no. 2, pp. 315–325, Feb. 2015.
[81] 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.
[82] 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.
[83] 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.
[84] 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.
[85] 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.
[86] K. Miyashita, "Job-shop scheduling with genetic programming," in *Proc. GECCO*, Las Vegas, NV, USA, 2000, pp. 505–512.
[87] C. D. Geiger, R. Uzsoy, and H. Aytu , "Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach," *J. Scheduling*, vol. 9, no. 1, pp. 7–34, Feb. 2006.
[88] 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.
[89] R. Hunt, M. Johnston, and M. Zhang, "Evolving machine-specific dispatching rules for a two-machine job shop using genetic programming," in *Proc. IEEE CEC*, Beijing, China, Jul. 2014, pp. 618–625.
[90] J. Zhang, J. Yang, and Y. Zhou, "Robust scheduling for multi-objective flexible job-shop problems with flexible workdays," *Eng. Optimiz.*, vol. 48, no. 11, pp. 1973–1989, 2016.
[91] J. Zhang and J. Yang, "Flexible job-shop scheduling with flexible workdays, preemption, overlapping in operations and satisfaction criteria: An industrial application," *Int. J. Prod. Res.*, vol. 54, no. 16, pp. 4894–4918, 2016.
[92] P. Brandimarte, "Routing and scheduling in a flexible job shop by tabu search," *Ann. Oper. Res.*, vol. 41, no. 3, pp. 157–183, Sep. 1993.
[93] 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.
[94] Y. Zhou, J.-J. Yang, and L.-Y. Zheng, "Hyper-heuristic coevolution of machine assignment and job sequencing rules for multi-objective dynamic flexible job shop scheduling," *IEEE Access*, vol. 7, pp. 68–88, 2019.
[95] 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.
[96] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm for multi-objective optimization," ETH, Zürich, Switzerland, Tech. Rep. TIK-report 103, May 2001. doi: 10.3929/ethz-a-004284029.
[97] 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.
[98] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.
[99] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm test suites," in *Proc. Symp. Appl. Comput.*, San Antonio, TX, USA, 1999, pp. 351–357.
[100] 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.
[101] 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.
[102] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 256–279, Jun. 2004.
[103] M. R. Sierra and C. A. C. Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and ϵ-dominance," in *Proc. EMO*, Guanajuato, Mexico, 2005, pp. 505–519.
[104] A. J. Nebro, J. J. Durillo, J. Garcia-Nieto, C. A. C. Coello, F. Luna, and E. Alba, "SMPSO: A new PSO-based metaheuristic for multi-objective optimization," in *Proc. MCDM*, Nashville, TN, USA, Apr. 2009, pp. 66–73.
[105] Y. Wang, L. Zheng, Y. Hu, and W. Fan, "Multi-source heterogeneous data collection and fusion for manufacturing workshop based on complex event processing," presented at the 48th Int. Conf. Comput. Ind. Eng. (CIE), Auckland, New Zealand, Dec. 2018.
[106] P. Fang, Y. Jiang, and R. Zhong, "Real-time monitoring of workshop status based on internet of things," presented at the 48th Int. Conf. Comput. Ind. Eng. (CIE), Auckland, New Zealand, Dec. 2018.



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 systems.

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 has served as the Vice President of the College, and also the Director of the Department of Industrial and Manufacturing Systems Engineering, Beihang University, Beijing, China. He has authored one book and over 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 also 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 simulations.