Phenotype and Genotype Based Sample Aware Surrogate-Assisted Genetic Programming in Dynamic Flexible Job Shop Scheduling

Luyao Zhu, *Student Member, IEEE,* Fangfang Zhang, *Member, IEEE,* Xiaodong Zhu, *Member, IEEE,* Ke Chen, *Member, IEEE,* and Mengjie Zhang, *Fellow, IEEE*

***Abstract*—Genetic programming (GP) has been widely applied to evolve scheduling heuristics for dynamic flexible job shop scheduling (DFJSS). However, the evaluation of GP individuals is computationally expensive, especially in large scale DFJSS scenarios. A k-nearest neighbor (KNN) based surrogate has been successfully used to reduce individual evaluation time for GP by predicting the fitness of an individual with the most similar sample in KNN. Particularly, the phenotypes of GP individuals have been utilised to generate samples for KNN based surrogates with a precondition that the fitness of individuals with the same phenotype is the same or similar. However, their real fitness may differ greatly due to different input decision situations for fitness calculations in DFJSS. Thus, only considering phenotypes of GP individuals to extract samples could decrease the accuracy of KNN surrogates. This paper proposes a novel KNN based surrogate assisted GP algorithm by considering both the phenotype and genotype of GP individuals to generate samples. Specifically, a genotypic characterisation based on terminal frequency is designed to measure the similarity of individual genotypes. The results show that with the same training time, the proposed algorithm can converge fast and achieve better scheduling heuristics than the state-of-the-art algorithm in most examined scenarios. With the same number of generations, the proposed algorithm can obtain comparable performance but only needs about one third training time of baseline GP. The effectiveness of the proposed algorithm is also verified from different aspects, e.g., relation between genotype correlation and fitness difference of individuals, and population diversity.**

***Index Terms*—Surrogate Samples, Genetic Programming, Dynamic Flexible Job Shop Scheduling, Genotype and Phenotype.**

I. Introduction

Job shop scheduling [1] is an important combinatorial optimisation, which aims to solve challenging and practical scheduling tasks in real life, such as resource allocation in grid computing [2] and operation processing in manufacturing [3]. Job shop scheduling involves the processing of a number of jobs, and each job comprises a series of operations that need to be completed by a set of machines, where each operation corresponds to a specific machine. The goal of job shop scheduling is to optimize production efficiency by allocating machine resources effectively. Flexible job shop scheduling [4], an extension of job shop scheduling, is more in line with practical production, as each operation can be carried out by several machines. Dynamic flexible job shop scheduling (DFJSS) [5] is a more complicated process than flexible job shop scheduling due to the complex interactions among multiple decision making processes in dynamic environments such as the continuous arrival of new jobs [6], [7]. In DFJSS, there are two decisions that need to be made simultaneously. The first decision concerns *machine assignment*, which consists of assigning a ready operation to a machine. The second decision pertains to *operation sequencing*, which involves choosing the next operation to be processed when a machine is idle.

DFJSS is an NP-hard problem [8]. *Exact optimisation approaches* such as integer linear programming [9] and dynamic programming [10] cannot solve it efficiently. *Approximate solution optimisation approaches* such as genetic algorithms [11], [12], can obtain effective solutions by finding the near-optimal solution. However, they are not able to tackle dynamic situations efficiently since the rescheduling process could take too long to react to decisions made in the moment. *Scheduling heuristics* [13], e.g., dispatching rules [14], [15], utilize priority functions to rank machines or operations at decision points to generate solutions for DFJSS efficiently. Specifically, in DFJSS, a scheduling heuristic comprises a *sequencing rule* to determine the order of operations and a *routing rule* to assign machines. However, the design of scheduling heuristics normally needs experts and plenty of time. Moreover, the manually designed scheduling heuristics are often *only* good in specific scenarios. Genetic programming (GP) [16], [17] has been widely used for automatic generation of scheduling heuristics in DFJSS [18]. However, the evaluations of GP individuals for DFJSS are normally based on simulations, which are time-consuming, especially in large scale job shop scene. Similar to species in the natural environment, the individuals of nature-inspired GP have both its phenotypes and genotypes [19], [20]. The phenotype of a GP individual normally connects closely its behaviour, e.g., indicated by phenotypic characterisation (PC) in DFJSS [21], [22]. The genotype of a GP individual normally refers to its structure including original genetic materials [23].

Surrogate models [24], [25] have been extensively applied to reduce computation cost in evolutionary algorithms

Luyao Zhu, Xiaodong Zhu and Ke Chen are with the School of Electrical and Information Engineering, Zhengzhou University, Zhengzhou 450001, China (e-mail: zhuluyao58@163.com, Zhu_xd@zzu.edu.cn, chenkezixf@zzu.edu.cn).

Fangfang Zhang, Mengjie Zhang are with the Evolutionary Computation and Machine Learning Research Group, School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: fangfang.zhang@ecs.vuw.ac.nz, mengjie.zhang@ecs.vuw.ac.nz).

including GP in DFJSS [26]. Unlike general evolutionary algorithms with vector-based representation, GP has a tree-based representation which makes it not straightforward to build surrogates. -nearest neighbour (KNN) based surrogates with PC have been successfully applied to GP in DFJSS for reducing the evaluation time of GP individuals due to its efficiency [26], [27]. The PC of a GP individual is a vector of ranks of machines and operations in a fixed number of job shop scenarios, which connects closely its behaviour for decision making [28]. Thus, individuals with the same PC tend to have same or similar fitness. The surrogate samples consist of PCs and corresponding fitness obtained by real evaluations of GP individuals. For predicting the fitness of one individual, according to the KNN algorithm, the fitness of the most similar PC in the surrogate will be the estimated fitness of a GP individual. It is clear that the samples are critical for the success of KNN based surrogates assisted GP in DFJSS.

