Multi-objective Dynamic Flexible Job Shop Scheduling with Biased Objectives via Multitask Genetic Programming

Fangfang Zhang, *Member, IEEE,* Gaofeng Shi, Yi Mei, *Senior Member, IEEE,* and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—Dynamic flexible job shop scheduling is an important combinatorial optimisation problem which has rich real-world applications such as product processing in manufacturing. Genetic programming has been successfully used to learn scheduling heuristics for dynamic flexible job shop scheduling. Intuitively, users prefer small and effective scheduling heuristics that can not only generate promising schedules but also are computationally efficient and easy to be understood. However, a scheduling heuristic with better effectiveness tends to have a larger size, and the effectiveness of rules and rule size are potentially conflicting objectives. With the traditional dominance relation based multi-objective algorithms, there is a search bias towards rule size, since rule size is much easier to be optimised than effectiveness and larger rules are easily abandoned, resulting in the loss of effectiveness. To address this issue, this paper develops a novel multi-objective genetic programming algorithm that takes size and effectiveness of scheduling heuristics for optimisation via multitask learning mechanism. Specifically, we construct two tasks for the multi-objective optimisation with biased objectives using different search mechanisms for each task. The focus of the proposed algorithm is to improve the effectiveness of learned small rules by knowledge sharing between constructed tasks which is implemented with the crossover operator. The results show that our proposed algorithm performs significantly better, i.e., with smaller and more effective scheduling heuristics, than the state-of-the-art algorithms in the examined scenarios. By analysing the population diversity, we find that the proposed algorithm has a good balance between exploration and exploitation during the evolutionary process.**

***Impact Statement*—Genetic programming is a popular approach to learning scheduling heuristics for scheduling problems. However, traditional dominace-relation based multi-objective genetic programming algorithms are limited to biasing easily to small and ineffective scheduling heuristics. This paper presents a ground-breaking approach to multi-objective dynamic flexible job shop scheduling by integrating rule size considerations via multitask learning. The effectiveness of the proposed algorithm is realised by knowledge sharing among rules of similar size. Through extensive experimental validation, the effectiveness of the approach is empirically demonstrated. The research contributes methodologically by introducing a novel framework for integrating rule size considerations via multitask learning with knowledge sharing into scheduling algorithms, paving the way for further advancements in optimisation techniques. The paper could offer diverse interpretable solutions to enhance productivity and reduce costs in various industries.**

***Index Terms*—Multi-objective Dynamic Flexible Job Shop Scheduling, Scheduling Heuristics, Multi-objective with Biased Objectives, Genetic Programming, Multitask Learning.**

I. Introduction

Job shop scheduling is an important combinatorial optimisation problem, which aims to optimise machine resources to process a set of jobs that consists of a number of operations [1]. In traditional job shop scheduling, each operation can be processed by a predefined machine. Flexible job shop scheduling is an extension of traditional job shop scheduling, where each operation can be processed on more than one machine [2]. Thus, we need to make *machine assignment* decision to allocate jobs/operations to machines, and *operation sequencing* decision to choose which operation will be processed next when a machine becomes idle and there are operations in its queue, simultaneously. Dynamic flexible job shop scheduling (DFJSS) [3] makes machine assignment and operation sequencing decisions under dynamic environments such as job arrivals over time and machine breakdown. DFJSS has many practical applications such as order picking in warehouse [4] and production process in manufacturing [5] which can contribute to great economic benefits. In such domains, *learning small/simple and effective* scheduling heuristics becomes more important, since people including production management and floor shop operators prefer to understand the learned rules rather than bearing the risk of loss [6].

According to the ways of finding solutions for job shop scheduling, the methods can be classified into *solution optimisation methods* and *hyper-heuristic methods*. Solution optimisation methods such as exact methods including linear programming [7] and traditional meta-heuristic methods including genetic algorithms [8], aim to optimise solutions (i.e., schedules in DFJSS) for a problem directly. However, exact methods are normally used for static and small scale problems due to their high computational complexity. Meta-heuristic methods can handle large scale problems well, however, traditional meta-heuristic methods are not efficient in dynamic problems since they face with rescheduling issues of dynamic events. Instead of learning solutions directly, scheduling heuristics, e.g., shortest processing time as priority rule [9], have been widely

Manuscript received XXX; revised XXX and XXX; accepted XXX. This work is partly supported by the MBIE Endeavour Smart Idea Fund RTVU2305. (*Corresponding author: Fangfang Zhang*)

Fangfang Zhang, Yi Mei, and Mengjie Zhang are with the Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: fangfang.zhang@ecs.vuw.ac.nz, yi.mei@ecs.vuw.ac.nz, mengjie.zhang@ecs.vuw.ac.nz). Gaofeng Shi is with Emerson in New Zealand (e-mail: xjsgf2010@gmail.com).

Colour versions of one or more of the figures in this article are available online at XXX.

Digital Object Identifier XXX

used to handle DFJSS by prioritising machines or operations due to their efficiency for handling DFJSS. However, it is time-consuming to manually design such rules that can be effective for a variety of scenarios. Hyper-heuristic approach targets on finding scheduling heuristics in heuristic search space. Genetic programming (GP), as a hyper-heuristic approach [10], [11], [12], [13], [14], has been successfully used to learn scheduling heuristics in DFJSS.

In general, a rule with smaller number of nodes is more likely to be understand easily and be accepted for real-world applications [15]. In addition, compared with large rules, small rules are computationally cheaper which can make real-time decisions more efficient. However, existing studies of GP in DFJSS mainly focus only on the effectiveness of scheduling heuristics [16], [17], [18]. Effective scheduling heuristics are more likely to have larger sizes [19], [20]. The objectives of effectiveness and rule size are potentially conflicting, which makes the optimisation of effectiveness and rule size a multi-objective optimisation problem. Dominance relation based multi-objective optimisation algorithms, such as non-dominated sorting genetic algorithm II (NSGA-II) [21], have been popularly used for multi-objective optimisation problems. However, taking the effectiveness (e.g., minimisation of max-flowtime) and rule size of scheduling as objectives with traditional non-dominated sorting has a bias issue to rule size, and the algorithm can only evolve small but ineffective scheduling heuristics. The reason is that the rule size is much easier to optimise than effectiveness, thus only small rules will survive during the evolutionary process. In this paper, we call a multi-objective problem in which an objective is easier to be optimised than other objectives a *multi-objective problem with biased objectives*.

The existing studies considering size and effectiveness of scheduling heuristics with GP in DFJSS are limited. However, there are some related studies from a similar dynamic combinatorial optimisation problem, i.e., arc routing. To handle the search bias issue in multi-objective optimisation with biased objectives, [22] investigated a two-stage non-dominated sorting GP (NSGP) to evolve routing policies for arc routing problems. Specifically, in the first stage, only the effectiveness of rules is considered, and a single objective optimisation problem is formed. At the end of the first stage, the population can contain large and effective rules. In the second stage, NSGP is used, inheriting the first stage's population containing effective (large) rules, and using an archive to save rules in the Pareto front across generations. α-dominance relation based NSGP was proposed for multi-objective GP by giving different selection pressures on biased objectives adaptively to select the parents [23]. The algorithm in [23] was further extended with a new α adaptation scheme and an archive strategy in [24]. The algorithm [24] was adapted into DFJSS but with the Pareto front obtained by traditional non-dominated sorting as a reference to update α and archive [25] rather than α-dominance relation. The results show that the algorithm in [25] performs better than the one in [24]. These studies show the effectiveness of using α-dominance relation for handling the search bias in multi-objective with biased objectives, however, they only focus on the balance of search on rule size and rule effectiveness.

Multitask learning is a type of machine learning approach where models are trained to perform multiple tasks simultaneously. Evolutionary multitask learning aims to solve multiple related tasks simultaneously with evolutionary computation algorithms [26]. This approach can lead to improved learning effectiveness and efficiency for the task-specific models compared to training them separately [27], [28]. Multitask learning has been successfully used to handle multiple tasks by sharing knowledge between tasks [29], [30], [31]. Inspired by multitask learning, the goal of this paper is to design an effective multi-objective algorithm via multitask learning by taking the optimisation with small rules as a separate task. Specifically, we will treat the optimisation with α-dominance for DFJSS as the first task (main task, has search control on size and effectiveness), and the optimisation with traditional dominance as the second task (auxiliary task, will bias to small rules). The proposed algorithm is expected to achieve better performance for DFJSS from the main task by sharing knowledge with the auxiliary task. It is noted that the goal of this paper is not to handle a multitask problem, but to use the idea of multitask learning for a multi-objective task/problem with biased objectives. Specifically, the major contributions of this paper are shown as follows:

1) We have proposed a new knowledge sharing based framework to improve the effectiveness of small rules obtained by α-dominance based multi-objective GP via multitask learning. To answer the question which individuals are small rules for knowledge sharing, we have developed an adaptive rule size range deciding strategy for small rules. In addition, different from the algorithm in [25], this paper has got rid of maintaining archive.
2) We have developed a novel and effective multi-objective GP algorithm via multitask learning with the proposed rule size range deciding strategy. Specifically, we evolve rules in the first subpopulation with the α-dominance multi-objective GP algorithm, and evolve rules in the second subpopulation with the traditional dominance multi-objective algorithm. More importantly, we share knowledge among rules with sizes within the range obtained by the designed rule size range deciding strategy from different subpopulations. The results show that the proposed multi-objective GP algorithm via multitask learning has achieved better performance with smaller and more effective scheduling heuristics compared with the state-of-the-art algorithms.
3) Further analyses show that the proposed algorithm can maintain both the size of the routing rule and sequencing rules' sizes well. We have investigated the effect of the proposed algorithm on population diversity, we find that the proposed algorithm has a higher population diversity in the early stage of the evolutionary process, representing a better exploration ability. In addition, the population diversity stays at a proper level, leading to a good exploitation ability in the late stage of the evolutionary process. We can see that the proposed algorithm has achieved a good balance of exploration

and exploitation.

A. Organisation

The rest of this paper is organised as follows. Section II presents the background of this paper. Detailed descriptions of the proposed algorithm are given in Section III. The experiment design is shown in Section IV. Results and discussions are presented in Section V. Further analyses are conducted in Section VI. Section VII concludes this paper.

II. Background

A. Multi-objective Dynamic Flexible Job Shop Scheduling

DFJSS requires the processing of jobs $\mathcal{J} = \{J_1, J_2, ..., J_n\}$ using a set of machines $\mathcal{M} = \{M_1, M_2, ..., M_m\}$. Each job is comprised of a sequence of operations that must be executed in order, and each operation can be processed on multiple machines [32], indicating the flexibility of machine resources. However, each operation will be executed on one of its feasible machines, and the duration of its processing relies on the machine that handles it. This paper works on the dynamic event of stochastic and dynamic arrival of new jobs [33], [34], as it is the most frequently occurring dynamic event in practical scenarios. The details regarding a new job remain undisclosed until it arrives on the shop floor. The main constraints of DFJSS are shown as follows.

