Evolving Scheduling Heuristics via Genetic Programming with Feature Selection in Dynamic Flexible Job Shop Scheduling

Fangfang Zhang, *Student Member, IEEE,* Yi Mei, *Senior Member, IEEE,* Su Nguyen, *Member, IEEE,* and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—Dynamic flexible job shop scheduling (DFJSS) is a challenging combinational optimisation problem that takes the dynamic environment into account. Genetic programming hyper-heuristics (GPHH) have been widely used to evolve scheduling heuristics for job shop scheduling. A proper selection of the terminal set is a critical factor for the success of GPHH. However, there is a wide range of features that can capture different characteristics of the job shop state. Moreover, the importance of a feature is unclear from one scenario to another. The irrelevant and redundant features may lead to performance limitations. Feature selection is an important task to select relevant and complementary features. However, little work has considered feature selection in GPHH for DFJSS. In this paper, a novel two-stage GPHH framework with feature selection is designed to evolve scheduling heuristics only with the selected features for DFJSS automatically. Meanwhile, individual adaptation strategies are proposed to utilise the information of both the selected features and the investigated individuals during the feature selection process. The results show that the proposed algorithm can successfully achieve more interpretable scheduling heuristics with fewer unique features and smaller sizes. In addition, the proposed algorithm can reach comparable scheduling heuristic quality with much shorter training time.**

***Index Terms*—Feature Selection, Genetic Programming, Hyper-heuristics, Interpretability, Dynamic Flexible Job Shop Scheduling.**

I. Introduction

Job shop scheduling (JSS) [1] is an important research problem which captures practical issues in real-world scheduling tasks such as manufacturing processes [2], [3] and cloud computing [4], especially in large-scale production environments. JSS is to process a number of jobs (each job has a sequence of operations) by a set of machines, where each operation can only be processed by a specific machine. Flexible JSS (FJSS) [5], as an extension of JSS, is closer to reality. Different from in JSS, an operation in FJSS can be processed by more than one candidate machine, and the processing time on each machine is different. In FJSS, two kinds of decisions need to be made simultaneously. One is *machine assignment* (i.e., assign an operation to a machine) and the other is *operation sequencing* (i.e., choose the next operation to be processed when a machine becomes idle).

The authors are with the Evolutionary Computation Research Group at the 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;). Su Nguyen is with the Centre for Data Analytics and Cognition, La Trobe University, Australia (e-mail: P.Nguyen4@latrobe.edu.au)

Unlike in JSS and DJSS, dynamic FJSS (DFJSS) aims to make decisions of both machine assignment and operation sequencing under a dynamic environment with unpredicted events such as new job arrivals [6], [7] and machine breakdown [8], [9]. DFJSS is an NP-hard problem [10] and cannot be solved efficiently with *exact optimisation methods* such as dynamic programming [11] and integer linear programming [12]. *Approximate solution optimisation methods* such as simulated annealing [13], tabu search [14], genetic algorithm [15], and particle swarm optimisation [16], [17], which aim to find a near-optimal solution have been applied for static JSS and FJSS. However, they cannot effectively handle dynamic environment, which requires real-time response. *Scheduling heuristics* such as priority-based dispatching rules [18], are the most popularly used heuristics for DFJSS. Scheduling heuristics are used to make real-time decisions based on the priority values of machines or operations at the decision points. A scheduling heuristic in DFJSS consists of a *routing rule* for machine assignment and a *sequencing rule* for operation sequencing [3]. There are some advantages of using scheduling heuristics. First, scheduling heuristics can make decisions in real-time, thus can handle dynamic problems well. Second, the scheduling heuristics can be implemented to real-world applications easily. Last but not least, domain knowledge can be easily incorporated with priority-based scheduling heuristics. However, it is hard to manually design effective rules due to the complexity of the investigated job shop environment.

Genetic programming (GP), as a hyper-heuristic (GPHH) method, has been successfully applied to automatically evolve scheduling heuristics for JSS, including FJSS and DFJSS [7], [19], [20], [21]. Except for scheduling heuristics, to the best of our knowledge, there are no other strategies in typical GPHH related to handling the dynamic features of the problems. A GP individual is a priority function, typically represented as a tree. GP evolves a population of such trees using a terminal set (i.e., the leaf nodes, reflecting the features of the job shop state) and a function set (i.e., non-leaf nodes, indicating the operations to combine the features in the priority function). The terminal set is a critical factor in the success of GPHH [3]. A compact terminal set can improve the effectiveness of GPHH. In DFJSS, a wide range of features about the job shop state (e.g., the processing time of each operation and the idle time of each machine) can be considered as terminals. However, the importance of a feature depends on job shop scenarios and objectives to be optimised. In practice, it is usually unknown which features are useful, which are not and which are more important than others. Therefore, existing studies typically place all the possible job shop features in the terminal set. As a result, the evolved rules tend to have a large number of different features, making it hard to interpret the rules [22]. Besides, a large terminal set with redundant or unrelated features leads to exponentially large and noisy search space, and negatively affects the capability of GP in searching the solution space.

To address the above issues, this paper proposes to use *feature selection* in GPHH for DFJSS. Feature selection [23] has been successfully used for different tasks such as classification [24], [25], [26], clustering [27], and regression [28]. It is noted that feature selection can not only reduce the search space of GP but also potentially improve the interpretability of evolved scheduling heuristics in DFJSS. The fewer features involved in scheduling heuristics, the easier it is to interpret the rules, as there are potentially fewer and hopeful less complicated relationships between the features. The frequency of terminals has been widely considered in feature selection based on the assumption that GP can automatically perform feature selection [29]. However, the embedded feature selection ability in GP is limited. The redundant branches in GP are likely to mislead the accuracy of feature selection, which can weaken the ability of problem-solving, such as the classification accuracy in the classification problems [29].

To the best of our knowledge, little is yet known about using feature selection in the variations of JSS. Feature selection based on the frequency of terminals was introduced to help GP to evolve dispatching rules for dynamic JSS in [30]. A novel feature importance measure instead of frequency was firstly introduced in [31] to select features for the dynamic JSS problem. Then, an efficient feature selection was proposed in [32] based on the feature importance measure in [31]. However, these approaches are only related to dynamic JSS. The feature selection technique was firstly used for DFJSS in [33]. In [33], a GPHH approach with feature selection was proposed for DFJSS, which involves two feature sets. However, the main drawback in [33] is that the selected features are only used to guide the behaviour of GP by mutation operator. It does not change the evolution sufficiently, which greatly limits the influence of the feature selection. This points to an important question how to apply the selected features effectively after obtaining a great feature set. It is still an important but an unexplored research topic in DFJSS.

The feature selection in GPHH for DFJSS is different from and more challenging than the traditional machine learning tasks, as the data is not available and the data should be generated before applying feature selection. The current state-of-the-art feature selection approach [32] used a short GP process with surrogate and niching techniques to evolve a diverse set of good GP individuals as the data for feature selection. Then the features were selected based on their importance to these individuals. There are two types of information obtained from the feature selection process. The first is the selected features, and the second is the promising individuals found during the feature selection process (i.e., the final population in this study). Most existing algorithms [34], [35] only use the first one and re-initialise the population using selected features. To improve effectiveness, this paper proposes novel algorithms to employ both types of information. Specifically, individual adaptation strategies are proposed to utilise the information of the selected features and examined individuals.

The overall goal of this paper is to *develop an effective feature selection approach with novel individual adaptation strategies via genetic programming* to automatically evolve more interpretable routing and sequencing rules simultaneously for DFJSS efficiently. The proposed algorithms are expected to help GPHH find more *interpretable rules only with selected features without sacrificing the performance*. Specifically, this work has the following research objectives:

1) Develop a new two-stage GPHH framework to utilise the information of both the selected features and the examined individuals during the feature selection process.
2) Propose novel individual adaptation strategies that inherit the information of the examined individuals during the feature selection.
3) Analyse how the proposed algorithms influence the effectiveness and interpretability of the evolved rules.
4) Analyse how the proposed individual adaptation strategies influence the efficiency of the proposed algorithms.

II. Background

A. Dynamic Flexible Job Shop Scheduling

In FJSS problem, n jobs $J = \{J_1, J_2, ..., J_n\}$ need to be processed by m machines $M = \{M_1, M_2, ..., M_m\}$. Each job J_j has an arrival time $at(J_i)$ and a sequence of operations $O_j = (O_{j1}, O_{j2}, ..., O_{ji})$. Each operation O_{ji} can only be processed by one of its optional machines $\pi(O_{ji})$ and its processing time $\delta(O_{ji})$ depends on the machine that processes it. It indicates that there are two decisions which are routing decision and sequencing decision in FJSS. In DFJSS, not only the two decisions need to be made simultaneously, but also the dynamic events are necessary to be taken into account when making schedules. This paper focuses on one dynamic event (i.e., continuously arriving new jobs). That is, the information of a job is unknown until its arrival time.

B. Genetic Programming Hyper-heuristics for JSS

A hyper-heuristic [36] is a heuristic search method that seeks to select or generate heuristics to efficiently solve hard computational search problems. The unique characteristic is that hyper-heuristic works on heuristic space rather than solution space. GPHH [37] has been successfully applied to evolve informative scheduling heuristics for combinational optimisation problems such as packing [38], timetabling [39], [40], arc routing [41], and dynamic JSS [42], [43], [44], [45]. At the beginning, a population of individuals with a size of $popsize$ are randomly initialised. In GP, the fitness of the i^{th} individual $fitness_{ind_i}$ is evaluated as follows. First, a simulation (*training instance*) is run with the individual ind_i to obtain the schedule S_i. Then, the fitness of the individual is assigned as the objective value (e.g., max-flowtime, mean-flowtime, and mean-weighted-flowtime) of the obtained schedule $Obj(S_i)$.

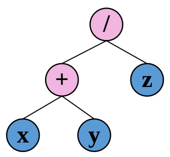


Fig. 1. An example of a GP individual with three features (e.g., x, y and z) and two arithmetic operators (e.g., + and /).

GP can automatically generate computer programs to solve problems without rich domain knowledge. An example of a GP individual [46] with three features (e.g., x, y and z) and two arithmetic operators (e.g., + and /) can be found in Fig. 1. The corresponding scheduling heuristic of this individual is $\frac{x+y}{z}$, and it will be used to prioritise the candidates, either machines or operations. There are some advantages of using GPHH for JSS, including DFJSS. One is its flexible representation. The structures of rules are not necessary to be designed in advance. The other is that the tree-based programs obtained by GP provide us opportunities to understand the behaviour of the rules, which is important for real-world applications.