Intuitively, the samples of KNN based surrogates that can cover diverse GP individual behaviours will have a better accuracy to predict fitness for individuals. However, the original KNN based surrogate is just selecting all individuals in the previous generation as samples without considering the information of GP individuals [27]. In terms of the studies of samples for building KNN based surrogates in GP for DFJSS, [28] proposed to add more samples across more generations. However, [28] only focuses on the number of samples without considering the quality of samples. Zhu et.al [29] proposed to group GP individuals with unique PCs, and then selected an individual (i.e., the smallest one) in each group for real evaluations to be a sample. Then, a surrogate will be built using selected samples to estimate fitness of the remaining individuals in the current generation. In this way, the training time can be significantly reduced due to the reduction of the number of truly evaluated individuals. However, this approach assumes that individuals with the same phenotype have similar fitness. According to our observations, there are still some individuals with the same phenotype but have quite different fitness values, since the decision situations for obtaining the PC is usually smaller than and different from the decision situations in the simulation for real fitness evaluations. Thus, an individual may not represent all individuals in one group, which can affect the accuracy of the surrogate. Note that it is not practical to use a large number of decision situations to get PCs of GP individuals for KNN surrogates, as this will significantly reduce the efficiency of KNN surrogates due to the expensive distance calculations with high dimensional PC vectors. In addition, it is nontrivial to select representative decision situations for generating PCs. A more advanced sample-aware surrogate-assisted GP for DFJSS is worth investigating.

The goal of this article is to improve upon the efficiency and effectiveness of the GP algorithm for automatically evolving scheduling heuristics in DFJSS by developing an effective surrogate sample selection strategy. The proposed algorithm is expected to both evolve promising scheduling heuristics and reduce the training time of GP for DFJSS. Specifically, the contributions of this paper are:

1) Instead of utilising phenotype information of GP individuals only, we have proposed to consider genotype as well to select samples for building surrogates. Specifically, we have designed an effective genotype measure by utilising the feature frequency of GP individuals, where genotype of a GP individual is represented by a vector with feature frequency information. Then, the Spearman correlation [30] is used to measure the similarity of genotypes between two individuals. If two individuals with the same phenotype have a high genotype correlation, they are more similar to have close fitness.
2) We proposed a novel GP algorithm to select representative samples for building KNN based surrogates in DFJSS by considering both the phenotypes and genotypes of GP individuals. Specifically, the individuals with the same PC value will be put into one group. Second, niching is adopted to subdivide individuals in promising groups into different niches based on the similarity of genotypes of individuals. Finally, the smallest individual in each niche is selected to be truly evaluated to form a sample for building surrogates. Thus, the samples in the surrogate contain individuals with diverse phenotypes and genotypes. The built surrogate will be utilized to estimate the fitness of unevaluated individuals in the current generation. In addition, the build surrogate will be used to detect promising groups for further splitting into niches in the next generation.
3) We have analysed the effectiveness and efficiency of the proposed algorithm in terms of the training time and the performance of learned scheduling heuristics. The effectiveness of the new sample selection strategy is also verified by analysing the number of extra evaluated individuals, the relation of genotype correlation and fitness difference, and fitness difference in the same and different niches. In addition, the proposed algorithm can successfully achieve smaller and effective scheduling heuristics for DFJSS. We also find that the proposed algorithm can maintain a proper population diversity to balance its exploitation and exploration ability during different evolutionary stages.

The rest of this article is organized as follows. Section II presents a literature review. Section III shows detailed descriptions of the proposed algorithm. The experimental design is provided in Section IV, whereas Section V contains the discussion and results. Section VI conducts further analyses. Finally, this article is concluded in Section VII.

II. Literature Review

A. Dynamic Flexible Job Shop Scheduling

The goal of DFJSS is to optimise the allocation of machine resources in a job shop. In DFJSS, there are m machines $\mathcal{M} = \{M_1, M_2, ..., M_m\}$ that are required to process n jobs $\mathcal{J} = \{J_1, J_2, ..., J_n\}$. Every job has a sequence of operations $\mathcal{O}_j = \{O_{j1}, O_{j2}, ..., O_{jl_j}\}$ that must be performed in a certain order. Each operation O_{ji} can be handled on several machines $M(O_{ji}) \subseteq \pi(O_{ji})$ [4]. Machine assignment and operation sequencing are required to be made concurrently under dynamic situations. This paper focuses on a common dynamic event, i.e., new job arrival [15]. The information

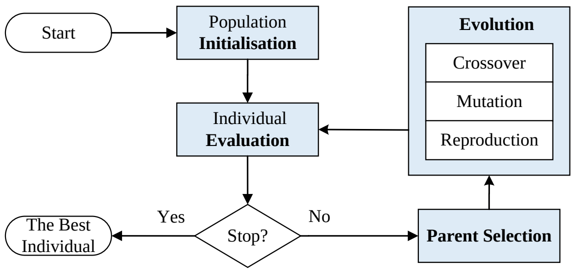

Fig. 1. The flowchart of GP.

regarding a new job remains unknown until it is delivered to the shop floor. The principal constraints in DFJSS are presented below.

- The processing of an operation cannot begin until all its preceding operations have been completed.
- Each operation is exclusive to one candidate machine for processing purposes.
- A machine cannot handle several operations at a time.
- Operations that have commenced processing cannot be interrupted or terminated until completion.

In this paper, we explore two commonly used objectives. Their calculations are demonstrated as follows.

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

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

B. Genetic Programming for DFJSS

GP has been successfully used to learn scheduling heuristics for JSS problems as a hyper-heuristic approach [31]–[36]. The success of GP is attributed to four main reasons. First, the flexible representation of GP can denote various scheduling heuristics for JSS. Different genotypes can represent scheduling heuristics with the same behavior, which provides a more diverse set of genetic materials to evolve promising scheduling heuristics. Second, the structures of heuristics (i.e., the optimal structure is normally unknown) are not required to be predefined. Third, GP programs [37]–[39] can be served as priority functions to rank machines or operations efficiently in DFJSS. Finally, the tree based scheduling heuristics tend to be easier to be interpreted, which is important for actual production scheduling.

Fig. 1 shows the flowchart of GP [40]. The main components in GP are initialisation, evaluation, parent selection and evolution. GP starts with a population of randomly generated individuals, whose goodness is measured by individual evaluation. While the stopping condition is not reached, offspring are generated with selected parents via parent selection by the genetic operators (i.e., crossover, mutation and reproduction) to form a new population. Otherwise, GP algorithm will output the best scheduling heuristic so far.

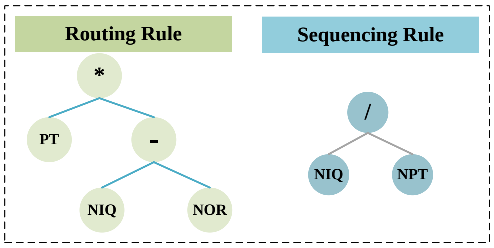

Fig. 2. An example of a GP individual with multi-tree representation for DFJSS.