- Each machine can execute a maximum of one operation at a given time.
- Each operation can only be processed by one of its feasible machines at a time.
- The processing of an operation cannot occur until all its preceding operations have been executed.
- Once an operation is commenced, it must be carried out without interruption until its completion.

The multi-objective studied in this paper consists of two objectives, i.e., one is effectiveness related, and the other is rule size (i.e., the number of nodes). Three commonly used effectiveness related objectives are considered to form different scenarios with rule size in this paper, which is shown below:

- Max-flowtime: $max\{C_1 - r_1, C_2 - r_2, ..., C_j - r_j\}$
- Mean-flowtime: $\frac{1}{n}\sum_{j=1}^{n}(C_j - r_j)$
- Mean-weighted-tardiness: $\frac{1}{n}\sum_{j=1}^{n} w_j * max\{0, C_j - d_j\}$

where C_j is the completion time of a job J_j, r_j is the release time of J_j, d_j is the due date of J_j, w_j is the weight (importance) of job J_j, and n is the number of jobs.

B. Multi-objective Genetic Programming for DFJSS

GP has been widely used to learn scheduling heuristics for dynamic scheduling [5], [18], [35]. NSGP can be considered as a variation of NSGA-II [21] that incorporates GP with a non-dominated sorting strategy for multi-objective DFJSS. GP has several advantages that make it a natural fit for learning scheduling heuristics for dynamic scheduling. First, GP offers a flexible representation that can represent various scheduling heuristics for DFJSS. Even scheduling heuristics with the same behaviour can be represented using different genotypes, providing diverse genetic materials to generate promising scheduling heuristics during the evolutionary process. This diversity allows for the exploration of a wider range of possible scheduling solutions. Second, scheduling heuristics represented by GP can be considered as priority functions, allowing for straightforward integration of domain expertise into the scheduling heuristics. Third, scheduling heuristics that employ tree-based structures tend to be easy to interpret. Last but not least, it is quite efficient to use scheduling heuristics to make real-time decisions in DFJSS by prioritising machines and operations, especially small rules, which is a crucial aspect for their practical applications.

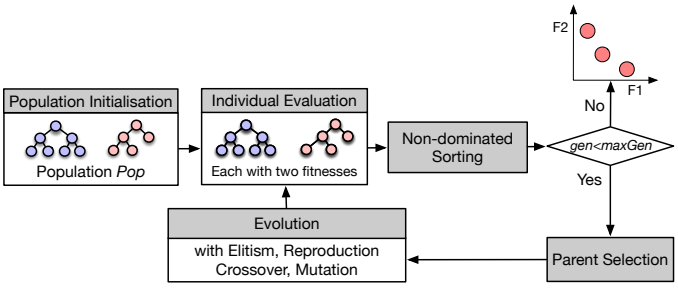


Fig. 1. The flowchart of NSGP to learn a Pareto front of scheduling heuristics for DFJSS.

Fig. 1 shows the flowchart of NSGP to learn scheduling heuristics for DFJSS. The main processes are the same as the typical GP for a single objective. The major difference is that the individuals have two objective values in multi-objective optimisation. GP mimics the evolutionary process in nature to improve the offspring generation by generation, and it has four main processes (i.e., initialisation, evaluation, parent selection, and evolution). GP starts with a number of randomly initialised individuals. The quality of each GP individual is measured with DFJSS instances (i.e., simulations) during the evaluation. If the stopping criterion (e.g., maximal number of generations, $maxGen$) is not met, parent selection is conducted to enhance the chance to produce new offspring with good quality by selecting individuals with good fitness as parents. Then, genetic operators, i.e., elitism, reproduction, crossover and mutation, are used to generate offspring based on the parents. Otherwise, the best learned Pareto front of scheduling heuristic so far is reported as the output of the NSGP algorithm for the DFJSS problem to be solved.

1) Representation: Using a routing rule for machine assignment and a sequencing rule for operation sequencing has shown to be an effective way to generate schedules for DFJSS [20], [36]. This paper applies multi-tree representation of GP to learn these two rules simultaneously [37]. Fig. 2 shows an example of a GP individual to represent the routing rule and the sequencing rule for DFJSS. The routing rule prioritises machines based on MWT + WIQ / NIQ, where MWT is needed time for a machine becomes idle, WIQ is the total time for a machine to finish the operations in the machine's queue, and NIQ is the number of operations in the queue of a machine. The sequencing rule is the well-known WSPT

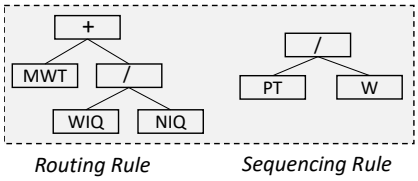


Fig. 2. An example of a GP individual with a routing rule and a sequencing rule for DFJSS.

TABLE I
AN EXAMPLE OF THE DECISION MAKING OF THE ROUTING RULE MWT + WIQ / NIQ AT A ROUTING DECISION POINT WITH THREE MACHINES.

Decision Situation	Machine Option	Feature (MWT	WIQ	NIQ)	Priority Value	Chosen Machine
1	M_1	200	500	25	220	M_2
	M_2	50	600	6	<u>150</u>	
	M_3	100	400	4	200	

rule, i.e., weighted shortest processing time, which prioritises operations according to PT / W, where PT is the processing time of an operation and W indicates the importance of an operation.

2) Decision Making with Scheduling Heuristics: In DFJSS, decision making is conducted at decision points, i.e., routing and sequencing decision points. Routing decision points are the situations in which an operation is ready to be processed (i.e., the first operation of a newly arrived job or the operation whose precedent operations have been processed). Sequencing decision points are the cases that when a machine becomes idle and there are operations waiting in its queue. Taking the routing decision process as an example, Table I shows an example of how the machines are selected to allocate a ready operation. Table I assumes the ready operation can be processed on machines M_1, M_2, and M_3. The priority values of the three machines are calculated based on the routing rule shown in Fig. 2. The priority values of M_1, M_2, and M_3 are calculated as 220, 150, and 200, respectively. As a result, the machine (i.e., M_2) with the smallest priority value (i.e., marked with an underline) is selected to process the operation.

C. α-dominance based Multi-objective GP

For job shop scheduling, most existing multi-objective GP algorithms are for learning the Pareto front for optimising objectives with similar difficulty where the non-dominance sorting will not bias specific objectives, e.g., max-flowtime and mean-flowtime, and max-tardiness and mean-tardiness [38], [39], [40]. Based on our preliminary investigations, these algorithms are not effective for multi-objective problems with biased objectives such as effectiveness and rule size.

Traditional dominance is a fundamental concept in multi-objective optimisation where a solution is said to dominate the other if it is better in at least one objective, and no worse in the other objectives [41], [42]. α-dominance is to identify the dominance relation among individuals [43]. As defined for α-dominance NSGP that optimises effectiveness and rule size [24], given two solutions x and y, we say that x α-dominates y if $\delta_{size}(x, y) \leq 0$, $\delta_{\mathit{eff}}(x, y) \leq 0$, and there is at least one

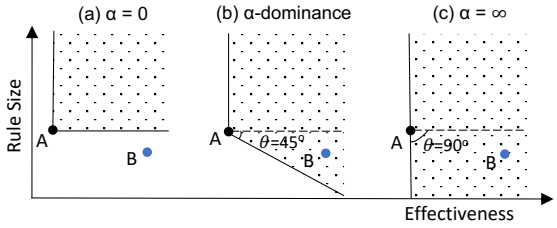


Fig. 3. An example of the dominance area with different α.

inequality, where

$$\delta_{size}(x, y) = size(x) - size(y) + \alpha * (\mathit{eff}(x) - \mathit{eff}(y)) \quad (1)$$

$$\delta_{\mathit{eff}}(x, y) = \mathit{eff}(x) - \mathit{eff}(y) \quad (2)$$

in which $size()$ indicates the size of rules, i.e., the number of nodes, and *eff()* represents the effectiveness of rules, e.g., max-flowtime. α-dominance based multi-objective GP has been successfully used in arc routing [22], [23], [24].

Fig. 3 shows an example of the dominance area with different α values, where θ represents the degree of α-dominance. α-dominance actually adjusts the objective bias by adapting α value to give the objective effectiveness more attention to be considered if needed. Taking some extreme as examples, when α is 0, it is actually the traditional dominance relation, and the multi-objective algorithm will consider the effectiveness and rule size equally. We can see with traditional dominance relation, solution A and B are non-dominated to each other as shown in Fig. 3(a). If we have α that is larger than 0, the α-dominance relation will bias to effectiveness. In this case, B will be dominated by A, and A has more chance to be kept in the population as shown in Fig. 3(b). If α is close to infinity, it means that we only consider effectiveness, which is actually a single objective optimisation as shown in Fig. 3(c). More details of α-dominance multi-objective GP can be found in [24], [25].

D. Related Work

Most existing work on GP for learning rules mainly focuses only on improving the effectiveness of rules [17], [38], [39], [44], [45], [46]. The studies that consider both the rule size and the effectiveness of rules are limited [24], [47]. According to the ways of studying the size and effectiveness of rules, we group the existing literature into two categories, i.e., *single-objective optimisation* and *multi-objective optimisation with biased objectives*.

1) Single Objective Optimisation: There are mainly three ways to learn small rules with GP under single objective optimisation. First, a simple way to learn simple and effective rules is to limit the rule size, e.g., tree depth [10], [48], [49]. However, how to set a proper limit for GP is non-trial, which is also problem dependent. Second, parsimony pressure is used to add a penalty to the fitness of each individual, and large rules might have bad fitness due to the penalty [50], [51]. Thus, small rules have higher chances than large rules to be selected as parents to generate small rules. Third, the learned rules can be simplified to be smaller [52]. A commonly used

Algorithm 1: The Proposed Multi-objective GP via Multitask Learning

```
   Input  : A multi-objective DFJSS task
   Output: Learned Pareto front of scheduling heuristics
1  gen ← 0
2  Initialisation: randomly initialise population with two subpopulations
3  while gen < maxGen do
4      Fitness evaluation for population (each individual has two objective values)
5      get ruleset_1 and ruleset_2 for knowledge sharing with the rule size range deciding strategy (Section III-B)
6      if gen < maxGen − 1 then
           // Evolution including parent selection and offspring generation
7          for i = 1 to 2 do
8              for j = 1 to subpopsize_i do
9                  if crossover then
10                     if random ≤ prob then
11                         Choose parent_1 from ruleset_i
12                         Choose parent_2 from the other rule set ruleset_¬i
13                         Produce one offspring with the origin-based offspring reservation strategy [53]
14                     else
15                         Choose two parents from subpopsize_i with tournament selection, and produce two offspring by conventional GP crossover
16                     end
17                 end
18                 if mutation or reproduction then
19                     Choose a parent from subpopsize_i and apply mutation or reproduction to generate one offspring
20                 end
21             end
22         end
23     end
24     gen ← gen + 1
25 end
26 return Learned Pareto front of subpopsize_1
```

way is to remove the redundant branches of GP individuals. However, detecting redundant branches is a challenging task, which might affect the effectiveness of rules.