C. Feature Selection

The performance of GP heavily relies on a proper selection of the terminal set [3]. In the terminal set, features (terminals) are not equally important. Besides, some features may be irrelevant, redundant or noisy, and the original features are typically not informative enough. All of these factors may lead to various performance limitations. Feature selection is an effective process for selecting a subset of relevant and complementary features [26], [47]. Feature selection algorithms are generally classified into three categories [26]: filter approaches, wrapper approaches, and embedded approaches.

However, there are some challenges which make traditional feature selection methods not directly applicable in DFJSS. First, the task (i.e., prioritising operations or machines) in DFJSS and the training instance are different from the traditional machine learning tasks. The training data are generated with the simulation execution in DFJSS while the training data already exist in the traditional machine learning tasks. In this case, filter approaches can not be applied since it is impossible to measure the importance of each feature based on filter measures such as entropy [48] and Pearson's correlation [49]. Second, it is much more computationally expensive if applying wrapper classifier in DFJSS than that of in traditional machine tasks. Specifically, running a GP process to obtain a reliable estimation of the best objective value of a terminal set is much slower than training a classifier (e.g., decision tree) in traditional machine tasks. Besides, in most embedded approaches, GP can handle both the feature selection and the regression [28], [50] or classification [51] tasks. However, they are the supervised problems, and feature selection is rarely used in the variations of JSS.

This paper extends the feature selection framework in [33] to only use selected features to evolve rules for DFJSS. In addition, this paper proposes novel individual adaptation strategies to help GPHH explore more interpretable rules only with selected features without losing the qualities of the rules.

III. The proposed algorithm

This section describes the proposed two-stage GP algorithm for DFJSS. The framework of the proposed algorithm is first illustrated, followed by the details of its key components.

A. Two-stage Framework

To utilise the information of both the selected features and investigated individuals during the feature selection process, this paper proposes a two-stage GP with feature selection and individual adaptation strategies. We introduce the framework in [6] with two subpopulations to evolve routing and sequencing rules simultaneously with the cooperation coevolutionary strategy for DFJSS. Thus, we can get two selected feature subsets, one for the routing rule, and the other for the sequencing rule. They are to measure the characteristics of the routing and sequencing rules, respectively. The flowchart of the two-stage framework is shown in Fig. 2. In stage 1, the conventional evaluation, selection and evolutionary operators are used. In stage 2, the initialisation and mutation are different. The initialisation adapts the final population of stage 1 as part of the initial population, and randomly generates the remaining individuals using the selected features. The mutation operator generates random sub-trees using the selected features only.

When using the feature selection method proposed in [32], a diverse set of good individuals are required to achieve high accuracy of the feature selection. The reason is that the good individuals in GP can be repeated, choosing all of them may be biased to specific features. *Stage 1* is designed to obtain a population with such individuals for feature selection. *The output of stage 1 is an informative population*. After stage 1 is completed, feature selection is conducted based on the final population obtained in stage 1.

Stage 2 is developed for making use of the information (i.e., final population and selected features) obtained in stage 1 to evolve more effective and interpretable rules. This paper develops a number of novel individual adaptation strategies to initialise the population in stage 2 that consists only of the selected features without deteriorating the performance much. It is noted that the selected features will be used in two situations in stage 2. One is to generate new individuals for building a new population during the initialisation process by the ramped half-and-half method. The other is to generate a new sub-tree by the grow method with only the selected features as the terminals to replace a selected subtree of a parent by mutation. Readers interested in the genetic operators of GP can refer to [46].

B. Stage 1

Extra feature selection process will make the approach more computationally expensive because more individual evaluations are needed. To reduce extra computing costs, a niching based and surrogate assisted method was proposed for feature selection in [32]. Simply speaking, niching technique maintains the diversity of individuals by building different niches and controlling the number of individuals in each niche. The method aims to get a diverse set of good individuals quickly

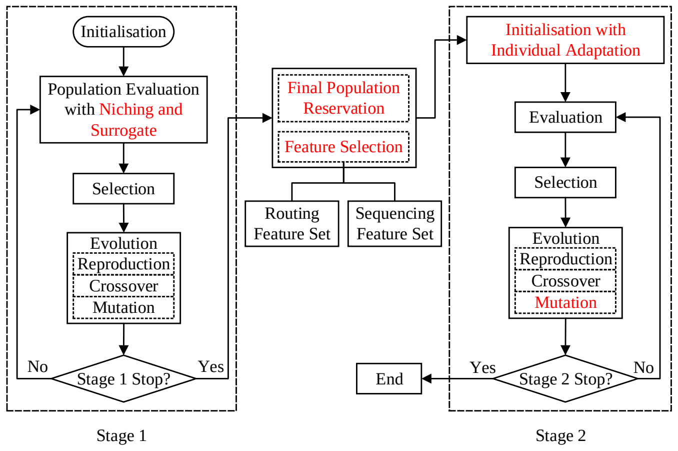


Fig. 2. The flowchart of two-stage genetic programming with feature selection and individual adaptation strategies (i.e., the reddish font parts are the main steps of the proposed algorithm).

Algorithm 1: Feature selection

```
1:  Input  : A diverse set of good individuals ($baseInds$) from stage 1
    Output: The selected features $F$
2:  set $F \leftarrow \{\}$
3:  for $i$ = 1 to $|features|$ do
4:      $vote_{f_i} \leftarrow 0$ // the number of votes for feature $f_i$
5:      for $j$ = 1 to $|baseInds|$ do
6:          Calculate the contribution $C_{f_i}$ of feature $f_i$
7:          $ind \leftarrow baseInds_j$
8:          if $C_{f_i}^{ind} > 0$ then
9:              $vote_{f_i} \leftarrow vote_{f_i}$ +1
10:         end
11:     end
12:     if $vote_{f_i} > 0.5 * |baseInds|$ then
13:         $T \leftarrow T \cup f_i$
14:     end
15: end
16: return $F$
```

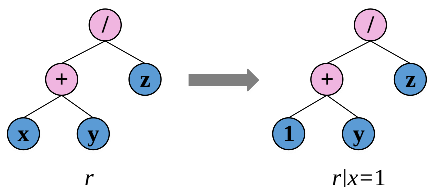


Fig. 3. An example of how to examine the contribution (denoted as C_x) of a feature x for an individual r.

for feature selection with surrogates, which is a simplified version of the original problem by only reducing the number of jobs. The surrogate technique was designed based on the assumption that the knowledge of solving simpler or auxiliary problems can be transferred to the original problems [52]. This paper applies the idea but explores it to the dynamic *flexible* JSS. The process is designed as stage 1 to get a diverse set of good individuals in this paper. In the original simulation, there are 5000 jobs and 10 machines. In the surrogate model, we shorten the simulation to 500 jobs and 5 machines.

C. Feature Selection

This paper applies the feature selection idea in [32]. There are three main steps in this feature selection method. First, top ten individuals in the population based on fitness values are selected as a diverse set of good individuals $baseInds$. Second, the importance of each feature is measured according to its contributions to the fitness of the individuals in $baseInds$ and an individual in $baseInds$ will vote for a feature if the feature has contributions to it. Finally, if a feature can get more than half of the votes, the feature will be selected. The pseudo-code of feature selection is shown in Algorithm 1.

The Importance of Features. The importance of a feature f is measured by its contributions to a set of individuals $baseInds$ (from line 4 to line 11). To calculate the contribution of a feature f to an individual r (i.e., denoted by C_f^r), the feature f is first replaced with the constant of one, then the contribution is calculated as the difference between the fitness before and after the replacement, as shown in Eq. (1).

$$C_f^r = fitness(r|f = 1) - fitness(r) \tag{1}$$

This paper is seeking to minimise the objective value. If $C_f > 0$, it means the fitness becomes worse without the measured feature, and the measured feature is an important feature. Thus, the measured feature can get one vote from the individual r.

Fig. 3 shows an example of a GP individual r with three features (x, y, and z). To examine the importance of feature x, x is firstly be replaced with 1, and the contribution of feature x is defined as $C_x^r = fitness(r|x = 1) - fitness(r)$.

Feature Selection Decision. This paper makes two extensions of the feature selection method in [32] to fit the DFJSS problems. First, two sets of individuals obtained from two sub-populations for evolving routing rules and sequencing rules are selected, respectively. Second, the feature selection method is applied for selecting the routing feature set and the sequencing feature set based on the two sets of individuals, respectively. Feature f is selected if it makes positive contributions to at least 50% of the selected individuals $baseInds$ (from line 12 to line 14).

D. Stage 2

Another important task is to inherit the information of the promising individuals in the final population of stage 1 while removing the unselected features in stage 2. To this end, we propose two strategies to adapt the individuals in the final population of stage 1 to stage 2. The idea is to generate new individuals with only the selected features but still have the same or similar behaviour with the promising individuals obtained in stage 1.

The first individual adaptation strategy is to simply replace each unselected feature with a constant of one. This can completely remove the unselected features from the individuals, while still keeping their structures as much as possible. If a feature is not selected, it is expected to have little contribution to a majority of individuals, and thus replacing them by one would not change the fitness much. A potential drawback is that the average quality of individuals in the first generation of stage 2 might not be as well as the last generation of stage 1. One reason is that the unselected features might still have some contributions to the quality of the individuals slightly. Another reason is that replacing a number of unselected features in an

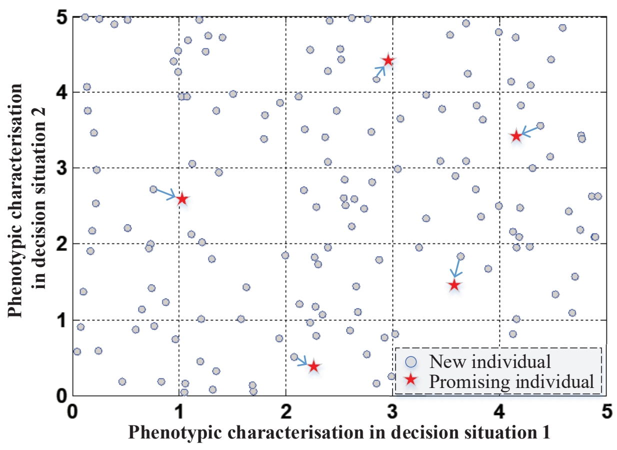


Fig. 4. The process of mimicking individuals by generating new individuals only with selected features.