TABLE I
AN EXAMPLE OF THE DECISION MAKING OF THE ROUTING RULE PT * (NIQ - NOR) AT A ROUTING DECISION POINT WITH THREE MACHINES.

Machine Number	(PT	Feature NIQ	NOR)	Priority Value	Chosen Machine
M_1	100	40	25	1500	
M_2	200	65	40	5000	M_3
M_3	150	30	25	**750**	

1) Representation: This paper applies multi-tree representation to learn scheduling heuristics for DFJSS [40]. Specifically, each GP individual consists of two trees, as shown in Fig. 2 (i.e., one is a routing rule, and the other is a sequencing rule). The fitness of an individual is determined by the cooperation between these two rules. The routing rule denotes a priority function of PT * (NIQ - NOR), PT refers to the processing time to operate on the specified machine, NIQ and NOR are the number of operations in the queue and operations left of a job, separately. The sequencing rule is NIQ / NPT, where NPT stands for the processing time of the subsequent operation.

2) Decision Marking: Assume a job can be handled by three candidate machines (i.e., M_1, M_2 and M_3) in Table I. Given the feature values of three machines, the priority values of M_1, M_2 and M_3 are 1500, 5000 and 750, respectively. According to the priority function values, M_3 will process the job (i.e., a smaller function value represents a higher priority in this paper). Similarly, the sequencing rule will select and process the operation with the highest priority on an idle machine.

C. Surrogate-Assisted GP for Job Shop Scheduling

1) Related Work: Surrogate-assisted evolutionary algorithm has been widely studied to deal with expensive fitness evaluations [24], [41]–[43] in the past decades. The fundamental idea of surrogates is to design a simple model to estimate the fitness of individuals instead of using the expensive real evaluations.

In terms of how surrogates are built to predict the fitness of individuals, we group the existing surrogate-assisted GP in job shop scheduling into two categories. One is to predict the fitness of individuals by finding the most similar sample in the KNN based surrogate model [44]. The other is to develop a simplified simulation model (i.e., the simulation situation with few machines and operations) as a surrogate model

TABLE II
AN ILLUSTRATION OF CALCULATING THE PHENOTYPIC CHARACTERISATION OF A ROUTING RULE WITH THREE DECISION SITUATIONS AND EACH WITH THREE CANDIDATE MACHINES.

Decision Situation	Rank Obtained by Reference Rule	Rank Obtained by a Routing Rule	PC_i
$1(M_1)$	$\underline{2}$	$\underline{1}$	
$1(M_2)$	1	3	**2**
$1(M_3)$	3	2	
$2(M_1)$	2	3	
$2(M_2)$	$\underline{3}$	$\underline{1}$	**3**
$2(M_3)$	1	2	
$3(M_1)$	2	3	
$3(M_2)$	3	2	**1**
$3(M_3)$	$\underline{1}$	$\underline{1}$	

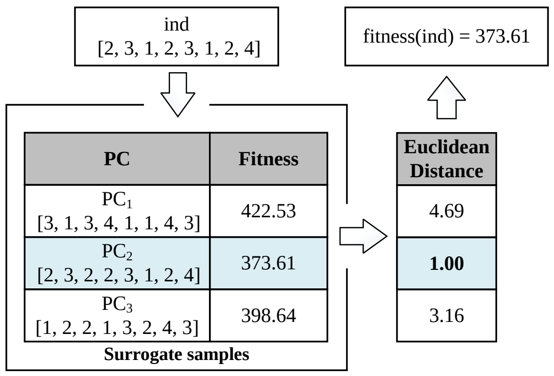
ind
[2, 3, 1, 2, 3, 1, 2, 4]
fitness(ind) = 373.61
PC
Fitness
PC1 [3, 1, 3, 4, 1, 1, 4, 3] 422.53
PC2 [2, 3, 2, 2, 3, 1, 2, 4] 373.61
PC3 [1, 2, 2, 1, 3, 2, 4, 3] 398.64
Surrogate samples
Euclidean Distance
4.69
1.00
3.16

Fig. 3. An illustration of using KNN based surrogate with PC for fitness prediction for GP individuals.

to predict the fitness of individuals [26], [45], [46]. KNN based surrogate models are more efficient than the surrogate models via simplified simulation since there are only simple calculations with the samples in KNN based surrogate models.

In terms of how the surrogates are used, there are two commonly used ways. First, most existing studies [27], [28], [45] focus on using surrogates as a preselection technique to hasten the convergence of GP by cheaply evaluating more individuals, and only the selected individuals will move to the next generation and get their real fitness. This way focuses on the effectiveness improvement with offspring preselection, and actually does not shorten the training time. Second, surrogates are used to directly reduce GP's training time in job shop scheduling [29], [46]. Various surrogates with different accuracy were designed and used at different generations to reduce the training time without sacrificing the performance of GP algorithm [46]. A KNN based sample aware algorithm was studied in [29] to not only reduce the training time but also improve the effectiveness of learned scheduling heuristics. However, [29] considers the phenotype of GP individuals only for choosing samples for KNN based surrogates which may limit its performance since GP individuals with the same phenotype can have quite different fitness.

2) Fitness Prediction with KNN in GP: This paper chooses the efficient KNN based surrogate as a baseline. The basic information of using KNN to estimate fitness of GP individuals is shown below.

a) Phenotypic Characterisation: Tree-based GP individuals can be efficiently represented as vectors through the use of PC for the construction of KNN based surrogates. [27]. A GP individual's PC is a vector that includes ranks of machines and operations in a set of routing and sequencing decision scenarios, representing the decision behaviour of a pair of scheduling rules. Specifically, an examined routing rule or sequencing rule will select the most prioritised machine or operation, and the value of PC's each dimension is the corresponding rank of the selected machines or operations by a reference rule. Table II shows an illustration of calculating the PC of a routing rule with three decision situations, and each decision situation comprises three candidate machines. In the first decision situation, M_1 obtains the highest priority according to the examined routing rule, and the rank of M_1 by a reference rule is 2. Thus, the first value of PC PC_1 is set to 2. Similarly, the PC values in other situations can be obtained. Finally, PC of the examined routing rule is $[2, 3, 1]$. For more details, readers can refer to [44], [47].

b) Fitness Prediction: Individuals with close PC vectors show a similar decision behaviour, which leads to comparable fitness. Fig. 3 illustrates how the KNN based surrogate with PC works. For a new individual ind whose PC is $[2, 3, 1, 2, 3, 1, 2, 4]$, KNN algorithm is applied to find the most similar PC (i.e., PC_2) in the surrogate samples with the smallest Euclidean distance, i.e., 1, and the fitness corresponding to PC_2, which is 373.61, will be the estimated fitness of ind.