2) Multi-objective with Biased Objectives: Studies of multi-objective optimisation with biased objectives including the rule size for learning scheduling heuristics in job shop scheduling are limited. However, GP has also been successfully used to learn routing policy for a similar dynamic combinatorial optimisation problem which is arc routing, this section will cover these related work as well. Taking the effectiveness and rule size as a multi-objective optimisation problem, α-dominance based NSGP with an archive has been successfully used to achieve small and effective routing policies [22], [23], [24], [25]. The key idea of these studies is to balance the search on effectiveness and rule size according to rule size in the population, and maintain an archive to save individuals in Pareto front across generations. These methods have shown their success in performance improvement on multi-objective problems with biased objectives. However, the key mechanisms of these algorithms are the control of the search of algorithms and the individual preservation in archive. There is no learning mechanism to improve the effectiveness of small rules, which is expected to improve the performance of multi-objective optimisation with effectiveness and rule size.

III. Proposed Multi-objective GP via Multitask Learning

A. Framework of the Proposed Algorithm

We divide the population into two subpopulations. In the first subpopulation, we use α-dominance based multi-objective GP that uses the Pareto front obtained from traditional non-dominated sorting as a reference to update α [25] (without archive) to learn scheduling heuristics (the first task). The first subpopulation will help to avoid the algorithm biases to very small rules. The second subpopulation will use traditional dominance relation for multi-objective optimisation which will bias small rule learning (the second task). In this framework, we consider $subpopulation_1$ for DFJSS as the main task that learns scheduling heuristics with α-dominance relation, while treats $subpopulation_2$ as the auxiliary task which learns scheduling heuristics without handling the objective bias issue. Overall, the learned scheduling heuristics in $subpopulation_1$ will be larger than the ones learned in $subpopulation_2$. It is noted that based on our preliminary investigations and also the conclusion in [23], neither the mechanism in $subpopulation_1$ nor $subpopulation_2$ can have good performance individually.

Algorithm 1 shows the framework of the proposed multi-objective GP via multitask learning. This paper proposes to improve the effectiveness of small rules in $subpopulation_1$ by knowledge sharing with the auxiliary task in $subpopulation_2$. Specifically, we will let rules with similar sizes from two subpopulations share knowledge to enhance their performance. Specifically, the candidate rules for knowledge sharing are decided by the proposed rule size range deciding strategy, and the rules ($ruleset_1$ for $subpopulation_1$ and $ruleset_2$ for $subpopulation_2$) within the size range will be selected (line 5). The knowledge sharing is realised by the crossover operator between rules in $ruleset_1$ and $ruleset_2$ (line 9 to line 13) with the origin-based offspring reservation strategy [53], i.e., save one offspring that corresponds to the parent for a particular task. In addition, the frequency of knowledge sharing is controlled by a parameter represented by $prob$, i.e., probability (line 10). Finally, the learned Pareto front of scheduling heuristics from $subpopulation_1$ will be reported as the output of the proposed algorithm.

B. Rule Size Range Deciding Strategy for Choosing Individuals for Knowledge Sharing

One key issue is how to decide which individuals can be used for knowledge sharing. As discussed earlier, in the proposed algorithm framework, the average rule size of $subpopulation_2$ will be smaller than that of $subpopulation_1$ due to the search bias to small rules in $subpopulation_2$. For knowledge sharing among individuals, intuitively, individuals with similar sizes are more likely to produce good offspring. In other words, breeding individuals from parents with quite different sizes tend to destroy individuals with small sizes or does not affect individuals with large sizes. An example of the mentioned issues can be found in Fig. 4. If we choose a large rule in $subpopulation_1$ (good effectiveness) and a small rule in $subpopulation_2$ (poor effectiveness) for crossover, the generated offspring $offspring_1$ is more likely not to be good

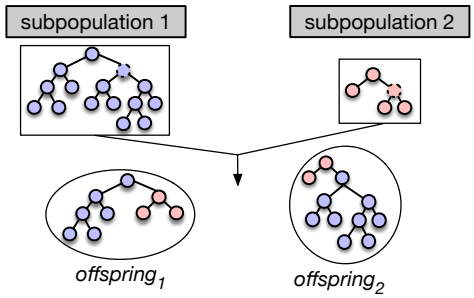


Fig. 4. An example of potential issue that uses a large rule and a small rule from $subpopulation_1$ and $subpopulation_2$ to generate offspring for via crossover.

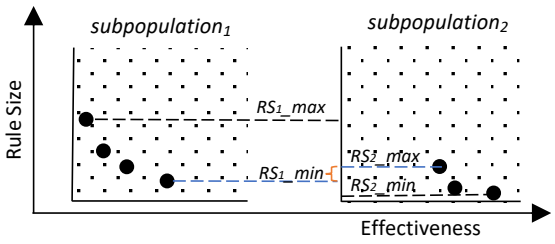


Fig. 5. An example of finding a rule size range to decide the individuals for knowledge sharing.

for $subpopulation_1$. The reason is that the potential large part of good genetic materials of the parent are destroyed, and the replaced genetic materials are not effective which are from small rules. Also, the generated offspring $offspring_2$ for $subpopulation_2$ is likely to be a large rule and will be abandoned quickly due to the small rule bias issue, thus be likely not benefit the evolutionary process of $subpopulation_2$.

To this end, this paper proposes to find the common rule size range of non-dominated individuals in the obtained Pareto fronts with traditional NSGP from two subpopulations as a filter to decide which individuals can be used to share knowledge. We use $ruleset_1$ and $ruleset_2$ to indicate the possible individual sets for knowledge sharing. Fig. 5 shows an example of finding the rule size range to decide the individuals for knowledge sharing. We propose to share knowledge among rules with similar sizes from two subpopulations. First, we will find the minimal rule sizes (RS_1_min and RS_2_min) and maximal rule sizes (RS_1_max and RS_2_max) of non-dominated individuals in the real Pareto fronts from $subpopulation_1$ and $subpopulation_2$. The range of rule sizes of individuals that can share knowledge with others will be $(max\{RS_1\_min, RS_2\_min\}, min\{RS_1\_max, RS_2\_max\})$. For the example in Fig. 5, it will be (RS_1_min, RS_2_max). More details can be found in Algorithm 2. We can see that the rule size range depends on the non-dominated individuals in the Pareto fronts obtained by the traditional NSGP from two subpopulations, but all individuals within this range in the population have chances to share knowledge with others.

C. Knowledge Sharing

Fig. 6 shows an example of knowledge sharing between subpopulations via crossover to generate offspring for $subpopulation_1$. When the crossover operator applies and $random \leq prob$ as shown in line 10 of Algorithm 1 where

Algorithm 2: Rule size range deciding strategy

Input : $subpopulation_1$, and $subpopulation_2$
Output: Chosen $ruleset_1$ and $ruleset_2$
1: $ruleset_1 = \emptyset$ and $ruleset_2 = \emptyset$
2: get non-dominated individuals $ParetoFront_1$ by traditional non-dominance relation for $subpopulation_1$
3: get non-dominated individuals $ParetoFront_2$ for $subpopulation_2$ with traditional non-dominance relation
4: get the rulesize range (RS_1_min, RS_1_max) of $ParetoFront_1$
5: get the rulesize range (RS_2_min, RS_2_max) of $ParetoFront_2$
6: the rules for knowledge sharing is within a range of $range \leftarrow (max\{RS_1\_min, RS_2\_min\}, min\{RS_1\_max, RS_2\_max\})$
7: $ruleset_1 \leftarrow$ individuals which are within $range$ from $subpopulation_1$
8: $ruleset_2 \leftarrow$ individuals which are within $range$ from $subpopulation_2$
9: **return** $ruleset_1$ and $ruleset_2$

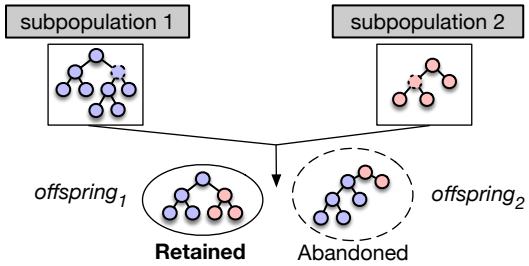


Fig. 6. An example of knowledge sharing between subpopulations via the crossover operator to generate offspring for $subpopulation_1$.

$prob$ is a transfer probability, two parents will be selected from $ruleset_1$ (subset of $subpopulation_1$) and $ruleset_2$ (subset of $subpopulation_2$), respectively. Following the suggestion in [53], we will randomly choose two subtrees and swap them to generate two offspring, and the one originally based on the parent from $subpopulation_1$ which is $offspring_1$ will be retained to the next generation for $subpopulation_1$. If we generate offspring for $subpopulation_2$, $offspring_2$ will be retained for $subpopulation_2$. Thus, the generated offspring consists of genetic materials from different subpopulations and the knowledge sharing is realised implicitly. Note that for the crossover between parents from the same subpopulation, both the generated two offspring will be selected for the next generation.

As we discussed earlier, the average rule sizes and effectiveness in $subpopulation_1$ are larger/better than the ones in $subpopulation_2$. The knowledge sharing via crossover between them is expected to have the following characteristics.

- The crossover operator for knowledge sharing tends to improve the effectiveness of small rules in $subpopulation_1$.
- This knowledge sharing design is expected to help improve the effectiveness of small rules in $subpopulation_1$ via shared knowledge, thus to improve the overall performance of the proposed algorithm. $subpopulation_1$ is the main focus of the proposed algorithm, and the output of the proposed algorithm is the final learned Pareto front from $subpopulation_1$.
- Although $subpopulation_2$ is used as auxiliary task, the knowledge sharing is also expected to improve the effectiveness of rules in $subpopulation_2$ due to the knowledge sharing with $subpopulation_1$. Note that this is a by-

product, and is not our focus.

D. Summary

Getting rid of the archive, the proposed multi-objective GP algorithm utilises the mechanism of multitask learning for handling multi-objective with biased objectives. The first task is to use α-dominance NSGP (without archive) to handle DFJSS, and the second task is to use traditional dominance relation to handle DFJSS. Compared with the state-of-the-art α-dominance based multi-objective GP with archive for DFJSS, the proposed algorithm that incorporates multitask learning mechanism has the following advantages.