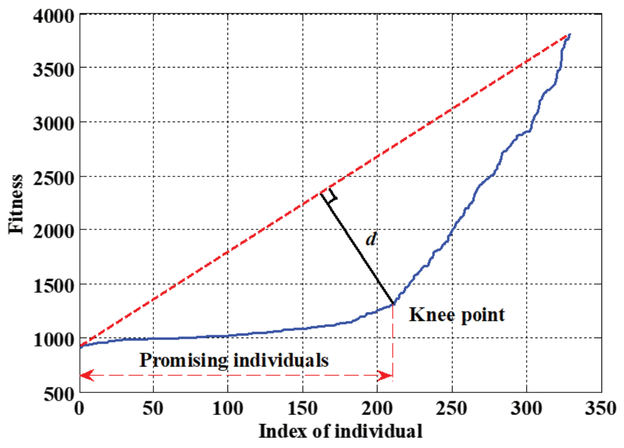


Fig. 5. The selected promising individuals based on knee-point.

individual by one will change the structure of an individual a lot, which is more likely to change the behaviour of the individual in certain ways.

The second individual adaptation strategy is based on the idea of "mimicking". Specifically, it randomly generates a large number of individuals with only the selected features. For each promising individual in the final population of stage 1, it is replaced by the randomly generated individual that has the closest behaviour with it. The behaviour is defined as the phenotypic characterisation [53], which is a numeric vector. Since the calculation of phenotypic characterisation is much cheaper than a full fitness evaluation, it is affordable to generate a large number of individuals for mimicking.

Fig. 4 shows an example of the process of mimicking individuals, where the phenotypic characterisation is 2-dimensional. Two decision situations are used to generate the phenotypic characterisation. A decision situation is when a rule is to make a decision (e.g., a machine becomes idle or an operation becomes ready). Briefly speaking, the phenotypic characterisation of a decision is defined as the rank of the machine or operation selected by an individual in the sorted list of the benchmark rules (e.g., SPT (shortest processing time) for sequencing decisions, and WIQ (work in the queue) for routing decisions). Interested readers to phenotypic characterisation can find more details in [53]. The stars indicate the promising individuals in stage 1 that need to be mimicked. A large number of new individuals (i.e., denoted as circles) are generated only with the selected features. The new individuals which are closest to the mimicked individuals will be chosen to replace the promising stage 1 individuals.

It is noted that it is not always possible to find individuals with the same behaviour (i.e., the distance between two individuals equals zero). The new individuals that have the most similar behaviours with promising individuals so far will be saved to the new population. For the rest of the elements (individuals) in the new population which are still empty will be randomly initialised with selected features. Finally, a new population only with the selected features is obtained.

It is noted that it is not meaningful to adapt all the individuals obtained from stage 1. Only the "promising" individuals are adapted in this paper. If the number of adapted individuals is too small, there will be too many randomly generated individuals, and the performance will be close to purely re-initialisation. If it is too large, it will bring some noise, and increase the training time.

This paper identifies the promising individuals using the knee point, which is a parameter-free approach. The knee point can be used as a demarcation point. The individuals whose fitnesses are smaller (i.e., minimising problem) than that of knee point are selected as promising individuals. An example can be found in Fig. 5. First, the individuals in the population are sorted based on fitness values in ascending order, and a curve related to fitness values is obtained. Second, a line is generated by connecting the points with the smallest and the largest fitness value. Then, the distance between each point on the curve and the line is calculated. The point that has the largest distance to the line is selected as the knee point, and the individuals whose fitness value is smaller than that of the knee point are selected to be adapted. Note that if there is more than one knee point, the knee point that has the largest distance to the generated line will be selected.

IV. Experiment Design

To investigate the *effectiveness* (the objective value on test instance), *efficiency* (the training time) in different scenarios of proposed *individual adaptation strategies*, a set of experiments have been designed.

A. Simulation Model

Simulation is an effective method to investigate complex problems [54]. The simulated environment in this paper is used as an experimental model to study factors affecting DFJSS. This paper assumes that there are 5000 jobs that need to be processed by ten machines in our simulation. For DFJSS simulation, new jobs will arrive over time according to a Poisson process with rate λ. Each job has a different number of operations that are randomly generated by a uniform discrete distribution between one and ten. In addition, the importance of jobs might be different, which are indicated by weights. The weights of 20%, 60%, and 20% of jobs are set as one, two,

TABLE I
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

TABLE II
THE PARAMETER SETTING OF GP.

Parameter	Value
Number of subpopulations	2
Subpopulation size	512
Method for initialising population	ramped-half-and-half
Initial minimum/maximum depth	2 / 6
maximal depth of programs	8
Crossover/Mutation/Reproduction rate	80% / 15% / 5%
Parent selection	Tournament selection with size 7
Number of generations in stage 1 and stage 2	50 / 50
Terminal/non-terminal selection rate	10% / 90%

and four, respectively. The processing time of each operation is assigned by uniform discrete distribution with the range [1,99]. The number of candidate machines for an operation follows a uniform discrete distribution between one and ten.

Utilisation level (p) is an essential factor to simulate different scenario environments. The bigger the utilisation level, the busier the job shop. To make sure the accuracy of collected data, warm-up jobs (first 1000 jobs) are used to get typical situations occurring in a long-term simulation of a dynamic job shop system and jobs arrive as a continuous arrival process. This paper collects data from the next 5000 jobs. The simulation stops when the 6000th jobs is finished.

B. Parameter Setting

In our experiment, the terminal set of GP is shown in Table I. The terminals are set as the features that indicate the characteristics related to machines (e.g., NIQ, WIQ, and MWT), operations (e.g., PT, NPT, and OWT), and jobs (e.g., WKR, NOR, W, and TIS). The function set is $\{+, -, *, /, max, min\}$, following the setting in [31], [55]. The arithmetic operators take two arguments. The "/" operator is protected division, returning one if divided by zero. The max and min functions take two arguments and return the maximum and minimum of their arguments, respectively. The other parameter settings of GP are shown in Table II.

C. Comparison Design

Five algorithms are taken into comparison in this paper. The cooperative coevolution genetic programming (CCGP) [6] (i.e., without feature selection and individual adaptation strategies) is selected as the baseline algorithm. The algorithm that was proposed in [33] (CCGP^2) is also compared. CCGP^2 only applies the selected features in stage 2 by mutation, and it is used to verify whether mutation operator will affect the performance. The proposed algorithms which incorporate with individual adaptation strategies are named as CCGP^{2a}(mimic) (i.e., mimicking the behaviour of promising individuals) and CCGP^{2a}(rep) (i.e., replacing the unselected features by one directly). To verify the effectiveness of CCGP^{2a}(mimic) and CCGP^{2a}(rep), the algorithm (i.e., named as CCGP^{2a}(rand)) that generates new population in stage 2 by randomly initialising all individuals is also compared. This is because using the selected features to re-initialise the new population in stage 2 randomly is the most straightforward way to eliminate unselected features intuitively.

D. The Measurement for Comparision

In order to verify the effectiveness and efficiency, the proposed algorithms are tested on *six different scenarios*. The scenarios consist of three objectives (e.g., max flowtime, mean flowtime, and mean weighted flowtime) and two utilisation levels (e.g., 0.85 and 0.95). For the sake of convenience, Fmax, Fmean, and WFmean stand for the max flowtime, mean flowtime, and mean weighted flowtime, respectively. The objective functions are shown as follows.

- Minimisation Fmax $= max\{C_1, C_i, ..., C_n\}$
- Minimisation Fmean $= \frac{\sum_{i=1}^{n}\{C_i - r_i\}}{n}$
- Minimisation WFmean $= \frac{\sum_{i=1}^{n} w_i * \{C_i - r_i\}}{n}$

where C_i is the completion time of job J_i, r_i is the release time of J_i, and w_i is the weight of J_i.

The evolved rule is tested on 50 different test instances and the average objective value across the 50 test instances is reported as the test performance of a rule, which can be a good approximation of the true performance of the rule.

V. RESULTS AND DISCUSSIONS

Wilcoxon signed-rank test with a significance level of 0.05 is used to verify the performance of proposed algorithms. In the following results, "-", "+", and "=" indicate the corresponding result is significantly better than, worse than or similar with its counterpart. To be specific, CCGP^2 is compared with CCGP while CCGP^{2a}(rand), CCGP^{2a}(rep), and CCGP^{2a}(mimic) are compared with CCGP^2. It is worth mentioning that this paper does not focus on the improvement of effectiveness but on the improvement of achieving more interpretable rules with fewer unique features and smaller size without losing the effectiveness.

A. Performance of Evolved Rules

Table III shows the mean and standard deviation of involved five algorithms according to 30 independent runs in six dynamic flexible scenarios. CCGP^2 which only makes use of selected features by mutation operator achieves similar performance with CCGP. One possible reason is that the GP itself can detect important features automatically. The other is that there is not much difference when only applying selected features by mutation with a small rate. However, the drawback is that the evolved rules by CCGP^2 still contain unselected

TABLE III
THE MEAN (STANDARD DEVIATION) OF THE OBJECTIVE VALUE OF THE FIVE ALGORITHMS OVER 30 INDEPENDENT RUNS FOR SIX DYNAMIC SCENARIOS.

Scenario	CCGP	CCGP^2	CCGP^{2a}(rand)	CCGP^{2a}(rep)	CCGP^{2a}(mimic)
<Fmax,0.85>	1223.83(41.78)	1225.41(43.51)(=)	1314.87(121.35)(+)	1237.53(81.28)(=)	1238.34(99.27)(=)
<Fmax,0.95>	1959.24(46.63)	1998.09(115.26)(=)	2054.56(204.36)(+)	2032.85(145.16)(=)	2034.08(153.61)(=)
<Fmean,0.85>	385.42(2.65)	384.77(1.32)(=)	387.34(2.23)(=)	385.07(1.24)(=)	385.14(1.87)(=)
<Fmean,0.95>	553.65(7.89)	552.88(6.78)(=)	559.21(8.21)(+)	553.07(6.31)(=)	551.20(6.11)(=)
<WFmean,0.85>	830.74(6.89)	829.58(5.56)(=)	833.02(6.15)(+)	830.11(5.42)(=)	831.51(6.52)(=)
<WFmean,0.95>	1109.89(13.07)	1110.86(12.01)(=)	1112.35(12.91)(=)	1109.58(7.96)(=)	1112.94(14.62)(=)

features. It does not make it easier to interpret the rules since more features are still involved in the rules.