From the above example, it is clear that the samples play a significant role in KNN based surrogate models with PC. The samples are expected to cover various behaviours in the population for the accuracy of estimating individuals. However, the studies on KNN based surrogate samples for DFJSS are still in their infancy. It is non-trivial to choose which samples to build the surrogate model. This paper will focus on the sample selection to select representative samples for surrogate-assisted GP in DFJSS.

III. PROPOSED PHENOTYPE AND GENOTYPE BASED SAMPLE AWARE SURROGATE-ASSISTED GP

A. Framework of the Proposed Algorithm

Fig. 4 shows the flowchart of the proposed surrogate-assisted GP algorithm. The core strategy for selecting surrogate samples by considering the phenotype and genotype of GP individuals of the proposed algorithm is highlighted in green. The key mechanism that can significantly reduce the training time is highlighted in blue, which is to take some selected individuals for real evaluations as samples to build surrogates and estimate the remaining individuals' fitness in the current generation. At *initialisation* stage, a population is randomly generated, and we evaluate all individuals at the *evaluation* stage. Then, a new population will be bred by genetic operators with selected parents during *evolution* stage. If the stop condition is not achieved,

- First, the built surrogate will estimate the fitness of newly generated individuals, and some top individu-

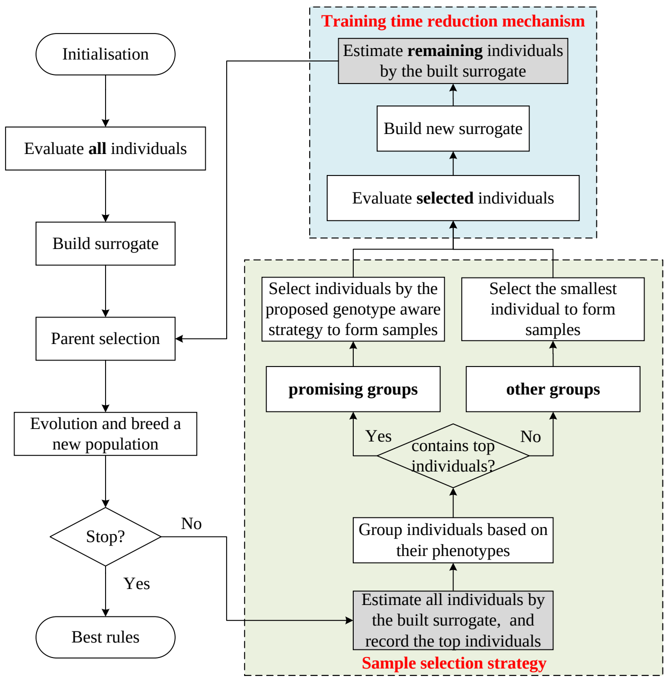
Training time reduction mechanism
Initialisation
Evaluate all individuals
Build surrogate
Parent selection
Evolution and breed a new population
Stop?
No
Yes
Best rules
Estimate remaining individuals by the built surrogate
Build new surrogate
Evaluate selected individuals
Select individuals by the proposed genotype aware strategy to form samples
Select the smallest individual to form samples
promising groups
other groups
Yes
contains top individuals?
No
Group individuals based on their phenotypes
Estimate all individuals by the built surrogate, and record the top individuals
Sample selection strategy

Fig. 4. Flowchart of the proposed algorithm.

als $topIndividuals$ according to their fitness will be recorded.

- Second, the PCs of all newly generated individuals are calculated, and the individuals with the same PC will be placed in one group. If the group contains any individuals in the top individual set, we define this group as a promising group. Thus, we name the groups as *promising groups* and *other groups*.
- Third, we will select individuals/samples for real evaluations to build a surrogate. For promising groups, we will select samples based on the proposed phenotype and genotype based sample selection strategy with niching techniques from each group. For other groups, we will select the smallest individual from each group.
- Finally, the remaining unevaluated individuals in the current generation in the population will be evaluated by the built surrogate.

While the stopping condition is reached, this algorithm will output the best scheduling heuristic learned so far.

Note that we use expensive evaluations for all individuals for the initialised population (the first generation) to get a good start point. We can also see that the built surrogate is used for two purposes, which is highlighted in grey. First, the built surrogate in the previous generation is used to estimate the fitness values of all newly generated individuals to decide top individuals, which will be used to decide which individual groups to be further split into niches with niching techniques. Second, the built surrogate in the current generation will be used to estimate the fitness of the remaining individuals. Note that all samples in the surrogate model are truly evaluated individuals.

B. Phenotype and Genotype Aware Sample Selection Strategy

Fig. 5 shows an example of the work procedure of the proposed phenotype and genotype aware sample selection strategy with nine newly generated individuals. The nine individuals are classified into four groups (i.e., PC_a, PC_b, PC_c and PC_d) according to their PCs. The individuals in the same group have the same PC. In other words, they have the same phenotype.

These groups are divided into promising PC groups and other PC groups based on the quality of contained individuals. If a group contains any individual in the top individuals set $topIndivduals$, this group is considered as a promising group. In the example shown in Fig. 5, the top individuals (i.e., ind_1, ind_2, ind_4) are highlighted in grey, PC_a and PC_d are promising groups since they contain top individuals ind_1 and ind_4, ind_2, respectively. PC_b and PC_c belong to other groups.

1) Genotype Sample Selection Strategy for Promising Groups: The niching techniques [48] have been widely applied to solve multi-objective [49] and multi-modal [50] optimization problems. The technique works by dividing the whole population into multiple niches with an expectation that each niche contains similar individuals. This paper adopts the niching technique to split individuals in the same promising group for selecting samples based on genotype. There are two parameters in the niching technique, i.e., the capacity defines the number of individuals in each niche, whereas the radius controls the coverage of each niche. In this paper, the capacity is infinity, and the radius is based on genotype correlations among individuals, which will be detailed in the subsequent subsection.