- First, we do not need to maintain the archive which can heavily affect the algorithm performance.
- Second, the proposed multi-objective GP via multitask learning provides a learning mechanism to improve the effectiveness of rules. This learning mechanism tries to improve the effectiveness of small rules directly by knowledge sharing.
- Third, this paper effectively uses the advantages of α-dominance based multi-objective GP to avoid the search bias issue but also utilises the traditional dominance relation based multi-objective GP as an auxiliary task to learn scheduling heuristics for DFJSS.

IV. Experiment Design

A. Simulation Model

This paper is based on the commonly used DFJSS instances in [54], [55], and follows the same settings as [47]. These instances consider the processing of 5000 jobs by 10 machines, with new jobs arriving continuously according to a Poisson process characterised by a rate λ. The number of operations for each job is randomly generated from a discrete uniform distribution ranging from 1 to 10. Each operation's number of candidate machines is determined by a uniform discrete distribution with values ranging from 1 to 10. The processing time for each operation is assigned using a uniform discrete distribution that ranges from 1 to 99. Additionally, the due date for each job is set to 1.5 times its processing time. The job weights or importance are assigned such that 20%, 60%, and 20% of jobs have weights of 1, 2, and 4, respectively, as described in [56]. To enhance the overall generalisation ability of evolved scheduling heuristics, a new random seed for the simulation is assigned to change the training instance used for each generation, as described in [57].

The level of utilisation (p) is employed in simulating various job shop scenarios [58]. This factor denotes the proportion of time a machine is expected to be occupied, and it is regulated by λ in the Poisson process. To calculate the utilisation level, one needs to determine the average processing time of the machines (μ) and the probability of a job visiting a machine (P_M). For instance, if each job has two operations, P_M is 2/10. Then, the utilisation level is estimated as $\lambda = \mu * P_M / p$. Generally, a higher utilisation level results in a busier job shop.

In order to determine the steady-state performance, the initial 1000 jobs are designated as warm-up jobs and are not factored into the objective calculations. This study gathers data from the subsequent 5000 jobs, with the simulation terminating upon completion of the 6000th job.

TABLE II
The terminal set.

Notation	Description
NIQ	The number of operations in the queue
WIQ	Current work in the queue
MWT	Waiting time of a machine
PT	Processing time of an operation on a specified machine
NPT	Median processing time for the next operation
OWT	The waiting time of an operation
WKR	Median amount of work remaining for a job
NOR	The number of operations remaining for a job
W	Weight of a job
TIS	Time in system

B. Design of Comparisons

We consider three different objectives with rule size (RS), i.e., max-flowtime (Fmax) and RS, mean-flowtime (Fmean) and RS, and mean-weighted-tardiness (WTmean) and RS, and three utilisation levels (i.e., 0.75, 0.85, and 0.95) which are typical distinct configurations in DFJSS [59], [60] to generate examined scenarios. A scenario refers to a particular problem that needs to be resolved, which includes instances generated from the same problem but with varying configurations such as different objectives and utilisation levels.

- The traditional non-dominated sorting based GP, named NSGP, is the baseline algorithm. Note that NSGP is a variation of NSGA-II [21] by replacing the genetic algorithm in NSGA-II with GP for the investigated DFJSS problems.
- The multi-objective GP with α-dominance and archive [24], named αNSGP_a, is compared to verify the effectiveness of the proposed algorithms.
- The state-of-the-art multi-objective GP algorithm in DFJSS based on αNSGP_a but uses the real Pareto front as a reference to update α and archive [25] is named ref_αNSGP_a.
- The proposed multi-objective GP algorithm *via multitask learning* mechanism is named VMT_αNSGP.

VMT_αNSGP will be compared with all other algorithms to verify the performance of the proposed multi-objective GP via multitask learning.

C. Parameter Settings

Table II presents the features of the job shop, which serve as the terminals of GP according to [61]. These features are typically derived from the attributes of machines (NIQ, WIQ, and MWT), operations (PT, NPT, and OWT), and jobs (WKR, NOR, W, and TIS) that exist within the job shop environment. In accordance with [61], the function set used in this study consists of $\{+, -, *, /, max, min\}$, with each function requiring two arguments. Notably, the "protected division" function returns a value of one when dividing by zero. Table III outlines the other parameter settings utilised in this study,

TABLE III
THE PARAMETER SETTINGS IN GP.

Parameter	Value
Population size	1000
*subpopulation 1 size	1000 * $ratio$
*subpopulation 2 size	1000 * (1 - $ratio$)
The number of elites	10
Parent selection	Tournament selection with size 7
Crossover / Mutation / Reproduction rate	80% / 15% / 5%
Method for initialising population	ramped-half-and-half
Initial minimum / maximum depth	2 / 6
Maximal depth of programs	8
Terminal / non-terminal selection rate	10% / 90%
The number of generations	51
*transfer probability $prob$	0.1

* for VMT_αNSGP only

as suggested by [56], [61], [62]. Since VMT_αNSGP has two subpopulations, for fair comparison, its population size remains constant at 1000, the same as NSGP, αNSGP_a, and ref_αNSGP_a. This paper uses a $ratio$ to control the number of individuals for each subpopulation.

V. RESULTS AND DISCUSSIONS

The performance of the algorithms based on 30 independent runs is ranked using Friedman's test with a significance level of 0.05. If the results of Friedman's test are significant, we further conduct the Wilcoxon rank-sum test with Bonferroni correction between the proposed algorithm VMT_αNSGP, and the other algorithms, with a significance level of 0.05 for the post-hoc pairwise comparisons. The terms "Win, Draw, Lose" are used to indicate the number of scenarios where VMT_αNSGP performs statistically better, similarly, or worse than a compared algorithm. The "Average Rank" represents the algorithm's average ranking in all examined scenarios. Furthermore, the algorithm is compared with the algorithm(s) that come before it one by one.

In the presented results, the symbols "↑", "↓", and "≈" denote statistical significance, indicating that the corresponding result is significantly better than, worse than, or similar to its counterpart. The evaluation metrics used in this study are hyper volume (HV) and inverted generational distance (IGD), which are commonly used in multi-objective optimisation [63], [64]. A higher (lower) HV (IGD) indicates better performance. Since the true Pareto front is unknown in our problem, we use an approximated Pareto front obtained by finding the non-dominated solutions of the 30 independent runs of all compared algorithms for calculating IGD.

A. Sensitivity Analyses of Parameter $ratio$ for VMT_αNSGP

As discussed earlier, parameter $ratio$ affects the computational resource allocation for $subpopulation_1$ and $subpopulation_2$ of VMT_αNSGP by controlling the subpopulation sizes. Intuitively, a large $ratio$ gives more resources to learn scheduling heuristics with α-dominance multi-objective GP in $subpopulation_1$, and fewer resources for traditional non-dominance based multi-objective GP in $subpopulation_2$. However, it is not clear what is a good $ratio$.

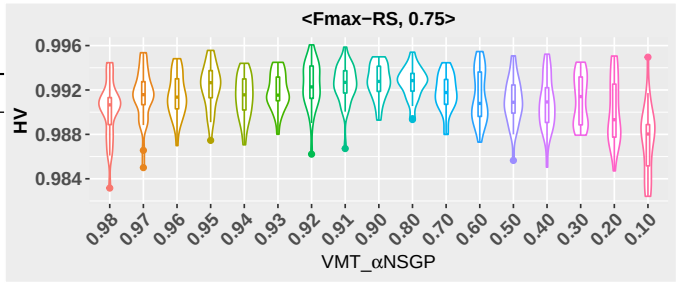


Fig. 7. Violin plots of training **HV** values of VMT_αNSGP with different ratios.

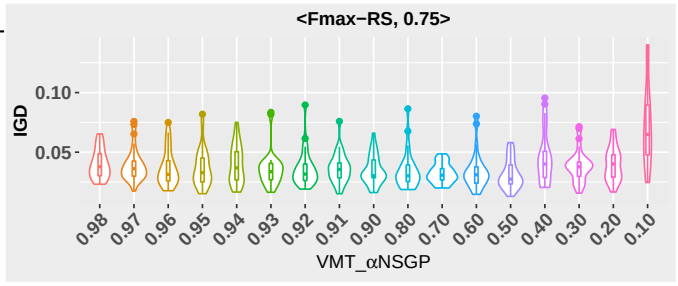


Fig. 8. Violin plots of training **IGD** values of VMT_αNSGP with different ratios.

To answer this question, this section investigates different settings of $ratio$, which are from 0.1 to 0.98. Fig. 7 shows the violin plots of training HV values of VMT_αNSGP with different ratio settings. We can see VMT_αNSGP achieves the best performance with $ratio$ of 0.9. In addition, we find that VMT_αNSGP with $ratio$ of 0.9 provides low standard deviations which indicates VMT_αNSGP with $ratio$ of 0.9 is more stable. Fig. 8 shows the violin plots of training IGD values of VMT_αNSGP with different ratio settings. The results also show that 0.9 is a good value to use for VMT_αNSGP since it achieves much better (smaller) IGD values than other settings. The same as our findings in HV, VMT_αNSGP with $ratio$ of 0.9 is also the most stable one that has the smallest standard deviation.

In summary, based on the parameter analyses of $ratio$, we decide to use $ratio$ of 0.9, and the following results of VMT_αNSGP are with $ratio$ of 0.9.

B. Quality of Learned Pareto Front

1) Statistical Test: Table IV shows the mean and standard deviations of HV and IGD values of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP on training and test instances over 30 runs in nine scenarios. Overall, in terms of HV and IGD, the results show that our proposed algorithm VMT_αNSGP achieves the best performance among all algorithms with the smallest average rank value in training and test. In addition, our proposed algorithm ref_αNSGP_a is the second best among the four compared algorithms.

For the HV in training, we can see that the adopted algorithm αNSGP_a from [24] is only significantly better than NSGP in 4 out of 9 scenarios. ref_αNSGP_a [25] is significantly better than NSGP in all scenarios, and also significantly better than αNSGP_a in 6 out of 9 scenarios. In

TABLE IV
THE MEAN (STANDARD DEVIATION) OF THE HV AND IGD ON **TRAINING AND TEST INSTANCES** OF NSGP, αNSGP_a, ref_αNSGP_a AND VMT_αNSGP BASED ON 30 INDEPENDENT RUNS IN NINE SCENARIOS WHICH ARE REPRESENTED BY THE OBJECTIVE AND UTILISATION LEVEL.