The performance of CCGP^{2a}(rand) is significantly worse than that of CCGP in most scenarios. One reason might be that a completely new population which has worse performance (i.e., a new start), is generated for stage 2. It is hard to achieve good performance compared with CCGP (i.e., actually evolved by 100 generations). CCGP^{2a}(mimic) and CCGP^{2a}(rep) (i.e., only with selected features) can achieve similar performance with CCGP^2 in most scenarios. It indicates that the proposed adaptation strategies are effective to take advantage of the population information from stage 1.

It is noted that all the algorithms have larger variances in scenario <Fmax,0.85> and <Fmax,0.95>, especially the algorithms with feature selection (e.g., CCGP^2, CCGP^{2a}(rand), CCGP^{2a}(rep), and CCGP^{2a}(mimic)). One possible reason is that max flowtime is more sensitive to the worst case of processing jobs than mean flowtime. The other is that the selected features might be not that good for all the problems which will lead to some outliers.

Fig. 6 shows the convergence curves of the average objective value on unseen instances of CCGP, CCGP^2, CCGP^{2a}(rand), CCGP^{2a}(rep), and CCGP^{2a}(mimic) according to 30 independent runs in different scenarios. In all scenarios, CCGP^{2a}(mimic), and CCGP^{2a}(rep) can mimic the behaviours well (i.e., shown at generation 50) that the performance does not decrease too much. CCGP^{2a}(mimic) can achieve almost the same performance in <Fmax,0.95> in the following several generations of generation 50 compared with CCGP^2. Compared with CCGP^{2a}(mimic), CCGP^{2a}(rep) even can get almost the same performance with CCGP^2 in all scenarios. This means that CCGP^{2a}(rep) has a more promising inheritance ability.

In general, the effectivenesses of CCGP^{2a}(mimic) and CCGP^{2a}(rep) are better than that of CCGP^{2a}(rand). Both CCGP^{2a}(mimic) and CCGP^{2a}(rep) can inherit the individuals' information well from stage 1 to stage 2.

B. Rule Size Analysis

The rule size is defined as the number of nodes in this paper, and the rule with a smaller size is preferred. There are a number of advantages of evolving smaller rules. First, smaller rules can save the training time. Second, smaller rules tend to be more interpretable by decision makers, particularly the floor operators of the job shop. Third, smaller rules are more acceptable by decision makers than larger rules due to the less complexities. Last but not the least, smaller rules are easier to

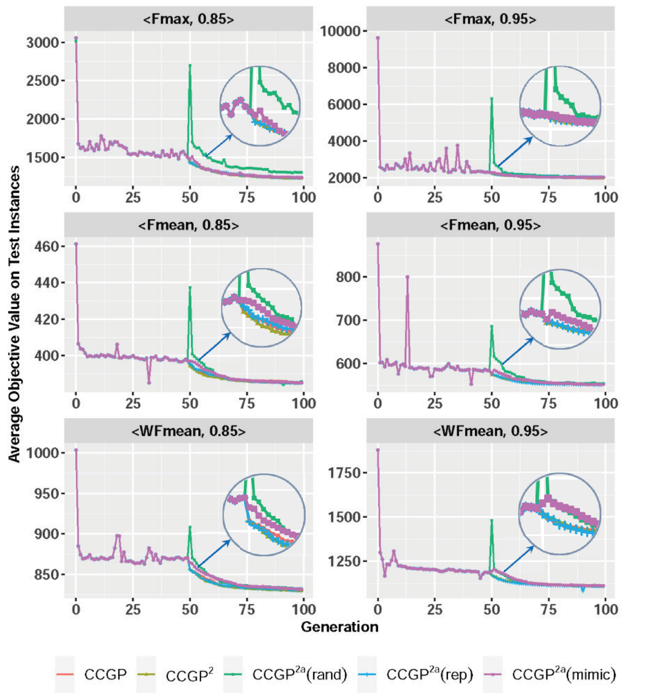


Fig. 6. The curves of average objective value on test instances of the five algorithms according to 30 independent runs in different scenarios.

implement to real-world applications, which are more efficient to make real-time decisions with dynamic events compared with larger rules.

Fig. 7 and Fig. 8 show the curves of the sizes (i.e., the mean value of 30 independent runs at each generation) of routing rules and sequencing rules. Both for routing rules and sequencing rules, the rule sizes of CCGP^{2a}(mimic) and CCGP^{2a}(rand) decrease dramatically at the beginning of stage 2 due to the individual adaptation strategies, especially the routing rules. It is noted that the size of the routing rules of CCGP^{2a}(rep) is not that small compared with CCGP^{2a}(mimic) and CCGP^{2a}(rand). This is because the structures of the large rules are still kept.

The size of the best routing rules obtained by CCGP^{2a}(mimic) is smaller than that of other algorithms in most scenarios (e.g., <Fmax,0.85>, <Fmean,0.85>, <Fmean,0.95>, <WFmean,0.85>, and <WFmean,0.95>). However, the size of the best routing rules of CCGP^{2a}(mimic) is similar with that of other algorithms in <Fmax,0.95>. This may be because Fmax with a higher utilisation level (i.e., 0.95) is more difficult to be optimised due to its sensitiveness to the worst case (i.e., the longest finished time among all

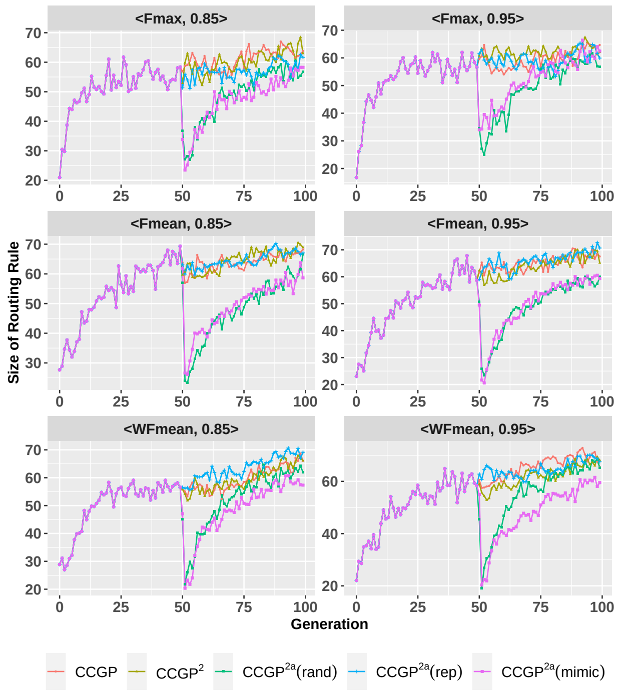


Fig. 7. The curves of *the best routing rule sizes* of the population of the five algorithms according to 30 independent runs in different scenarios.

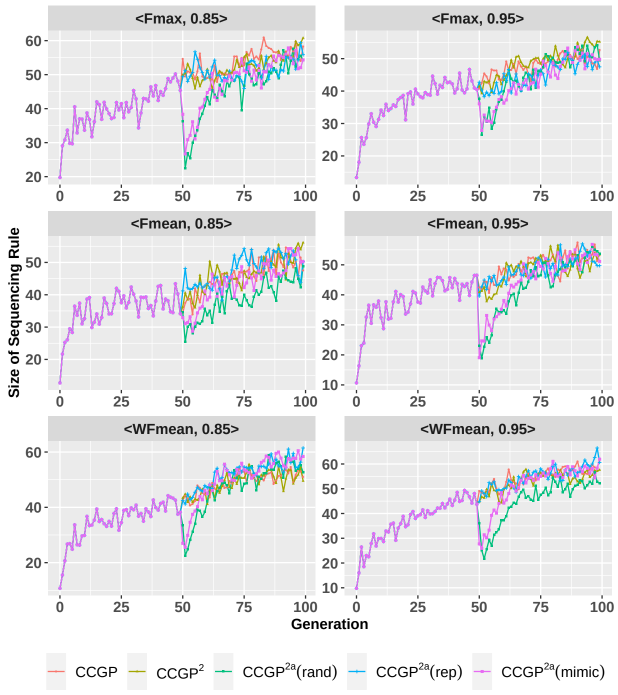


Fig. 8. The curves of *the best sequencing rule sizes* of the population of the five algorithms according to 30 independent runs in different scenarios.

TABLE IV
THE MEAN (STANDARD DEVIATION) OF **THE RULE SIZES** OBTAINED BY CCGP, CCGP^2 AND CCGP^{2a}(mimic) OVER 30 INDEPENDENT RUNS FOR SIX DYNAMIC FLEXIBLE SCENARIOS.

Scenario	CCGP	CCGP^2	CCGP^{2a}(mimic)
<Fmax,0.85>	122.07(30.60)	123.80(26.72)	112.60(25.75)
<Fmax,0.95>	117.27(25.22)	117.27(28.72)	112.27(31.38)
<Fmean,0.85>	115.73(24.91)	125.07(18.18)	108.93(26.32)
<Fmean,0.95>	121.27(17.60)	118.53(28.08)	110.47(22.31)
<WFmean,0.85>	116.87(23.88)	115.67(26.53)	115.73(26.31)
<WFmean,0.95>	127.40(24.40)	125.67(23.84)	121.47(22.80)

jobs). Compared with the curves of the sizes of sequencing rules, the proposed individual adaptation strategies have a great impact on the size of routing rules. The sizes of the best sequencing rules obtained by CCGP^{2a}(mimic) is similar with that of other algorithms.

Note that routing rule and sequencing rule work together in DFJSS. It makes sense to *take a routing rule and a sequencing rule as a pair to measure the rule size*. This is because a smaller routing rule and a larger sequencing can have the same ability for DFJSS compared with a larger routing rule and a smaller sequencing rule based on our observation. Based on the analyses as mentioned easier, it turns out that CCGP^{2a}(mimic) can achieve similar performance with small rule size. The rule sizes of only three algorithms (e.g., CCGP, CCGP^2, and CCGP^{2a}(mimic)) are further compared.