An example of the proposed genotype based sample selection strategy with niching technique to further split individuals in promising groups into niches, which is shown in the upper right corner of Fig. 5. In group PC_a, two niches are generated based on the genotypic similarity. The smallest rule in each niche (i.e., ind_1, ind_7) will be selected as the sample. As for PC_d, there is only one niche existing, which indicates the genotypes of ind_2 and ind_6 are similar, and the ind_2 with a smaller size will be selected as a sample.

Algorithm 1 shows the proposed phenotype and genotype aware sample selection strategy. In each generation, we first calculate the PC values of all individuals (line 3). Second, we group the individuals by PCs, and the individuals with the same PC will be placed in one group (line 6). Third, we design a genotype based sample selection strategy to subdivide individuals in the same phenotype group (lines 8-32). For each promising PC group, all individuals will be sorted based on their rule sizes (the number of tree nodes) from smallest to largest (line 9). After that, the first individual with the smallest rule size will be moved into one niche as the center, then the genotype correlation between the center and other individuals is calculated in order (line 24). If the genotype correlation value is larger than the threshold gct, genotypes of these two individuals are similar, and the newcomer individual will be added to the niche (lines 25-28). After calculating the correlation coefficients between all individuals and the niche center point, the remaining individual with the smallest rule

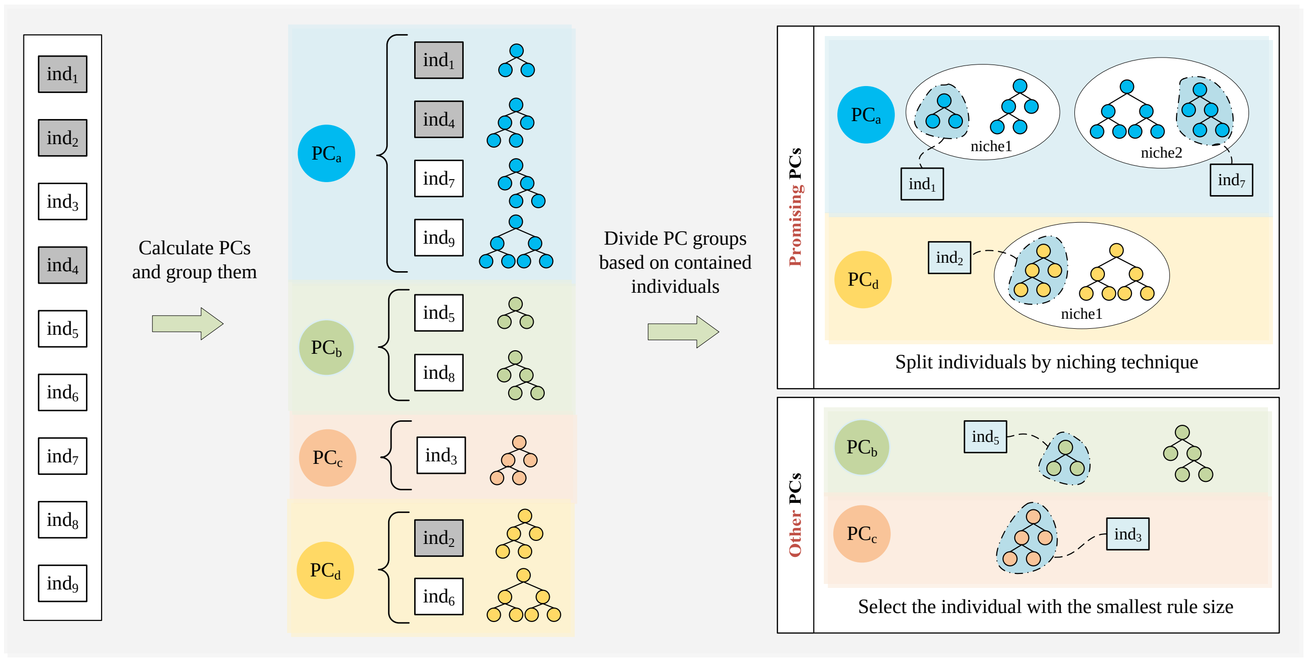
ind1
ind2
ind3
ind4
ind5
ind6
ind7
ind8
ind9
Calculate PCs and group them
PCa
PCb
PCc
PCd
Divide PC groups based on contained individuals
Promising PCs
niche1
niche2
Split individuals by niching technique
Other PCs
Select the individual with the smallest rule size

Fig. 5. An example of the work procedure of the proposed phenotype and genotype aware sample selection strategy.

size outside the niche will be considered as a new center to start a new niche. The algorithm will keep cycling until all individuals are inside one niche. In this way, each PC group is divided into one or multiple niches, then the centers of each niche are selected as the surrogate samples for real evaluations. Since we sort individuals based on rule size before applying the proposed genotype based sample selection strategy in each PC group, according to Algorithm 1, the center individual of each niche will be the smallest individual in that niche.

2) Sample Selection for Other Groups: Since the contained individuals are not effective in other groups, further splitting them to select samples will not affect the algorithm's performance too much. Instead, it will increase the computational cost. Thus, we simply choose to evaluate the smallest individual in other groups to keep the algorithm efficient, which is illustrated in the lower right corner of Fig. 5. For example, ind_5 in PC_b group will be selected. There is a special case (i.e., PC_c) in which there is only one individual in the PC group, thus, we will evaluate it directly to form a sample.

C. Genotypic Characterisation and Genotype Correlation of Individuals

As we mentioned earlier, individuals in the same PC group may have different real fitness since the PC values are generated with a small fixed number of decision situations, which does not necessarily represent all decision situations in a real DFJSS simulation for calculating real fitness. Thus, we design a genotypic characterisation (GC) to measure the similarity of individuals with the same PC to further distinguish individuals from the perspective of their genotypes.