	HV				**IGD**			
	Training							
Scenario	NSGP	αNSGP_a	ref_αNSGP_a	VMT_αNSGP	NSGP	αNSGP_a	ref_αNSGP_a	VMT_αNSGP
<Fmax-RS, 0.75>	0.869(0.018)	0.889(0.031)(↑)	0.895(0.022)(↑≈)	0.940(0.014)(↑↑↑)	0.278(0.027)	0.159(0.051)(↑)	0.133(0.035)(↑↑)	0.070(0.023)(↑↑↑)
<Fmax-RS, 0.85>	0.840(0.033)	0.878(0.041)(↑)	0.894(0.019)(↑≈)	0.945(0.013)(↑↑↑)	0.272(0.031)	0.177(0.050)(↑)	0.137(0.037)(↑↑)	0.070(0.020)(↑↑↑)
<Fmax-RS, 0.95>	0.869(0.025)	0.887(0.053)(↑)	0.908(0.025)(↑≈)	0.954(0.010)(↑↑↑)	0.168(0.022)	0.121(0.044)(↑)	0.101(0.028)(↑≈)	0.048(0.014)(↑↑↑)
<Fmean-RS, 0.75>	0.982(0.002)	0.979(0.008)(≈)	0.987(0.003)(↑↑)	0.991(0.002)(↑↑↑)	0.198(0.034)	0.111(0.036)(↑)	0.063(0.034)(↑↑)	0.049(0.027)(↑↑≈)
<Fmean-RS, 0.85>	0.975(0.001)	0.976(0.012)(≈)	0.987(0.004)(↑↑)	0.990(0.003)(↑↑↑)	0.190(0.015)	0.115(0.036)(↑)	0.062(0.029)(↑↑)	0.053(0.020)(↑↑↑)
<Fmean-RS, 0.95>	0.969(0.003)	0.972(0.012)(≈)	0.985(0.006)(↑↑)	0.989(0.004)(↑↑↑)	0.065(0.016)	0.033(0.013)(↑)	0.023(0.008)(↑↑)	0.025(0.016)(↑↑≈)
<WTmean-RS, 0.75>	0.986(0.001)	0.981(0.011)(≈)	0.992(0.002)(↑↑)	0.994(0.002)(↑↑↑)	0.135(0.018)	0.074(0.026)(↑)	0.051(0.020)(↑↑)	0.033(0.014)(↑↑↑)
<WTmean-RS, 0.85>	0.977(0.000)	0.968(0.027)(≈)	0.987(0.004)(↑↑)	0.991(0.003)(↑↑↑)	0.114(0.011)	0.053(0.022)(↑)	0.027(0.014)(↑↑)	0.026(0.010)(↑↑≈)
<WTmean-RS, 0.95>	0.948(0.008)	0.968(0.015)(↑)	0.977(0.007)(↑↑)	0.982(0.007)(↑↑↑)	0.118(0.021)	0.038(0.012)(↑)	0.024(0.007)(↑↑)	0.025(0.012)(↑↑≈)
Win / Draw / Lose	9 / 0 / 0	9 / 0 / 0	9 / 0 / 0	N/A	9 / 0 / 0	9 / 0 / 0	5 / 4 / 0	N/A
Average Rank	3.59	3.03	2.14	**1.24**	3.94	2.74	1.92	**1.4**
	Test							
<Fmax-RS, 0.75>	0.835(0.033)	0.890(0.036)(↑)	0.890(0.025)(↑≈)	0.933(0.017)(↑↑↑)	0.222(0.030)	0.135(0.038)(↑)	0.122(0.019)(↑≈)	0.080(0.018)(↑↑↑)
<Fmax-RS, 0.85>	0.837(0.040)	0.903(0.034)(↑)	0.912(0.019)(↑≈)	0.948(0.015)(↑↑↑)	0.227(0.036)	0.115(0.035)(↑)	0.100(0.022)(↑≈)	0.065(0.018)(↑↑↑)
<Fmax-RS, 0.95>	0.907(0.023)	0.928(0.041)(↑)	0.943(0.023)(↑≈)	0.971(0.008)(↑↑↑)	0.131(0.023)	0.095(0.031)(↑)	0.081(0.024)(↑≈)	0.042(0.012)(↑↑↑)
<Fmean-RS, 0.75>	0.985(0.001)	0.980(0.008)(≈)	0.988(0.003)(↑↑)	0.991(0.002)(↑↑↑)	0.131(0.025)	0.066(0.025)(↑)	0.037(0.021)(↑↑)	0.040(0.015)(↑↑≈)
<Fmean-RS, 0.85>	0.980(0.001)	0.979(0.011)(≈)	0.990(0.004)(↑↑)	0.992(0.002)(↑↑↑)	0.091(0.014)	0.047(0.015)(↑)	0.027(0.010)(↑↑)	0.026(0.012)(↑↑≈)
<Fmean-RS, 0.95>	0.975(0.002)	0.975(0.013)(≈)	0.987(0.005)(↑↑)	0.991(0.003)(↑↑↑)	0.069(0.016)	0.036(0.015)(↑)	0.025(0.009)(↑↑)	0.021(0.009)(↑↑↑)
<WTmean-RS, 0.75>	0.988(0.001)	0.982(0.011)(↓)	0.993(0.002)(↑↑)	0.995(0.001)(↑↑↑)	0.095(0.019)	0.044(0.021)(↑)	0.024(0.010)(↑↑)	0.026(0.011)(↑↑≈)
<WTmean-RS, 0.85>	0.983(0.000)	0.971(0.027)(≈)	0.991(0.003)(↑↑)	0.994(0.002)(↑↑↑)	0.044(0.006)	0.023(0.010)(↑)	0.015(0.004)(↑↑)	0.009(0.007)(↑↑↑)
<WTmean-RS, 0.95>	0.964(0.003)	0.974(0.015)(↑)	0.983(0.006)(↑↑)	0.987(0.005)(↑↑↑)	0.110(0.015)	0.047(0.010)(↑)	0.027(0.009)(↑↑)	0.024(0.009)(↑↑≈)
Win / Draw / Lose	9 / 0 / 0	9 / 0 / 0	9 / 0 / 0	N/A	9 / 0 / 0	9 / 0 / 0	5 / 4 / 0	N/A
Average Rank	3.58	2.99	2.16	**1.27**	3.95	2.73	1.94	**1.38**

* One algorithm is compared with all algorithms before it. It might have more signals in (), if there is more than one algorithm for comparisons.

terms of IGD in training, the results show that our proposed ref_αNSGP_a outperforms αNSGP_a in 8 out of 9 scenarios. More importantly, for HV of test, we can see that our proposed ref_αNSGP_a has significantly better performance than NSGP in all scenarios, and outperforms αNSGP_a in most of the scenarios. However, we can see that αNSGP_a only performs better than NSGP in 4 out of 9 scenario, and even performs worse than NSGP in one scenario <WTmean-RS, 0.75>. In addition, for the IGD in test, ref_αNSGP_a performs better than αNSGP_a in six out of nine scenarios. We can see that the idea of using real Pareto front as a reference for α and archive updating represented by ref_αNSGP_a can achieve competitive scheduling heuristics for multi-objective with effectiveness and rule size. This finding is consistent with the conclusion in [25].

Compared with ref_αNSGP_a (the best among NSGP, αNSGP_a and ref_αNSGP_a), Table IV shows that VMT_αNSGP is significantly better than ref_αNSGP_a in all scenarios in terms of HV in both training and test scenarios. In addition, regarding IGD, VMT_αNSGP shows its superiority over ref_αNSGP_a in 5 out of 9 scenarios in both training and test. From the perspective of obtained average rank, VMT_αNSGP has a better rank than ref_αNSGP_a for HV on training (1.24 < 2.14) and test (1.27 < 2.16), and IGD on training (1.4 < 1.92) and test (1.38 < 1.94). This verifies the effectiveness of proposed VMT_αNSGP that utilises the mechanism of multitask learning for multi-objective optimisation with biased objectives.

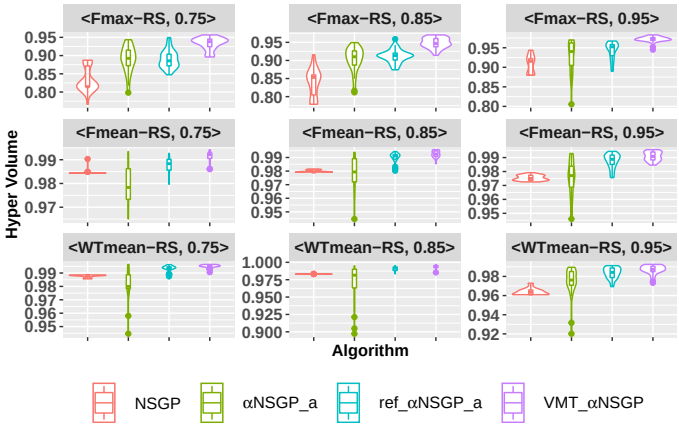


Fig. 9. Violin plots of **HV** values of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP over 30 independent runs in nine test scenarios.

2) Violin Plots of HV and IGD: Fig. 9 shows the violin plots of HV values of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP over 30 independent runs in nine test scenarios. First, we can see that ref_αNSGP_a performs better than NSGP and αNSGP_a with a better HV distribution of higher HV values. We also find that although αNSGP_a is better than NSGP overall, αNSGP_a has large standard deviations in all scenarios. This shows the drawback of using αNSGP_a dominance based multi-objective that utilises Pareto front obtained by α-dominance [24] to update α and

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3456086

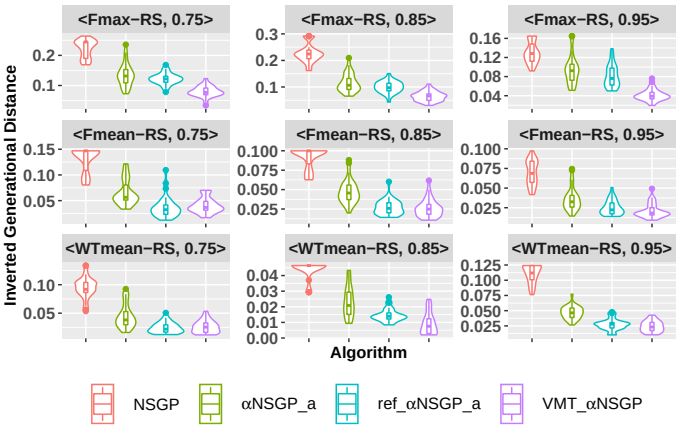


Fig. 10. Violin plots of **IGD** values of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP over 30 independent runs in nine test scenarios.