Table IV shows the mean and standard deviation of the rule sizes (i.e., routing rule plus sequencing rule) evolved by CCGP, CCGP^2, and CCGP^{2a}(mimic) according to 30 independent runs. It shows that the rule sizes obtained by these three algorithms are similar. The main difference of the evolved rules is that the rules evolved by CCGP^{2a}(mimic)) only contain selected features (i.e., fewer features) while the rules evolved by CCGP and CCGP^2 consist of all features possibly. The rule sizes are further studied in next subsection.

C. Unique Feature Analysis

The number of unique features in the rules is one of the indicators of the complexity of evolved rules. The unique feature means the feature that is needed to construct the rules. The smaller the number, the easier to interpret the rules.

Table V shows the mean (standard deviation) of the number of unique features in routing rules in different scenarios. There is no statistical difference between CCGP and CCGP^2 related to the number of unique features in routing rules in all scenarios. This means that applying selected features only to mutation is not an effective way for improving the interpretability of evolved routing rules. In addition, no matter what kind of individual adaptation strategy is used (i.e., CCGP^{2a}(mimic) with mimicking behaviour strategy, CCGP^{2a}(rep) with replacing by one strategy and CCGP^{2a}(rand) with 100% randomly initialisation strategy), the number of unique features in routing rules are significantly smaller than its counterpart.

Table VI shows the mean (standard deviation) of the number of unique features in sequencing rules in different scenarios. For all the algorithms (e.g., CCGP^{2a}(mimic), CCGP^{2a}(rep), CCGP^{2a}(rand), and CCGP^2) that involve feature selection, the number of unique features in sequencing rules is significantly smaller in all scenarios, especially the individual adaptation strategies related algorithms (e.g., CCGP^{2a}(mimic), CCGP^{2a}(rep), and CCGP^{2a}(rand)).

Assuming that the rules evolved by CCGP^{2a}(mimic)) *with fewer features are easier to be simplified by the algebraic*

TABLE V
THE MEAN (STANDARD DEVIATION) OF **THE AVERAGE NUMBER OF UNIQUE FEATURES OF ROUTING RULES** OBTAINED BY THE FIVE ALGORITHMS OVER 30 INDEPENDENT RUNS FOR SIX DYNAMIC FLEXIBLE SCENARIOS.

Scenario	CCGP	CCGP^2	CCGP^{2a}(rand)	CCGP^{2a}(rep)	CCGP^{2a}(mimic)
<Fmax,0.85>	8.40(1.33)	7.80(1.47)	6.27(1.68)(-)	6.57(1.74)(-)	6.67(1.81)(-)
<Fmax,0.95>	8.67(0.92)	8.37(1.33)	6.73(1.72)(-)	6.83(1.64)(-)	6.70(1.66)(-)
<Fmean,0.85>	8.03(1.03)	7.67(1.03)	5.70(1.62)(-)	5.83(1.46)(-)	5.63(1.52)(-)
<Fmean,0.95>	8.40(1.10)	7.87(1.14)	5.57(1.33)(-)	5.73(1.53)(-)	5.83(1.56)(-)
<WFmean,0.85>	8.27(1.23)	7.93(1.23)	5.60(1.63)(-)	6.17(2.09)(-)	5.70(1.99)(-)
<WFmean,0.95>	8.20(1.13)	7.50(1.57)	5.70(1.47)(-)	5.90(1.73)(-)	5.70(1.99)(-)

TABLE VI
THE MEAN (STANDARD DEVIATION) OF **THE AVERAGE NUMBER OF UNIQUE FEATURES OF SEQUENCING RULES** OBTAINED BY THE FIVE ALGORITHMS OVER 30 INDEPENDENT RUNS FOR SIX DYNAMIC FLEXIBLE SCENARIOS.

Scenario	CCGP	CCGP^2	CCGP^{2a}(rand)	CCGP^{2a}(rep)	CCGP^{2a}(mimic)
<Fmax,0.85>	7.13(1.59)	6.53(1.14)(-)	5.23(1.41)(-)	5.00(1.20)(-)	5.20(1.27)(-)
<Fmax,0.95>	7.40(1.57)	6.53(1.41)(-)	4.97(1.25)(-)	5.03(1.27)(-)	5.17(1.39)(-)
<Fmean,0.85>	6.57(2.10)	5.47(1.36)(-)	3.53(1.07)(-)	3.40(1.00)(-)	3.70(1.06)(-)
<Fmean,0.95>	6.90(1.60)	6.03(1.43)(-)	4.00(1.23)(-)	3.97(1.19)(-)	3.70(0.99)(-)
<WFmean,0.85>	6.53(1.59)	5.17(1.05)(-)	4.00(0.79)(-)	3.93(0.69)(-)	4.00(0.79)(-)
<WFmean,0.95>	6.80(1.52)	5.70(1.47)(-)	4.33(0.92)(-)	4.17(0.95)(-)	4.27(0.87)(-)

TABLE VII
THE MEAN(STANDARD DEVIATION) OF **TRAINING TIME** (MINUTES) BY THE FIVE ALGORITHMS IN SIX DYNAMIC FLEXIBLE SCENARIOS.

Scenario	CCGP	CCGP^2	CCGP^{2a}(rand)	CCGP^{2a}(rep)	CCGP^{2a}(mimic)
<Fmax,0.85>	100(17)	96(14)(=)	80(15)(-)	83(13)(-)	75(10)(-)
<Fmax,0.95>	107(15)	111(20)(=)	88(13)(-)	92(15)(-)	88(12)(-)
<Fmean,0.85>	98(12)	99(13)(=)	78(11)(-)	87(12)(-)	77(12)(-)
<Fmean,0.95>	109(15)	108(16)(=)	86(11)(-)	94(16)(-)	85(12)(-)
<WFmean,0.85>	94(11)	98(11)(=)	82(9)(-)	88(15)(=)	83(15)(-)
<WFmean,0.95>	113(16)	109(16)(=)	90(13)(-)	98(16)(-)	88(13)(-)

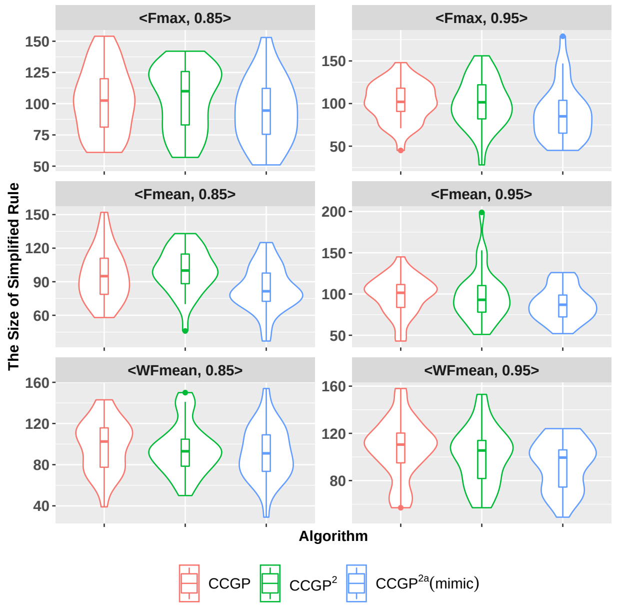


Fig. 9. The violin plot of rule sizes (i.e., routing rule plus sequencing rule) obtained by CCGP, CCGP^2, and CCGP^{2a}(mimic) after simplification over 30 independent runs in six different scenarios.

transformation. Simplification aims to simplify the complicated expression by some kinds of algebraic operations to make the evolved rules easier to be interpreted. For example, given a rule (A - B) / (A - B), it will be simplified as one. Moreover, the rule A + B + A + A will become 3 * A + B while the rule A * B / B will be simplified as A.

Fig. 9 shows the violin plot of rule size taking routing and sequencing rules as a pair obtained by CCGP, CCGP^2, and CCGP^{2a}(mimic) over 30 independent runs in six scenarios. It shows that the rule sizes of simplified rules evolved by CCGP^{2a}(mimic) are much smaller than that of CCGP and CCGP^2. It indicates that CCGP^{2a}(mimic) has more potential to get smaller rules which are important for interpreting rules. Note that the components produced by basic functions (e.g., +, -, *, /) are easier to be simplified than that of generated by max and min.

Fig. 10 shows the scatter plot of the rule sizes of routing and sequencing rules before and after simplification. It shows that the routing rule sizes become much smaller (i.e., move to a lower position) in all scenarios. In addition, the sequencing rule sizes tend to be smaller (i.e., move to left position) compared with the sizes without simplification in all scenarios.

In general, after simplification, CCGP^{2a}(mimic) can achieve smaller size routing rules and sequencing rules than that of CCGP and CCGP^2.

D. Training Time

Training time is an important criterion to measure the efficiency of algorithms. Table VII shows the training time of CCGP, CCGP^2, CCGP^{2a}(rand), CCGP^{2a}(rep), and CCGP^{2a}(mimic). The training time of CCGP^2 has no significant difference compared with CCGP^2. It is obvious that the training time of CCGP^{2a}(rand), CCGP^{2a}(rep) (i.e., in five

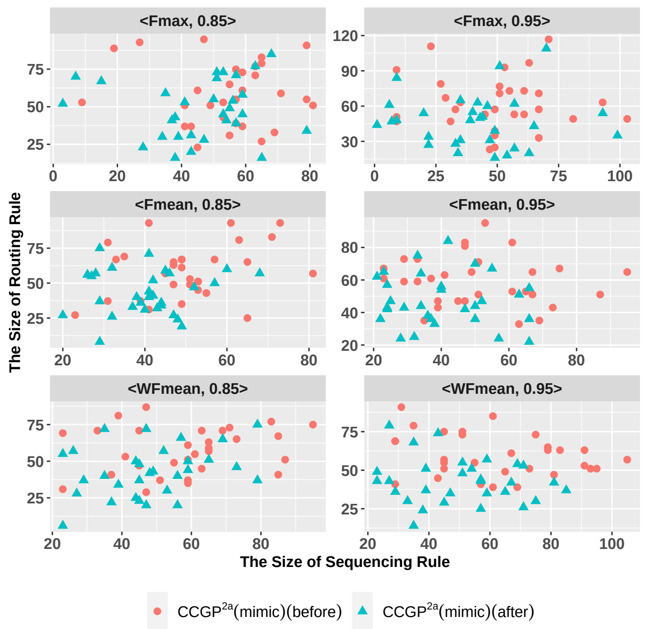


Fig. 10. The scatter plot of the rule sizes of routing rules and sequencing rules before and after simplification obtained by CCGP^{2a}(mimic) over 30 independent runs in six different scenarios.