This paper proposes to design the GC according to the frequency of individual terminals since terminals are crucial to the quality of evolved individuals and individuals with close performance tend to use similar terminals [31]. As described earlier, each GP individual is made up of two trees. For each tree, we record the occurrence of each terminal and then divide it by the total occurrence to get the frequency for each terminal. Table III presents an example of the terminal frequency calculation of an individual consisting of three terminals (MWT, W and PT). For example, the frequency of MWT in the routing rule is calculated by $10/(10+2+6) = 0.56$. Similarly, the frequency of other terminals can be obtained. Finally, the GC is shown in Fig. 6. In this way, GC of each rule is a vector, where each dimension value equals to the frequency of a terminal. The order of terminals is fixed for GC to make it comparable among individuals. Routing GC and sequencing GC constitute the whole GC of a GP individual.

After calculating the GCs of individuals, we use Spearman correlation coefficient [30] to measure the genotypic similarity

TABLE III
AN EXAMPLE OF THE TERMINAL FREQUENCY CALCULATION OF AN INDIVIDUAL CONSISTING OF THREE TERMINALS (MWT, W AND PT).

Rule	Terminal	Occurrence	Relative Frequency
Routing	MWT	10	0.56
	W	2	0.11
	PT	6	0.33
Sequencing	MWT	0	0
	W	2	0.20
	PT	8	0.80

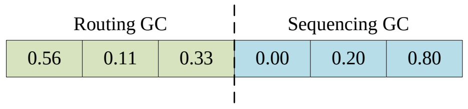

Fig. 6. Example of designed genotypic characterisation of an individual.

Algorithm 1: Proposed genotype aware sample selection strategy for promising groups

```
Input  : Newly generated individuals $ind_1$, $ind_2$, ... , $ind_{popsize}$
         top individuals $topIndividuals$
Output: Representative samples for building surrogate $samples$
1:  Initialise $samples = \emptyset$;
2:  for $i$ = 1 to $popsize$ do
3:      Get PC for individual $ind_i$;
4:      inNiche($ind_i$)=false;
5:  end
6:  $PC \leftarrow$ Promising PC groups;
7:  $ind^{pc} \leftarrow$ Individuals in each promising PC group;
8:  for $i$ = 1 to $|PC|$ do
9:      Sort individuals in $PC_i$ based on rule sizes, smallest to largest;
10:     for $j$ = 1 to $|ind^{pc_i}|$ do
11:         if inNiche($ind_j^{pc_i}$) == true then
12:             continue;
13:         else
14:             Creat a new niche $\{ind_j^{pc_i}\}$;
15:             Set inNiche($ind_j^{pc_i}$) = true;
16:             Add $ind_j^{pc_i}$ to $samples$;
17:             Calculate genotypic characterisation (GC) for individual
                  $ind_j^{pc_i}$ (see Section III-C);
18:         end
19:         for $k$ = 1 to $|ind^{pc_i}|$ do
20:             if inNiche($ind_k^{pc_j}$) == true then
21:                 continue;
22:             else
23:                 Calculate GC for individual $ind_k^{pc_j}$;
24:                 Calculate correlation $c$ of $ind_j^{pc_i}$ and $ind_k^{pc_i}$;
25:                 if $c \geq gct$ then
26:                     Set inNiche($ind_k^{pc_i}$) = true;
27:                     Add $ind_k^{pc_i}$ to the niche $\{ind_j^{pc_i}\}$;
28:                 end
29:             end
30:         end
31:     end
32: end
33: return $samples$
```

between two individuals. A high correlation coefficient indicates these two individuals have similar genotype.

D. Summary

The proposed algorithm considers both the phenotype and genotype of individuals to distinguish individuals for sample selection to build surrogates. Specifically, genotype based sample selection strategy is used within the individuals with the same phenotype. The selected surrogate samples have a wider spread in both the phenotypic and genotypic space, which is expected to lead to a more accurate estimation of individual fitness.

This design has a number of advantages. First, this algorithm considers both phenotype and genotype, which can make the selected samples more representative. Second, making the center of each niche to be the smallest rule as a reference for GC calculations or sample selections can bring two benefits. One is that small rules have potential to be more interpretable and preferable in practical situations. The other is that small rules might have better generalisation ability to avoid overfitting issue. Last but not least, small rules tend to take less time for priority calculations, which can react to make real decisions efficiently.

IV. Experiment Design

A. Simulation Model

Simulation is an effective technique to study complex dynamic JSS problems. [51]. Referring to widely used DFJSS simulation [52], [53], the simulation model assumes that 10 machines are required to process 5000 jobs. The jobs in this study involve a varying number of operations and the number of available machines for a particular operation, which are randomly distributed between 1 and 10 following a uniform distribution. The importance of jobs is determined by their respective weight. Three different weights are assigned for jobs: 20% receive a weight of 1, 60% receive a weight of 2, and the remaining 20% are allocated a weight of 4. In addition, the processing time for each operation is generated from a uniform discrete distribution with values ranging from 1 to 99. A job's due date is determined to be 1.5 times its processing time. New jobs will arrive at the job shop over time following a Poisson process with the rate λ. *Utilisation level* (p) is a crucial factor in simulating job shop scenarios [54], which represents the fraction of time a machine is expected to be occupied. This factor is directly related to how busy the job shop is. It can be calculated as follows.

$$\lambda = \mu * P_M / p \tag{1}$$

where λ defines the expected utilization rate of a machine in a Poisson process. μ represents the average processing time of machines, and P_M represents the likelihood of a job entering a machine. To obtain the steady-state performance, the initial 1000 jobs are regarded as warm-up jobs and omitted from the objective calculation. Data collection will begin from the subsequent 5000 jobs. The simulation will terminate upon completion of the 6000th job.

During the training stage, all algorithm generate a new training instance following a set of suggestions in [38], [55] in each generation to enhance the generalization capabilities of the evolved scheduling rules. During the test stage, the best scheduling rule obtained from the training process is applied to 50 previously unseen DFJSS test instances, and the test performance is evaluated based on the mean objective value across the entire test set.

B. Design of Comparisons

Three utilisation levels (i.e., 0.75, 0.85, 0.95), along with two commonly used objectives, i.e., mean-flowtime (Fmean) and mean-weighted-tardiness (WTmean) are used to form six scenarios. The examined scenarios are named as <objective, utilisation level> such as <WTmean, 0.85>.

This paper compares the proposed algorithm with the state-of-the-art algorithms on sample aware surrogate-assisted GP for JSS. The details are shown as follows.