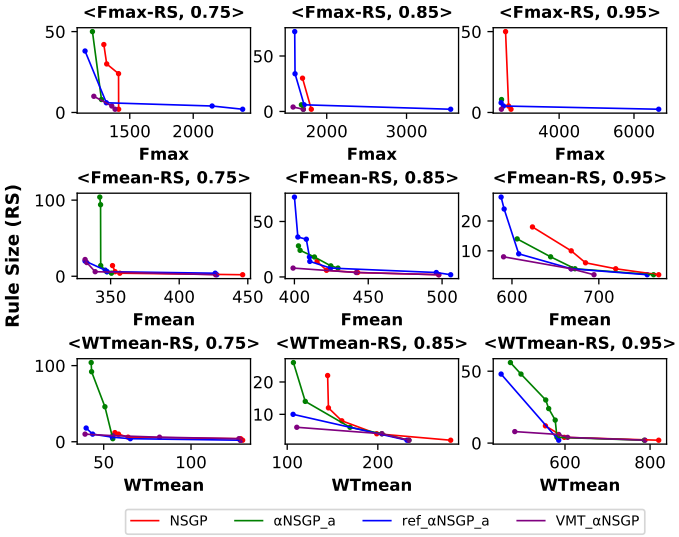


Fig. 11. Learned Pareto front of the run with medium HV values of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP in nine test scenarios.

archive. On the contrary, this shows the effectiveness of using non-dominated individuals from the real Pareto front for α and archive updating in [25]. Second, it is clear that the proposed VMT_αNSGP performs the best with the highest HV value distributions among all compared algorithms followed by ref_αNSGP_a. We can also see that the standard deviations of HV values obtained by VMT_αNSGP are quite small, which shows its stabilisation on performance for different instances.

Fig. 10 shows the violin plots of IGD values of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP over 30 independent runs in nine test scenarios. We can see the same pattern as discussed about HV in Fig. 9. In general, our proposed algorithm, especially VMT_αNSGP performs the best among all compared algorithms.

C. Learned Pareto Front

The problem investigated in this paper is a minimisation problem, a smaller rank value indicates a better performance. Fig. 11 shows the learned Pareto front of the run with medium HV value of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP in nine test scenarios. We can see that mostly

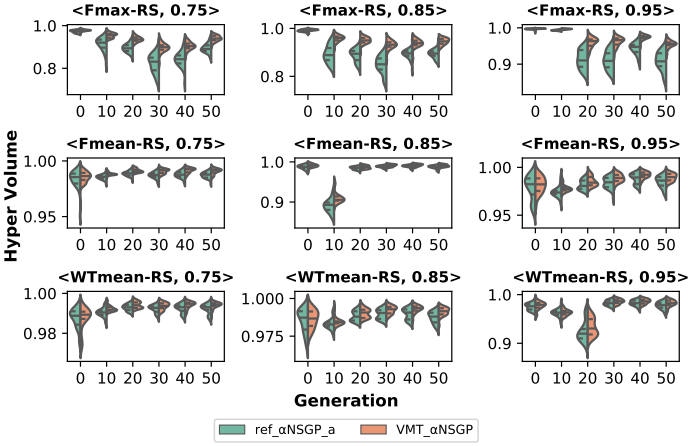


Fig. 12. Violin plots of test **HV** values of ref_αNSGP_a and VMT_αNSGP on generation 0, 10, 20, 30, 40 and 50 in nine scenarios.

NSGP can get some small scheduling heuristics, however, the effectiveness of the learned scheduling heuristics is not good as others. This is also the challenge we are handling in this paper. ref_αNSGP_a can obtain better Pareto fronts than αNSGP_a in different scenarios. This verifies the effectiveness of the proposed using non-dominated individuals from the real Pareto front as a reference for updating α and archive [25]. The proposed algorithm VMT_αNSGP can achieve the best Pareto front among compared algorithms. This verifies the effectiveness of the proposed idea of utilising multitask learning mechanism for multi-objective optimisation with biased objectives.

VI. Further Analyses

The previous section mainly focuses on the analyses of the effectiveness of the proposed algorithm. This section will further investigate the effects of the proposed algorithm in terms of its performance over generations, sizes of learned routing and sequencing rules in the population, sizes of scheduling heuristics for each task, and population diversity.

A. Quality of Learned Pareto Front Over Generations

From the previous section, we know that NSGP and αNSGP_a are much worse than ref_αNSGP_a and VMT_αNSGP, this section will only focus on ref_αNSGP_a and VMT_αNSGP to further analyse their performance. Figs. 12 and 13 show the violin plots of test HV and IGD of ref_αNSGP_a and VMT_αNSGP on six selected generations, i.e., generation 0, 10, 20, 30, 40 and 50, respectively. We can see that at the beginning (i.e., generation 0), the difference from ref_αNSGP_a and VMT_αNSGP is not large. Along with generations, VMT_αNSGP starts to be better than ref_αNSGP_a from an early stage, i.e., generation 10. Finally, VMT_αNSGP achieves better performance than ref_αNSGP_a with higher HV and smaller IGD. This shows that the learned scheduling heuristics across generations with proposed algorithm VMT_αNSGP have good generalisation ability to achieve good performance on unseen test scenarios.

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3456086

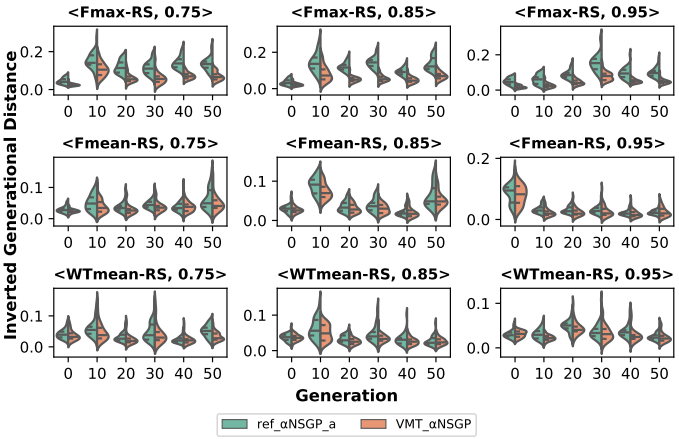


Fig. 13. Violin plots of test **IGD** values of ref_αNSGP_a and VMT_αNSGP on generation 0, 10, 20, 30, 40 and 50 in nine scenarios.

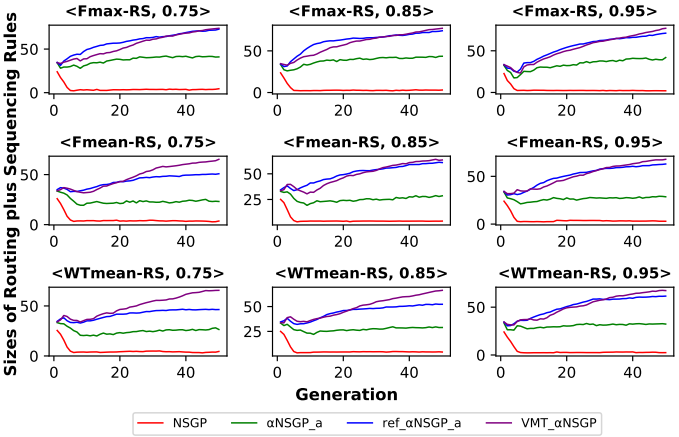


Fig. 14. Curves of average sizes of evolved scheduling heuristics (routing rule plus sequencing rule) in the population over generations of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP in nine scenarios.

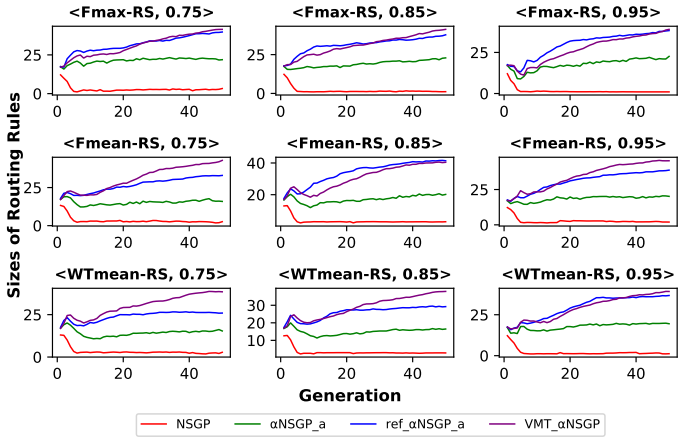


Fig. 15. Curves of average sizes of evolved **routing rules** in the population over generations of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP in nine scenarios.

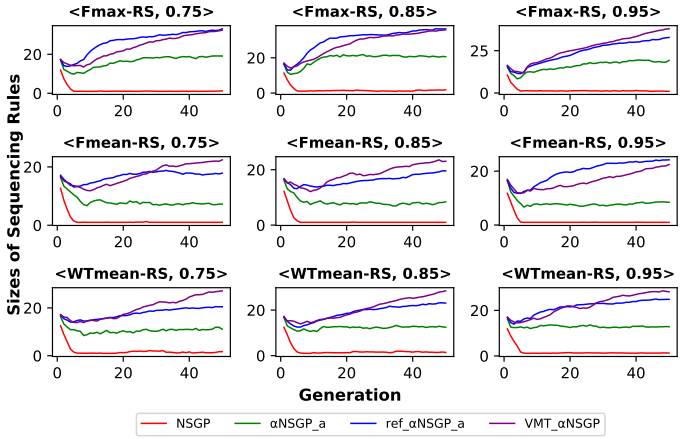


Fig. 16. Curves of average sizes of evolved **sequencing rules** in the population over generations of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP in nine scenarios.

B. Sizes of Learned Scheduling Heuristics in the Population

To have a better understanding of how the algorithms affect the sizes of learned scheduling heuristics, this section will investigate the average sizes of evolved scheduling heuristics over population. Since the routing rule and the sequencing rule work together to make schedules in DFJSS, it is reasonable to consider their sizes together [47].

Fig. 14 shows the curves of average sizes of evolved scheduling heuristics (routing plus sequencing) in the population over generations. The results show that the rule sizes of NSGP are smaller than αNSGP_a, ref_αNSGP_a and VMT_αNSGP from a very early stage, and then reduce dramatically along with generations, resulting in quite smaller rules. This is why NSGP cannot achieve good performance, since the traditional dominance relation based multi-objective optimisation has the issue of biasing the objective (e.g., rule size) that is easy to optimise.