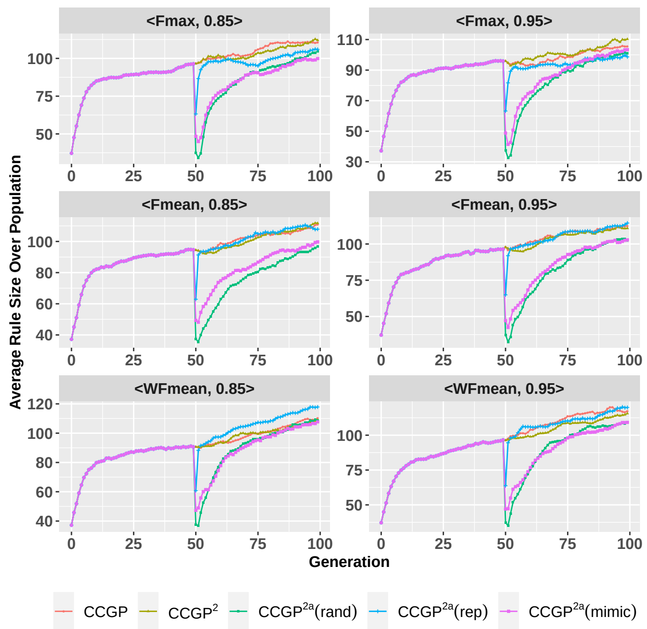


Fig. 11. The *average rule (routing rules plus sequencing rules) sizes over population* of the five algorithms in different scenarios.

scenarios), and CCGP^{2a}(mimic) decrease dramatically compared with that of CCGP and CCGP^2. The main difference of training time between them is caused by the individual adaptation strategy. Intuitively, for CCGP^{2a}(rand), CCGP^{2a}(rep) and CCGP^{2a}(mimic), more training time might be needed due to the extra algorithm operators. However, it turns out that the training time of individual adaptation strategies related algorithms are smaller than that of CCGP and CCGP^2, especially CCGP^{2a}(rand) and CCGP^{2a}(mimic).

When looking at the curve of average rule size in the population in Fig. 11, the average rule sizes over the population of CCGP^{2a}(rand) and CCGP^{2a}(mimic) are smaller than its counterparts. The reason is that all the individuals in the population at generation 50 are re-initialised and they have smaller sizes. The most computationally expensive part is the evaluation and smaller rules tend to save computational time. Thus, CCGP^{2a}(rand) and CCGP^{2a}(mimic) can reduce the computational time significantly.

In general, it is noted that CCGP^{2a}(mimic) is more efficient even more algorithm operators (e.g., feature selection and mimicking individuals' behaviours operations in the algorithm) are involved.

VI. Further Analyses

A. Feature Analysis

Fig. 12 and Fig. 13 shows the selected and unselected features of 30 runs in both sequencing rules and routing rules in six different scenarios, respectively. For a feature, a bigger blue area (the larger frequency the corresponding feature is selected) means the corresponding feature is more important. It is noted that the selected features of CCGP^2, CCGP^{2a}(rand), CCGP^{2a}(rep), and CCGP^{2a}(mimic) are the same in the same run (i.e., with same random value) since the evolutionary processes are the same in stage 1. In each run, both for sequencing rules and routing rules, the selected features vary from each other based on the proposed feature selection algorithm. This means that the selected features can be adjusted adaptively with the proposed two-stage framework and that the selected features are based on the specific problems (i.e., different runs).

For sequencing rules, the top three important features for scenario <Fmax,0.85> are PT, TIS, and WKR, as shown in Fig. 12. Except for these three features, NIQ and NOR are also important in scenario <Fmax,0.95>. Compared with <Fmax,0.85>, Fig. 12 shows that more features are selected in scenario <Fmax,0.95>. It might be because a higher utilisation level makes the problem more difficult to be optimised. For minimising mean flowtime (e.g., <Fmean,0.85> and <Fmean,0.95>), PT and WKR play an important role (i.e., they are selected in all 30 runs). When taking the mean weighted flowtime into consideration (e.g., <WFmean,0.85> and <WFmean,0.95>), except for PT and WKR, W is also a significant feature. It is consistent with our intuition that PT (i.e., processing time) and WKR (i.e., median amount of work remaining for a job) are important factors for flowtime related objectives. In addition, W is often chosen for minimising mean weighted flowtime rather than max flowtime and mean flowtime. It is consistent with our expectation, since the calculations of mean flowtime and max flowtime do not involve W at all.

For routing rules, three features which are MWT, OWT, and WIQ, are significant for evolving routing rules in all scenarios, as shown in Fig. 13. It is consistent with our intuition that the machine which has less workload (WIQ) and earlier ready time (MWT) is preferred, since the new operation has a higher chance to be processed early. In addition, this paper will allocate a new operation once it becomes a ready operation

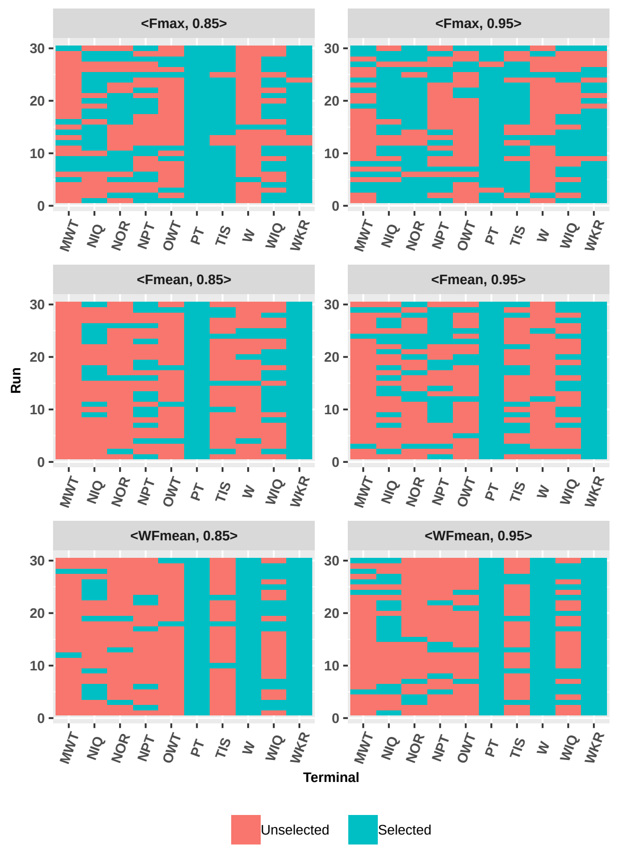


Fig. 12. Selected and unselected features of *sequencing rules* of 30 independent runs in different scenarios.

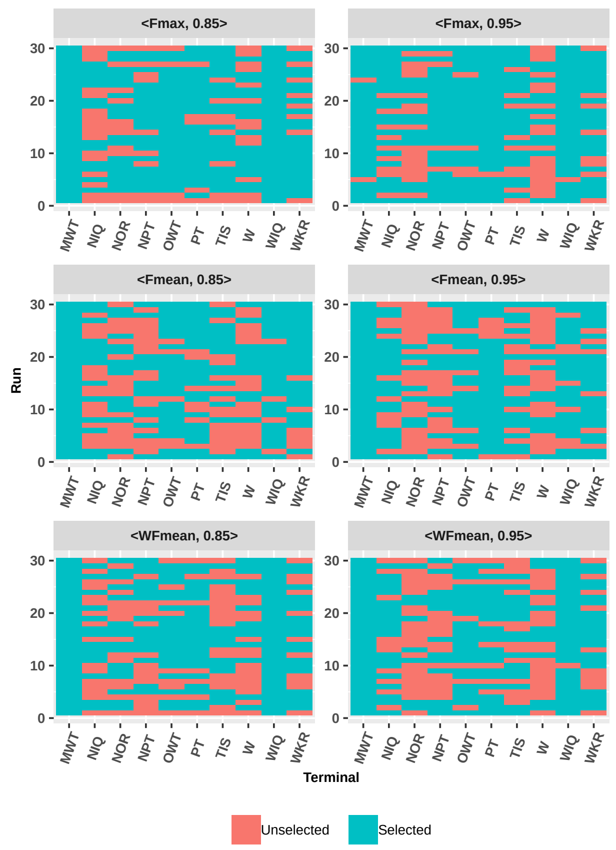


Fig. 13. Selected and unselected features of *routing rules* of 30 independent runs in different scenarios.

(i.e., OWT, the operation waiting time equals zero). OWT is naturally a very important factor for the routing process (i.e., kind of indicator). Compared with the selected features in sequencing rules, more features are used in routing rules (i.e., more blue areas). It indicates that the evolved routing rules might be more complex than sequencing rules.

Fig. 14 shows the distributions of test objective values of the 30 independent runs of CCGP^{2a}(mimic) in scenario <Fmax,0.85>, categorised by whether each feature is selected or not in sequencing rules. TIS is not selected in three runs, the test performance is much worse when it is selected. However, even NIQ is a relatively important feature (i.e., it is selected in half runs roughly) and it is not selected in half runs, the test performance is still very good. It is interesting that WKR is not selected in three runs, however, WKR has a greater impact on two runs while it has no effect on one run.

B. Rule Analysis

This paper takes the best routing rule and the corresponding sequencing rule (i.e., its objective value is 1096.02) achieved by CCGP^{2a}(mimic) and its corresponding best rules (i.e., the run with the same random seed) obtained by CCGP^2 (i.e., its objective value is 1099.61) in scenario <WFmean,0.95> as an example. This is because the objective in scenario <WFmean,0.95> is more difficult to be optimised than other scenarios. It is noted that the smaller the values calculated by the rules, the more priority the machine or operation has.

For CCGP^2, in the examined run, the routing terminal set consists of NIQ, WIQ, MWT, PT, NPT, OWT, WKR, NOR, and W (i.e., **nine** features). There are **six** features which are NIQ, WIQ, MWT, PT, WKR, and W, are selected as sequencing terminal set. For the routing rules obtained by CCGP^{2a}(mimic), all the features in the routing terminal set (i.e., **seven** features, NIQ, WIQ, MWT, PT, OWT, WKR, and W) are used. When further looking into the sequencing rule obtained by CCGP^{2a}(mimic), **five** of them (e.g., NIQ, WIQ, PT, WKR, and W) are involved. Fewer features are used by CCGP^{2a}(mimic) compared with CCGP^2.