1) GP: The GP algorithm with multi-tree representation [40] is selected as a baseline algorithm to learn two rules simultaneously for DFJSS for comparison.
2) SGP_Ran: SGP_Ran simply selects samples *randomly* from the population is included for comparison to show the importance of sample aware surrogate building.

TABLE IV
THE TERMINAL SET.

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

3) SGP_PC [29]: The state-of-the-art surrogate-assisted GP algorithm with sample selection strategy is also included for comparison, where only the smallest rule in each PC group will be truly evaluated to form samples for surrogates.
4) SGP_PCGC: Our proposed algorithm that considers the phenotype represented by *phenotypic characterisation* and genotype represented by *genotypic characterisation* of GP individuals for sample selection in surrogate.

For algorithms SGP_Ran, SGP_PC and SGP_PCGC, the individuals have either estimated fitness or real fitness. This is because only a subset of individuals are evaluated with expensive simulations, and used for extracting samples to build surrogates to estimate the fitness of remaining individuals in the current generation. During the evolution process, the real fitness and the estimated fitness are treated equally for parent selection. However, individuals with true fitness values better reflect the quality of learned scheduling heuristics compared to those with estimated fitness. Thus, we choose the best individual with real evaluation in each generation as the best learned scheduling heuristic for record. In addition, for a fair comparison, the number of selected samples in SGP_Ran is equal to the number of unique PC.

C. Parameter Settings

Each GP individual of the algorithm consists of terminals and functions. The terminals can be regarded as the features of the job shop, which are related to machines (i.e., MWT, WIQ and NIQ), operations (i.e., NPT, OWT and PT) and jobs (i.e., WKR, NOR, TIS and W). The details are shown in Table IV. The function set is set to $\{+, -, *$, protected $/$, max, $min\}$. The protected "/" returns one if divided by zero. The min and max functions return the minimum and maximum of their arguments, respectively. The other parameter settings of GP as suggested in [39], [56], are shown in Table V. The top individuals ratio "30%" and genotype correlation threshold gct (i.e., 0.7) control the number of samples/individuals in niches and the number of PC groups for niching, respectively.

TABLE V
THE PARAMETER SETTINGS IN GP.

Parameter	Value
The number of generations	100
Population size	500
Parent selection	Tournament selection with size 5
The number of elites for each subpopulation	10
Initial minimum / maximum depth	2 / 6
Maximal depth of programs	8
Crossover / Mutation / Reproduction rate	80% / 15% / 5%
Terminal / non-terminal selection rate	10% / 90%
Method for initialising population	ramped-half-and-half
The genotype correlation threshold gct	0.7
The top individual ratio	30%
*The fixed training time	150

* : for experiments with the fixed training time (in minutes) only

V. RESULTS AND DISCUSSIONS

We conducted a series of experiments in different scenarios to investigate the *efficiency* (i.e., training time) and *effectiveness* (i.e., objective values on test instances) of our proposed algorithm. 30 independent runs have been conducted to verify the performance of the proposed algorithm. The Friedman's test and Wilcoxon rank-sum test with a significance level of 0.05 are used to verify the effectiveness of the proposed algorithm. The algorithm's ranking across all examined scenarios, as determined by Friedman's test, is reflected in the "Average Rank". According to the Wilcoxon rank-sum test, the following results show that "$\uparrow$", "$\downarrow$", "$\approx$" indicate an algorithm is significantly better than, worse than, or similar to the compared algorithm before it.

A. Quality of the Learned Scheduling Heuristics ***with the Same Training Time***

When using the training time as the stopping criterion, the number of generations of different algorithms vary since they have different evaluation cost at each generation. We cannot compare the performance among them with the learned scheduling heuristics in the same generation [26]. Borrowing the idea in [26], the computational budget is divided equally into 60 groups, the training time of each group is 2.5 minutes (150/60). The beginning point of kth group is $2.5 * k$, such as 5 minutes, 10 minutes and so on. The population in the generation closest to the beginning point of each group will be used to represent the performance of the algorithms during the evolutionary process.

a) The Quality of the Learned Best Scheduling Heuristics: Table VI shows the objective values on test instances of GP, SGP_Ran, SGP_PC and SGP_PCGC with the same training time in six scenarios. Overall, the results show that the phenotype and genotype based sample aware SGP_PCGC performs the best with the smallest average rank value of 1.85 among all involved algorithms. We can also see that SGP_PCGC achieves the best mean objective values in five out of six scenarios (i.e., <Fmean, 0.75>, <Fmean, 0.85>, <Fmean, 0.95>, <WTmean, 0.75> and <WTmean, 0.85>).

It is interesting that SGP_Ran obtains comparable performance with GP in 5 out of 6 scenarios, and achieve significantly better performance in one scenario. However, we know that the samples for building surrogates in SGP_Ran

TABLE VI
THE MEAN (STANDARD DEVIATION) OF OBJECTIVE VALUES ON TEST INSTANCES OF GP, SGP_RAN, SGP_PC AND SGP_PCGC **WITH THE SAME TRAINING TIME** (IN MINUTES) ACCORDING TO 30 INDEPENDENT RUNS IN SIX SCENARIOS.

Scenarios	GP	SGP_Ran	SGP_PC	SGP_PCGC
<Fmean, 0.75>	336.57(1.61)	335.52(1.0)(↑)	335.37(0.71)(↑)(≈)	**335.09(0.55)**(↑)(≈)(≈)
<Fmean, 0.85>	385.47(2.29)	385.82(3.00)(≈)	384.10(0.98)(↑)(↑)	**383.62(0.79)**(↑)(↑)(↑)
<Fmean, 0.95>	554.93(7.55)	552.58(3.85)(≈)	549.11(3.32)(↑)(↑)	**547.81(1.67)**(↑)(↑)(↑)
<WTmean, 0.75>	27.24(1.41)	27.03(0.92)(≈)	26.97(1.75)(↑)(≈)	**26.63(1.05)**(↑)(↑)(≈)
<WTmean, 0.85>	75.61(2.41)	75.91(2.39)(≈)	74.46(1.93)(↑)(↑)	**74.24(1.63)**(↑)(↑)(≈)
<WTmean, 0.95>	297.94(11.53)	300.39(12.00)(≈)	**291.84(5.11)**(↑)(↑)	292.59(8.10)(↑)(↑)(≈)
Win / Draw / Lose	6 / 0 / 0	5 / 1 / 0	2 / 4 / 0	N/A
Average Rank	3.17	2.86	2.12	**1.85**