α-dominance relation based algorithm including αNSGP_a, ref_αNSGP_a and VMT_αNSGP start with similar rule sizes, and can evolve rules larger than NSGP successfully. First, compared with NSGP, we can see that αNSGP_a with α-dominance can successfully evolve larger scheduling heuristics than NSGP in the evolutionary process. Second, compared with αNSGP_a, ref_αNSGP_a and VMT_αNSGP have successfully maintained the larger scheduling heuristics in the population. Third, by looking at the curves of ref_αNSGP_a and VMT_αNSGP, we can find that the rule size of ref_αNSGP_a and VMT_αNSGP are similar at the early stage of the evolutionary process. However, VMT_αNSGP can evolve slightly larger rules than ref_αNSGP_a in most of the scenarios from the second half of the evolutionary process (i.e., <Fmax-RS, 0.95>, <Fmean-RS, 0.75>, <Fmean-RS, 0.85>, <Fmean-RS, 0.95>, <WTmean-RS, 0.75>, <WTmean-RS, 0.85>, <WTmean-RS, 0.95>), and finally learns slightly larger scheduling heuristics.

It is interesting to know either the routing rule size or the sequencing rule size is increased. Figs. 15 and 16 show the curves of average sizes of evolved the routing rule and the sequencing rule in the population over generations of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP in nine scenarios, respectively. First, we find that the size of the routing rule is generally larger than the sequencing rule, which is consistent with the findings in [19], [47]. Second, although there are differences between the sizes of the routing rule and the sequencing rule, the patterns of the average sizes over population along with generations of the routing rule and the

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2024.3456086

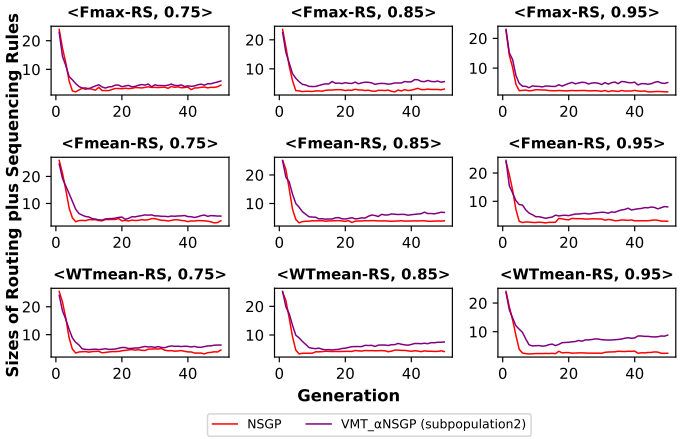


Fig. 17. Curves of average sizes of evolved scheduling heuristics (**routing rule plus sequencing rule**) in the population over generations of NSGP and VMT_αNSGP (subpopulation 2) in nine scenarios.

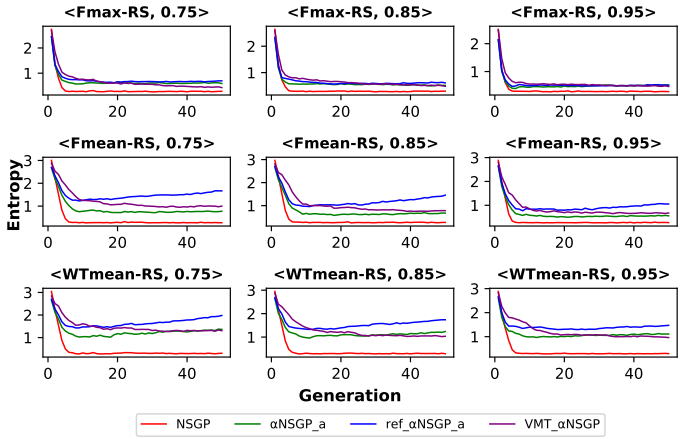


Fig. 18. Curves of entropy values which represents the diversity of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP over 30 independent runs.

sequencing rule are similar, and also similar to their sum rule size as shown in Fig. 14. This indicates the proposed VMT_αNSGP has successfully taken both the routing rule and the sequencing rule into consideration.

C. Sizes of Learned Scheduling Heuristics Obtained by NSGP and the Auxiliary Subpopulation of VMT_αNSGP

We know that NSGP only relies on dominance relation to learn the Pareto front and does not have any control over the objective bias issue in multi-objective optimisation. The auxiliary (second) subpopulation of VMT_αNSGP (named as VMT_αNSGP (subpopulation 2)) also does not have any control over the objective bias issue. The only difference between NSGP and VMT_αNSGP is that VMT_αNSGP shares knowledge between its first subpopulation with the same mechanism of ref_αNSGP_a. Another difference is that NSGP has a population size of 1000, while VMT_αNSGP (subpopulation 2) contains 100 individuals.

Fig. 17 shows the curves of average sizes of evolved scheduling heuristics (routing rule plus sequencing rule) in the population along with generations of NSGP and VMT_αNSGP (subpopulation 2). The results show that although VMT_αNSGP (subpopulation 2) has a smaller number of individuals, the sizes of rules are larger than NSGP. The reason is that VMT_αNSGP has knowledge sharing mechanism, and there are chances that the offspring in subpopulation 2 of VMT_αNSGP are generated with larger and more effective individuals in subpopulation 1 and thus more likely to have effective and large scheduling heuristics in subpopulation 2. This verifies the effectiveness of the proposed knowledge sharing with multitask learning mechanism in VMT_αNSGP from the perspective of the effect of algorithms on rule size.

D. Diversity of Population

To investigate the effect of the proposed algorithm on the population diversity, we use entropy to measure the diversity of individuals during the evolutionary process. The entropy is calculated as $entropy = -\sum_{c \in C} \frac{|c|}{|inds|} log(\frac{|c|}{|inds|})$, where C is the set of clusters obtained by the DBScan clustering algorithm [65] with the phenotypic distance measure [56] and a cluster radius of zero. $|inds|$ represents the number of individuals. A larger entropy value indicates a higher population diversity.

Fig. 18 shows the curves of entropy values of NSGP, αNSGP_a, ref_αNSGP_a and VMT_αNSGP over 30 independent runs. It is clear that NSGP loses diversity quickly and stays at a low value after around generation 5. This is consistent with our intuition that along with generations, NSGP only contains small scheduling heuristics which have a low diversity. Compared with NSGP, we can see that the population diversity is clearly increased by αNSGP_a which shows the effectiveness of αNSGP_a to maintain a population with large scheduling heuristics. In addition, ref_αNSGP_a and VMT_αNSGP have a good diversity across generations than αNSGP_a. Compared with ref_αNSGP_a, VMT_αNSGP has a higher diversity at the early stage (i.e., before about generation 10), and reaches a lower diversity in the late stages during the evolutionary process. This shows the effectiveness of the exploration and exploitation ability of VMT_αNSGP, which is one of the reasons that lead to the good performance of VMT_αNSGP. In other words, although ref_αNSGP_a can maintain a good population diversity, continuously increasing diversity in the late stages of the evolutionary process might decrease its performance.

VII. Conclusions

This paper successfully developed an effective multi-objective GP algorithm to learn scheduling heuristics for DFJSS with biased objectives. The proposed algorithm uses a multitask learning mechanism that shares knowledge between α-dominance and traditional dominance relations.

The proposed algorithm outperforms the state-of-the-art in scheduling heuristics for DFJSS in Pareto fronts. It uses a simpler yet more effective multi-objective GP via multitask learning, eliminating the need for an archive used by the α-dominance multi-objective algorithm. The quality of learned scheduling heuristics and the knowledge-sharing mechanism for performance improvement are verified by rule size and population diversity. The proposed algorithm evolves similar or slightly larger scheduling heuristics, including routing

and sequencing rules, compared to the state-of-the-art. The proposed algorithm preserves diverse populations early for exploration and maintains diversity for effective exploitation, outperforming the state-of-the-art.

Future research can explore the routing and sequencing rule sizes separately and model multi-objective optimisation with three objectives. Additionally, we will investigate multi-objective optimisation with various biased objectives.

REFERENCES

[1] A. S. Manne, "On the job-shop scheduling problem," *Operations Research*, vol. 8, no. 2, pp. 219–223, 1960.

[2] I. A. Chaudhry and A. A. Khan, "A research survey: review of flexible job shop scheduling techniques," *International Transactions in Operational Research*, vol. 23, no. 3, pp. 551–591, 2016.

[3] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 1, pp. 147–167, 2023.

[4] R. De Koster, T. Le-Duc, and K. J. Roodbergen, "Design and control of warehouse order picking: A literature review," *European Journal of Operational Research*, vol. 182, no. 2, pp. 481–501, 2007.

[5] J. Branke, S. Nguyen, C. W. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 110–124, 2016.

[6] L. Planinić, M. Durasević, and D. Jakobović, "Towards interpretable dispatching rules: Application of expression simplification methods," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*. IEEE, 2021, pp. 1–8.

[7] L. Meng, C. Zhang, Y. Ren, B. Zhang, and C. Lv, "Mixed-integer linear programming and constraint programming formulations for solving distributed flexible job shop scheduling problem," *Computers & Industrial Engineering*, vol. 142, p. 106347, 2020.

[8] S. C. Liu, Z. G. Chen, Z. H. Zhan, S. W. Jeon, S. Kwong, and J. Zhang, "Many-objective job-shop scheduling: a multiple populations for multiple objectives-based genetic algorithm approach," *IEEE Transactions on Cybernetics*, 2021.

[9] P. D. Dominic, S. Kaliyamoorthy, and M. S. Kumar, "Efficient dispatching rules for dynamic job shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 24, no. 1-2, pp. 70–75, 2004.

[10] E. Hart and K. Sim, "A hyper-heuristic ensemble method for static job-shop scheduling," *Evolutionary Computation*, vol. 24, no. 4, pp. 609–635, 2016.

[11] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches," in *Handbook of metaheuristics*. Springer, 2010, pp. 449–468.

[12] N. Pillay and W. Banzhaf, "A genetic programming approach to the generation of hyper-heuristics for the uncapacitated examination timetabling problem," in *Proceedings of the Portuguese Conference on Artificial Intelligence*, 2007, pp. 223–234.

[13] J. Lin, L. Zhu, and K. Gao, "A genetic programming hyper-heuristic approach for the multi-skill resource constrained project scheduling problem," *Expert Systems with Applications*, vol. 140, p. 112915, 2020.

[14] R. Ojstersek, M. Brezocnik, and B. Buchmeister, "Multi-objective optimization of production scheduling with evolutionary computation: A review," *International Journal of Industrial Engineering Computations*, vol. 11, no. 3, pp. 359–376, 2020.

[15] Y. Mei, Q. Chen, A. Lensen, B. Xue, and M. Zhang, "Explainable artificial intelligence by genetic programming: A survey," *IEEE Transactions on Evolutionary Computation*, 2022.

[16] D. Jakobović, L. Jelenković, and L. Budin, "Genetic programming heuristics for multiple machine scheduling," in *European Conference on Genetic Programming*. Springer, 2007, pp. 321–330.

[17] M. Xu, Y. Mei, F. Zhang, and M. Zhang, "Genetic programming with lexicase selection for large-scale dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2023. doi: 10.1109/TEVC.2023.3244607.