However, the evolved scheduling heuristics obtained by CCGP^{2a}(mimic) have better test performance than that of CCGP^2. The routing rule obtained by CCGP^{2a}(mimic) can be simplified, as shown in Eq. (2).

$$\begin{aligned} R_1 =& min\{2*NIQ, max(\frac{NIQ*PT}{MWT}, \\ & PT*WIQ*min(WIQ,WKR))\}+ \\ & NIQ*\frac{PT}{W}-MWT \end{aligned} \tag{2}$$

It is obvious that this routing rule is quite small after simplification. In terms of the features related to machines, this routing rule prefers to choose the machine which has a large waiting time (i.e., MWT) and a smaller NIQ (i.e., the number of operations in the queue) and WIQ (i.e., the workload of machines). In terms of the features of operations, this routing rule tends to choose the machine which can process an operation more efficient with a smaller PT (i.e., processing

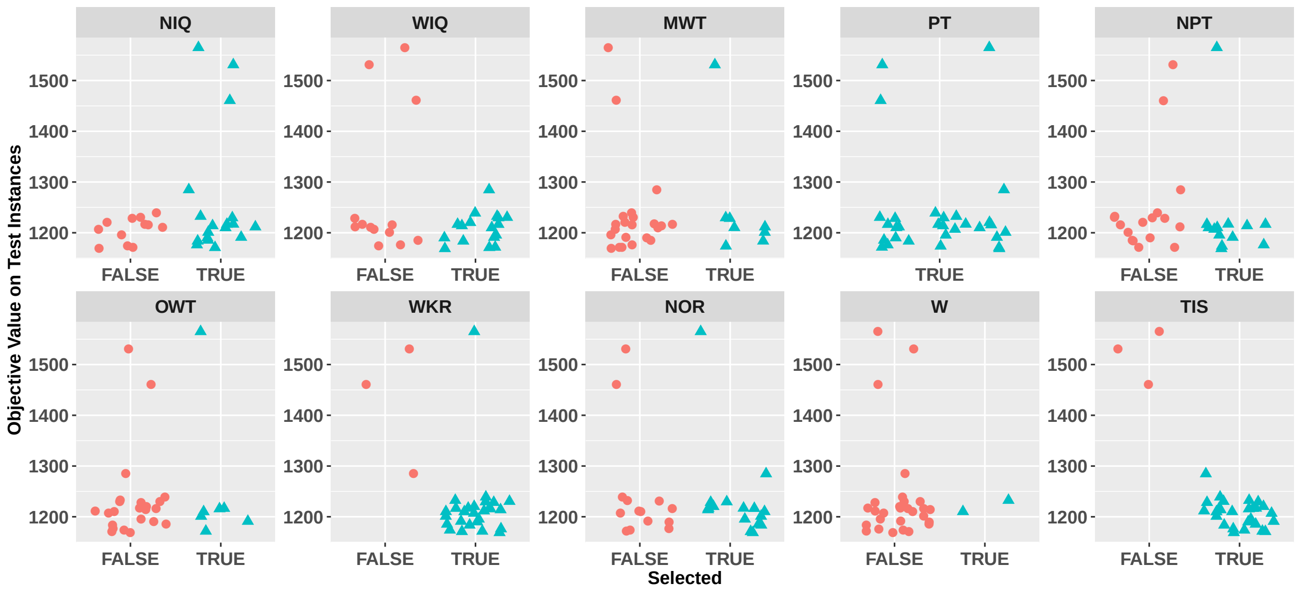


Fig. 14. The distributions of the test objective values of the 30 independent runs of CCGP^{2a}(mimic) in scenario <Fmax,0.85>, categorised by whether each feature is selected or not in sequencing rules.

time). In addition, from the perspective of operation, W (i.e., weight that used to indicate the importance of operation) is a constant when choosing a machine. From a machine's point of view, the operation which is more important with a larger W has more priority to select a machine.

The corresponding sequencing rule obtained by CCGP^{2a}(mimic) is simplified, as shown in Eq. (3).

$$\begin{aligned} S_1 =&\frac{PT+WKR}{W} \\ &-\frac{W}{PT}(W*WIQ-W+WIQ)-\frac{W}{PT}* \\ &(W*WIQ+WKR+\frac{PT+WKR}{W*WIQ-W+WKR}) \end{aligned} \tag{3}$$

It is noted that for all the operation in the queue of a machine, the machine-related feature such as WIQ (i.e., the workload of a machine) is the same for all operations. This means that it is not a vital important feature. This sequencing rule prefers to choose the operation with smaller PT (i.e., processing time) and larger W (i.e., weight, the importance of an operation). It is interesting that this sequencing rule prefers to smaller WKR (i.e., median amount of work remaining for a job) based on the first line of S_1 while it tends to select the operations with larger WKR based on the third line of S_1 partially. It indicates that although we can interpret the rules to some extent, it is still hard to completely understand the behaviour of rules. We will continue to work on this topic in the future.

The corresponding routing and sequencing rules obtained by CCGP^2, can be simplified as shown in Eq. (4) and Eq. (5), respectively. The structures of both this routing rule and sequencing rule are more complex than that of CCGP^{2a}(mimic) even after simplification. From the perspective of the components, the function max and min are used a lot and nested inside each other. It is hard to know which component has played a real role since it relies on multiple factors. It is not easy to understand it from a human perspective.

$$\begin{aligned} R_2 =&max\{NIQ^2,(NIQ+NPT)*min(NIQ,NOR)\} \\ &-min(MWT,\frac{WKR}{MWT*WKR-1}) \\ &*max\{WKR,-MWT*WKR+ \\ &\frac{NIQ*PT*max\{WIQ,\frac{min(MWT,PT)}{(W+WKR)}\}}{max\{NIQ^2,\frac{NOR-W+WKR}{W},\frac{NIQ+NPT}{(min(NIQ,NOR))^{-1}}\}}\} \end{aligned} \tag{4}$$

$$\begin{aligned} S_2 =&NIQ(PT-W)(PT+\frac{WKR}{W}*max\{PT,\frac{WIQ}{W}, \\ &\frac{max\{WIQ,\frac{WKR}{W}\}}{W}\})+max\{\frac{WIQ}{W^2},\frac{NIQ}{W^2}+WIQ\} \\ &*(MWT+W+max\{\frac{WKR}{W^2},NIQ-1\}) \\ &*\frac{max\{WIQ,\frac{WKR}{W}\}}{W} \end{aligned} \tag{5}$$

In general, the evolved scheduling heuristics obtained by CCGP^{2a}(mimic) are more interpretable and effective with a smaller number of unique features and small rule sizes.

VII. Conclusions and Future Work

The goal of this paper is to develop an effective feature selection algorithm for evolving more interpretable rules for DFJSS automatically via GPHH without compromising any performance. The goal was achieved by proposing a two-stage framework with novel individual adaptation strategies that can utilise the information of the selected features and the investigated individuals in the feature selection process well.

The results shows that the evolved rules by CCGP^{2a}(mimic) have better interpretability due to fewer unique features in the rules and smaller rule sizes by introducing feature selection. This is also verified by the semantic analyses of the routing and sequencing rules evolved by CCGP^{2a}(mimic). In terms of training time, CCGP^{2a}(mimic) is more effective than that of the baseline algorithm by reducing the average rule size over population. Besides, the smaller rules achieved by $\mathrm{CCGP}^{2a}(\mathrm{mimic})$ can respond to scheduling needs faster in real time due to its less complexity. In general, the proposed algorithm $\mathrm{CCGP}^{2a}(\mathrm{mimic})$ can efficiently evolve more interpretable rules automatically without comprising any performance.

Some interesting directions can be further investigated in the near future. Since the unclear question of interpretability, more promising measurements for evaluating the interpretability of obtained scheduling heuristics by GP are worth to be studied. In addition, this work already shows the ability to obtain similar performance only with selected features. We would like to find more promising ways by local search to improve its performance further.

Acknowledgment

This work was supported in part by the Marsden Fund of New Zealand Government under Contracts VUW1509 and VUW1614, the Science for Technological Innovation Challenge (SfTI) fund under grant E3603/2903, and the MBIE SSIF Fund under Contract VUW RTVU1914. This work of Fangfang Zhang was supported by China Scholarship Council (CSC)/Victoria University Scholarship.

References

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

[2] C. D. Geiger, R. Uzsoy, and H. Aytuğ, "Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach," *Journal of Scheduling*, vol. 9, no. 1, pp. 7–34, 2006.

[3] 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.

[4] S. Bennett, S. Nguyen, and M. Zhang, "A hybrid discrete particle swarm optimisation method for grid computation scheduling," in *Proceedings of the Congress on Evolutionary Computation*. IEEE, 2014, pp. 483–490.

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

[6] D. Yska, Y. Mei, and M. Zhang, "Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2018, pp. 306–321.

[7] 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.

[8] J. Xiong, L.-n. Xing, and Y.-w. Chen, "Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns," *International Journal of Production Economics*, vol. 141, no. 1, pp. 112–126, 2013.

[9] J. Park, Y. Mei, S. Nguyen, G. Chen, and M. Zhang, "Investigating a machine breakdown genetic programming approach for dynamic job shop scheduling," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2018, pp. 253–270.

[10] Y. N. Sotskov and N. V. Shakhlevich, "Np-hardness of shop-scheduling problems with three jobs," *Discrete Applied Mathematics*, vol. 59, no. 3, pp. 237–266, 1995.

[11] H. Chen, C. Chu, and J. Proth, "An improvement of the lagrangean relaxation approach for job shop scheduling: a dynamic programming method," *IEEE Transactions on Robotics and Automation*, vol. 14, no. 5, pp. 786–795, 1998.

[12] F. Y.-P. Simon *et al.*, "Integer linear programming neural networks for job-shop scheduling," in *Proceedings of the IEEE International Conference on Neural Networks*. IEEE, 1988, pp. 341–348.

[13] P. J. Van Laarhoven and E. H. Aarts, "Simulated annealing," in *Simulated annealing: Theory and applications*. Springer, 1987, pp. 7–15.

[14] F. Glover and M. Laguna, "Tabu search," in *Handbook of Combinatorial Optimization*. Springer, 1998, pp. 2093–2229.

[15] G. V. Conroy, "Handbook of genetic algorithms," *The Knowledge Engineering Review*, vol. 6, no. 4, pp. 363–365, 1991.

[16] J. Kennedy, "Particle swarm optimization," *Encyclopedia of Machine Learning*, pp. 760–766, 2010.

[17] K. Chen, F. Zhou, and B. Xue, "Particle swarm optimization for feature selection with adaptive mechanism and new updating strategy," in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 419–431.

[18] 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.

[19] K. Miyashita, "Job-shop scheduling with genetic programming," in *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 2000, pp. 505–512.

[20] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Genetic programming for evolving due-date assignment models in job shop environments," *Evolutionary Computation*, vol. 22, no. 1, pp. 105–138, 2014.

[21] 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.

[22] L. H. Gilpin, D. Bau, B. Z. Yuan, A. Bajwa, M. Specter, and L. Kagal, "Explaining explanations: An overview of interpretability of machine learning," in *Proceedings of the International Conference on Data Science and Advanced Analytics*. IEEE, 2018, pp. 80–89.

[23] I. Guyon and A. Elisseeff, "An introduction to variable and feature selection," *Journal of Machine Learning Research*, vol. 3, pp. 1157–1182, 2003.

[24] M. Dash and H. Liu, "Feature selection for classification," *Intelligent Data Analysis*, vol. 1, no. 3, pp. 131–156, 1997.

[25] A. K. Uysal, "An improved global feature selection scheme for text classification," *Expert Systems with Applications*, vol. 43, pp. 82–92, 2016.

[26] B. Xue, M. Zhang, W. N. Browne, and X. Yao, "A survey on evolutionary computation approaches to feature selection," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 4, pp. 606–626, 2016.

[27] A. Lensen, B. Xue, and M. Zhang, "Particle swarm optimisation representations for simultaneous clustering and feature selection," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*, 2016, pp. 1–8.

[28] Q. Chen, M. Zhang, and B. Xue, "Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 5, pp. 792–806, 2017.

[29] Q. U. Ain, B. Xue, H. Al-Sahaf, and M. Zhang, "Genetic programming for feature selection and feature construction in skin cancer image classification," in *Proceedings of the Pacific Rim International Conference on Artificial Intelligence*, 2018, pp. 732–745.

[30] R. Hunt, *Genetic Programming Hyper-heuristics for Job Shop Scheduling*. Victoria University of Wellington, 2016.

[31] Y. Mei, M. Zhang, and S. Nguyen, "Feature selection in evolving job shop dispatching rules with genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2016, pp. 365–372.

[32] Y. Mei, S. Nguyen, B. Xue, and M. Zhang, "An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 5, pp. 339–353, 2017.

[33] F. Zhang, Y. Mei, and M. Zhang, "A two-stage genetic programming hyper-heuristic approach with feature selection for dynamic flexible job shop scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference*. IEEE, 2019, pp. 347–355.

[34] S. Gu, R. Cheng, and Y. Jin, "Feature selection for high-dimensional classification using a competitive swarm optimizer," *Soft Computing*, vol. 22, no. 3, pp. 811–822, 2018.

[35] G. I. Sayed, A. E. Hassanien, and A. T. Azar, "Feature selection via a novel chaotic crow search algorithm," *Neural Computing and Applications*, vol. 31, no. 1, pp. 171–188, 2019.

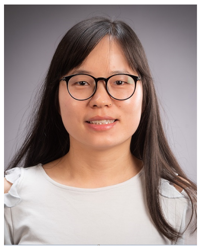
[36] J. Branke, S. Nguyen, C. W. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 1, pp. 110–124, 2016.

[37] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Computational Intelligence*. Springer, 2009, pp. 177–201.

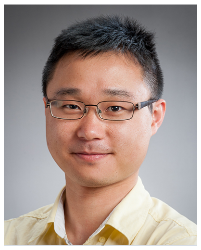
[38] E. K. Burke, M. R. Hyde, G. Kendall, and J. R. Woodward, "A genetic programming hyper-heuristic approach for evolving 2-d strip packing heuristics," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 6, pp. 942–958, 2010.

[39] M. B. Bader-El-Den, R. Poli, and S. Fatima, "Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework," *Memetic Computing*, vol. 1, no. 3, pp. 205–219, 2009.

[40] 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 Aritficial Intelligence*, 2007, pp. 223–234.
[41] M. A. Ardeh, Y. Mei, and M. Zhang, “Genetic programming hyper-heuristic with knowledge transfer for uncertain capacitated arc routing problem,” in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2019, pp. 334–335.
[42] 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.
[43] 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.
[44] M. Durasevic and D. Jakobovic, “Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment,” *Genetic Programming and Evolvable Machines*, vol. 19, no. 1-2, pp. 9–51, 2018.
[45] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, “Guided subtree selection for genetic operators in genetic programming for dynamic flexible job shop scheduling,” in *Proceedings of the European Conference on Genetic Programming*. Springer, 2020, pp. 262–278.
[46] R. Poli, W. B. Langdon, N. F. McPhee, and J. R. Koza, *A field guide to genetic programming*. Lulu. com, 2008.
[47] K. Chen, F. Zhou, and X. Yuan, “Hybrid particle swarm optimization with spiral-shaped mechanism for feature selection,” *Expert Systems with Applications*, vol. 128, pp. 140–156, 2019.
[48] S. Jurado, À. Nebot, F. Mugica, and N. Avellana, “Hybrid methodologies for electricity load forecasting: Entropy-based feature selection with machine learning and soft computing techniques,” *Energy*, vol. 86, pp. 276–291, 2015.
[49] R. Saidi, W. Bouaguel, and N. Essoussi, “Hybrid feature selection method based on the genetic algorithm and pearson correlation coefficient,” in *Machine Learning Paradigms: Theory and Application*, 2019, pp. 3–24.
[50] Q. Chen, M. Zhang, and B. Xue, “Structural risk minimization-driven genetic programming for enhancing generalization in symbolic regression,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 703–717, 2019.
[51] K. Nag and N. R. Pal, “A multiobjective genetic programming-based ensemble for simultaneous feature selection and classification,” *IEEE Transactions on Cybernetics*, vol. 46, no. 2, pp. 499–510, 2016.
[52] S. Nguyen, M. Zhang, and K. C. Tan, “Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules,” *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2951–2965, 2017.
[53] T. Hildebrandt and J. Branke, “On using surrogates with genetic programming,” *Evolutionary Computation*, vol. 23, no. 3, pp. 343–367, 2015.
[54] J. P. Davis, K. M. Eisenhardt, and C. B. Bingham, “Developing theory through simulation methods,” *Academy of Management Review*, vol. 32, no. 2, pp. 480–499, 2007.
[55] F. Zhang, Y. Mei, and M. Zhang, “Surrogate-assisted genetic programming for dynamic flexible job shop scheduling,” in *Proceedings of the Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 766–772.

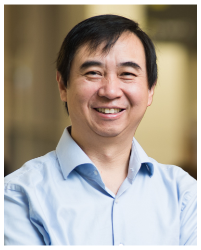


Fangfang Zhang (Student Member, IEEE) received the B.Sc. and M.Sc. degrees from Shenzhen University, Shenzhen, China, in 2014 and 2017, respectively. She is currently pursuing the Ph.D. degree in computer science with the School of Engineering and Computer Science, Victoria University of Wellington, New Zealand. She has over 20 journal and conference papers. Her current research interests include evolutionary computation, hyper-heuristic, job shop scheduling, and multitasking optimisation. Miss. Zhang is a member of the IEEE Computational Intelligence Society and Association for Computing Machinery, and has been severing as reviewers for top international journals and conferences. She is also a committee member of the IEEE NZ Central Section.



Yi Mei (M'09-SM'18) is a Senior Lecturer at 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. He has more than 100 fully referred publications, including the top journals in EC and Operations Research such as IEEE TEVC, IEEE TCYB, Evolutionary Computation Journal, European Journal of Operational Research, ACM Transactions on Mathematical Software. He serves as a Vice-Chair of the IEEE CIS Emergent Technologies Technical Committee, and a member of Intelligent Systems Applications Technical Committee. He is an Editorial Board Member/Associate Editor of three International Journals, and a guest editor of a special issue of the Genetic Programming Evolvable Machine journal. He serves as a reviewer of over 30 international journals.

Su Nguyen (M'13) received his Ph.D. degree in Artificial Intelligence and Data Analytics from Victoria University of Wellington, New Zealand, in 2013. Nguyen is a Senior Research Fellow and Algorithm Lead at CDAC, La Trobe University, Australia. His expertise includes evolutionary computation (EC), simulation optimization, automated algorithm design, interfaces of AI/OR, and their applications in logistics, energy, and transportation. He has 70+ publications in top EC/OR peer-reviewed journals and conferences and his current research focuses on novel people-centric artificial intelligence to solve dynamic and uncertain planning tasks by combining the creativity of evolutionary computation and power of advanced machine learning algorithms. He was the chair (2014-2018) of IEEE task force on Evolutionary Scheduling and Combinatorial Optimisation and is a member of IEEE CIS Data Mining and Big Data technical committee. He delivered technical tutorials about EC and AI-based visualisation at Parallel Problem Solving from Nature Conference (2018) and IEEE World Congress on Computational Intelligence (2020). He served as an editorial member of Complex and Intelligence Systems and the guest editor of the special issue on “Automated Design and Adaption of Heuristics for Scheduling and Combinatorial Optimization” in Genetic Programming and Evolvable Machines journal.



Mengjie Zhang (M'04-SM'10-F'19) received the B.E. and M.E. degrees from Artificial Intelligence Re- search 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 Professor of Computer Science, Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) in the Faculty of Engineering. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multi-objective optimization, feature selection and reduction, job shop scheduling, and transfer learning. He has published over 500 research papers in refereed international journals and conferences. Prof. Zhang is a Fellow of Royal Society of New Zealand, a Fellow of IEEE, and an IEEE Distinguished Lecturer. He was the chair of the IEEE CIS Intelligent Systems and Applications Technical Committee, the chair for the IEEE CIS Emergent Technologies Technical Committee, the chair of Evolutionary Computation Technical Committee, and a member of the IEEE CIS Award Committee. He is a vice-chair of the Task Force on Evolutionary Computer Vision and Image Processing, and the founding chair of the IEEE Computational Intelligence Chapter in New Zealand. He is also a committee member of the IEEE NZ Central Section.