[18] M. Dumić and D. Jakobović, "Ensembles of priority rules for resource constrained project scheduling problem," *Applied Soft Computing*, p. 107606, 2021.

[19] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Importance-aware genetic programming for automated scheduling heuristics learning in dynamic flexible job shop scheduling," in *Proceedings of the Parallel Problem Solving from Nature*. Springer, 2022, pp. 48–62.

[20] ——, "Learning strategies on scheduling heuristics of genetic programming in dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2022, pp. 1–8.

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

[22] S. Wang, Y. Mei, and M. Zhang, "Two-stage multi-objective genetic programming with archive for uncertain capacitated arc routing problem," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2021, pp. 287–295.

[23] ——, "A multi-objective genetic programming approach with self-adaptive α dominance to uncertain capacitated arc routing problem," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2021, pp. 636–643.

[24] ——, "A multi-objective genetic programming algorithm with α dominance and archive for uncertain capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, 2022.

[25] G. Shi, F. Zhang, and Y. Mei, "Interpretability-aware multi-objective genetic programming for scheduling heuristics learning in dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2023, pp. 1–8.

[26] A. Gupta, Y. S. Ong, and L. Feng, "Multifactorial evolution: toward evolutionary multitasking," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 343–357, 2015.

[27] J. Lin, H. L. Liu, K. C. Tan, and F. Gu, "An effective knowledge transfer approach for multiobjective multitasking optimization," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3238–3248, 2021.

[28] Z. Huang, Y. Mei, F. Zhang, and M. Zhang, "Multitask linear genetic programming with shared individuals and its application to dynamic job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2023. DOI: 10.1109/TEVC.2023.3263871.

[29] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Multitask genetic programming-based generative hyper-heuristics: A case study in dynamic scheduling," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10 515–10 528, 2021.

[30] L. Zhou, L. Feng, J. Zhong, Y. S. Ong, Z. Zhu, and E. Sha, "Evolutionary multitasking in combinatorial search spaces: A case study in capacitated vehicle routing problem," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*. IEEE, 2016, pp. 1–8.

[31] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Task relatedness based multitask genetic programming for dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 6, pp. 1705–1719, 2022.

[32] P. Brucker and R. Schlie, "Job-shop scheduling with multi-purpose machines," *Computing*, vol. 45, no. 4, pp. 369–375, 1990.

[33] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "A preliminary approach to evolutionary multitasking for dynamic flexible job shop scheduling via genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2020, pp. 107–108.

[34] M. Durasevic and D. Jakobovic, "A survey of dispatching rules for the dynamic unrelated machines environment," *Expert Systems with Applications*, vol. 113, pp. 555–569, 2018.

[35] S. Nguyen, D. Thiruvady, M. Zhang, and D. Alahakoon, "Automated design of multipass heuristics for resource-constrained job scheduling with self-competitive genetic programming," *IEEE Transactions on Cybernetics*, vol. 52, no. 9, pp. 8603–8616, 2021.

[36] Y. Zhou, J. Yang, and Z. Huang, "Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming," *International Journal of Production Research*, vol. 58, no. 9, pp. 2561–2580, 2020.

[37] F. Zhang, Y. Mei, and M. Zhang, "Genetic programming with multi-tree representation for dynamic flexible job shop scheduling," in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 472–484.

[38] ——, "Evolving dispatching rules for multi-objective dynamic flexible job shop scheduling via genetic programming hyper-heuristics," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2019, pp. 1366–1373.

[39] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Multitask multiobjective genetic programming for automated scheduling heuristic learning in dynamic flexible job-shop scheduling," *IEEE Transactions on Cybernetics*, vol. 53, no. 7, pp. 4473–4486, 2022.

[40] S. Nguyen, M. Zhang, and K. C. Tan, "Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems," in *Proceedings of the Congress on Evolutionary Computation*. IEEE, 2015, pp. 2781–2788.

[41] A. Goli, A. Ala, and M. Hajiaghaei-Keshteli, "Efficient multi-objective meta-heuristic algorithms for energy-aware non-permutation flow-shop scheduling problem," *Expert Systems with Applications*, vol. 213, p. 119077, 2023.

[42] J. G. Falcón-Cardona, H. Ishibuchi, C. A. C. Coello, and M. Emmerich, "On the effect of the cooperation of indicator-based multiobjective evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 681–695, 2021.

[43] K. Ikeda, H. Kita, and S. Kobayashi, "Failure of pareto-based moeas: Does non-dominated really mean near to optimal?" in *Proceedings of the Congress on Evolutionary Computation*, vol. 2. IEEE, 2001, pp. 957–962.

[44] M. DJurasevic, F. J. Gil-Gala, and D. Jakobovic, "Constructing ensembles of dispatching rules for multi-objective tasks in the unrelated machines environment," *Integrated Computer-Aided Engineering*, no. Preprint, pp. 1–18, 2023.

[45] H. Fan, H. Xiong, and M. Goh, "Genetic programming-based hyper-heuristic approach for solving dynamic job shop scheduling problem with extended technical precedence constraints," *Computers & Operations Research*, vol. 134, p. 105401, 2021.

[46] 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, 2018.

[47] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job shop scheduling," *IEEE Transactions on Cybernetics*, vol. 51, no. 4, pp. 1797–1811, 2021.

[48] R. Braune, F. Benda, K. F. Doerner, and R. F. Hartl, "A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems," *International Journal of Production Economics*, vol. 243, p. 108342, 2022.

[49] Z. Zojaji, M. M. Ebadzadeh, and H. Nasiri, "Semantic schema based genetic programming for symbolic regression," *Applied Soft Computing*, vol. 122, p. 108825, 2022.

[50] L. Hong, J. H. Drake, J. R. Woodward, and E. Özcan, "A hyper-heuristic approach to automated generation of mutation operators for evolutionary programming," *Applied Soft Computing*, vol. 62, pp. 162–175, 2018.

[51] A. S. Sambo, R. M. A. Azad, Y. Kovalchuk, V. P. Indramohan, and H. Shah, "Time control or size control? reducing complexity and improving accuracy of genetic programming models," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2020, pp. 195–210.

[52] S. Panda, Y. Mei, and M. Zhang, "Simplifying dispatching rules in genetic programming for dynamic job shop scheduling," in *Proceedings of the Evolutionary Computation in Combinatorial Optimization*. Springer, 2022, pp. 95–110.

[53] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Multitask genetic programming based generative hyper-heuristics: A case study in dynamic scheduling," *IEEE Transactions on Cybernetics*, vol. 52, no. 10, pp. 10 515–10 528, 2022.

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

[55] S. Nguyen, M. Zhang, D. Alahakoon, and K. C. Tan, "Visualizing the evolution of computer programs for genetic programming," *IEEE Computational Intelligence Magazine*, vol. 13, no. 4, pp. 77–94, 2018.

[56] T. Hildebrandt and J. Branke, "On using surrogates with genetic programming," *Evolutionary Computation*, vol. 23, no. 3, pp. 343–367, 2015.

[57] S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: a survey with a unified framework," *Complex & Intelligent Systems*, vol. 3, no. 1, pp. 41–66, 2017.

[58] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic programming via iterated local search for dynamic job shop scheduling," *IEEE Transactions on Cybernetics*, vol. 45, no. 1, pp. 1–14, 2015.

[59] F. Zhang, Y. Mei, and M. Zhang, "A new representation in genetic programming for evolving dispatching rules for dynamic flexible job shop scheduling," in *Proceedings of the European Conference on Evolutionary Computation in Combinatorial Optimization*. Springer, 2019, pp. 33–49.

[60] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Correlation coefficient based recombinative guidance for genetic programming hyper-heuristics in dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 552–566, 2021.

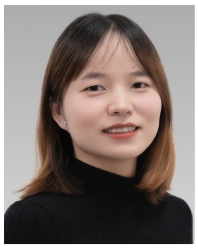
[61] F. Zhang, Y. Mei, S. Nguyen, M. Zhang, and K. C. Tan, "Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 4, pp. 651–665, 2021.

[62] J. R. Koza and R. Poli, "Genetic programming," in *Search Methodologies*. Springer, 2005, pp. 127–164.

[63] Y. Tian, X. Zhang, R. Cheng, and Y. Jin, "A multi-objective evolutionary algorithm based on an enhanced inverted generational distance metric," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 5222–5229.

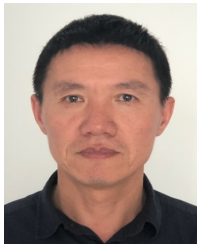
[64] H. Sabireen and N. Venkataraman, "A hybrid and light weight meta-heuristic approach with clustering for multi-objective resource scheduling and application placement in fog environment," *Expert Systems with Applications*, p. 119895, 2023.

[65] M. Ester, H. P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 226–231.

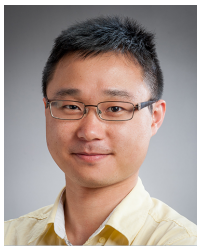


Fangfang Zhang (Member, IEEE) received the B.Sc. and M.Sc. degrees from Shenzhen University, China, and the Ph.D. degree in computer science from Victoria University of Wellington, New Zealand, in 2014, 2017, and 2021, respectively.

She is currently a lecturer with the Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. Her research interests include evolutionary computation, hyper-heuristic learning/optimisation, job shop scheduling, surrogate, and multitask learning. Dr. Fangfang is an Associate Editor of Expert Systems With Applications, and Swarm and Evolutionary Computation. She is the secretary of the IEEE New Zealand Central Section.

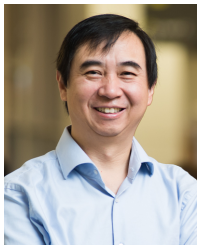


Gaofeng Shi received the Master of Computer Science from Victoria University of Wellington, New Zealand in 2022. He graduated with information management diploma from East China University of Science and Technology, China in 2004, and Graduate Diploma in Computing from Unitec, New Zealand in 2016. His study and research interests include machining learning, evolutionary computation and job shop scheduling. He is currently a control system engineer in Emerson.



Yi Mei (M'09-SM'18) received the B.Sc. and Ph.D. degrees from the University of Science and Technology of China, Hefei, China, in 2005 and 2010, respectively.

He is an Associate Professor with the School of Engineering and Computer Science, Victoria University of Wellington, Wellington, New Zealand. His research interests include evolutionary scheduling and combinatorial optimisation, machine learning, genetic programming, and hyper-heuristics.



Mengjie Zhang (M'04-SM'10-F'19) received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Baoding, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively.

He is currently a Professor of Computer Science, Victoria University of Wellington, New Zealand. He is a Fellow of the Royal Society, and Fellow of Engineering of New Zealand, a Fellow of IEEE and an IEEE Distinguished Lecturer.