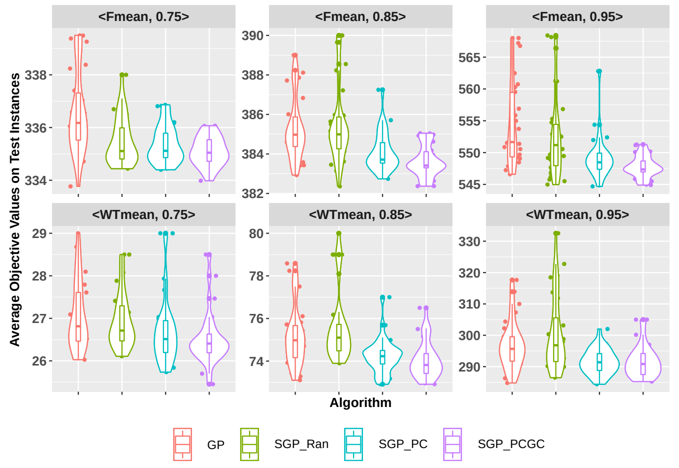

Fig. 7. The violin plots of the average objective values on test instances of GP, SGP_Ran, SGP_PC and SGP_PCGC with the same training time (in minutes) according to 30 independent runs in six scenarios.

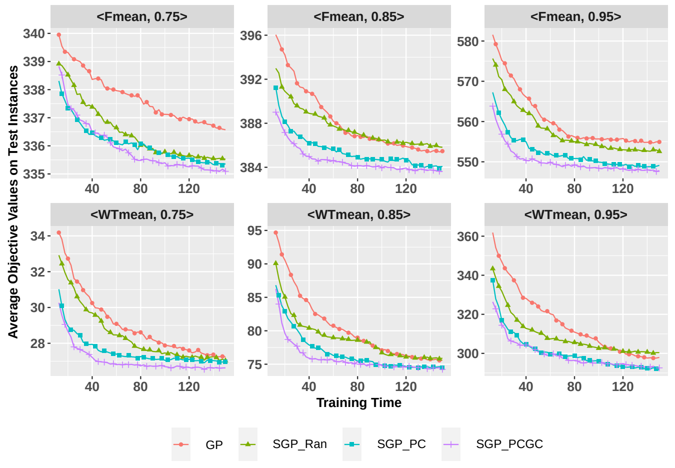

Fig. 8. The curves of average objective values according to 30 independent runs on test instances of GP, SGP_Ran, SGP_PC and SGP_PCGC with the same training time (in minutes) in six different scenarios.

are randomly selected, and the fitness estimation might not be accurate. A reason for this phenomenon is that with the same training time, the number of generations of SGP_Ran is much larger than GP, which makes up the performance of SGP_Ran. In addition, the results show that SGP_PC and SGP_PCGC win baseline GP in all scenarios, and significantly outperform than SGP_Ran in most of scenarios. This verifies the effectiveness of sample aware KNN based surrogates. More importantly, compared with SGP_PC, SGP_PCGC achieves smaller mean objective values and standard deviation values in five out of six scenarios, and obtains significantly better performance in two out of six scenarios. This verifies the effectiveness of the proposed phenotype and genotype based sample aware surrogate-assisted GP in DFJSS.

Fig. 7 shows the violin plots of the objective values on test instances of GP, SGP_Ran, SGP_PC and SGP_PCGC with the same training time in six scenarios. SGP_PCGC shows its superiority with lower distributions of objective values in five out of six scenarios (i.e., <Fmean, 0.75>, <Fmean, 0.85>, <Fmean, 0.95>, <WTmean, 0.75> and <WTmean, 0.85>). The results demonstrate the effectiveness of SGP_PCGC.

b) The Curves of the Average Objective Values on Test Instances: Fig. 8 shows the curves of average objective values on test instances of GP, SGP_Ran, SGP_PC and SGP_PCGC with the same training time in six different scenarios based on 30 independent runs. The results show that SGP_PCGC can obtain better scheduling heuristics than other algorithms from the early stage during the evolutionary process in four out of six scenarios (i.e., <Fmean, 0.85>, <Fmean, 0.95>, <WTmean, 0.75> and <WTmean, 0.85>). We can see that SGP_PCGC can sustain the advantage into the later evolutionary processes to achieve competitive performance. The effectiveness of the proposed phenotype and genotype based sample aware surrogate-assisted GP is further verified.

In general, we can observe that proposed algorithm SGP_PCGC can improve the quality of evolved scheduling heuristics and achieve better scheduling heuristics faster than its counterparts in most scenarios while fixing the training time.

*B. Quality of the Learned Scheduling Heuristics **with the Same Number of Generations** and Training Efficiency*

Compared with GP, only a subset of individuals of SGP_Ran, SGP_PC and SGP_PCGC are evaluated with expensive simulations in each generation. It is interesting to know how these algorithms perform with the same number of generations.

TABLE VII
THE MEAN (STANDARD DEVIATION) OF OBJECTIVE VALUES ON TEST INSTANCES OF GP, SGP_RAN, SGP_PC AND SGP_PCGC **WITH THE SAME NUMBER OF GENERATIONS** ACCORDING TO 30 INDEPENDENT RUNS IN SIX SCENARIOS.

Scenarios	GP	SGP_Ran	SGP_PC	SGP_PCGC
<Fmean, 0.75>	336.94(1.72)	337.93(1.71)(↓)	**336.33(1.60)**(≈)	336.72(1.78)(≈)
<Fmean, 0.85>	**385.62(2.74)**	389.43(5.05)(↓)	386.13(3.68)(≈)	386.88(3.63)(≈)
<Fmean, 0.95>	554.55(8.05)	566.01(11.91)(↓)	555.13(6.76)(≈)	**553.59(7.37)**(≈)
<WTmean, 0.75>	**27.22(1.46)**	29.40(2.71)(↓)	27.69(1.90)(≈)	27.97(2.04)(≈)
<WTmean, 0.85>	**75.51(2.03)**	80.49(6.67)(↓)	77.86(5.57)(≈)	77.24(4.27)(≈)
<WTmean, 0.95>	**296.91(10.74)**	314.46(18.17)(↓)	298.93(9.33)(≈)	297.53(8.91)(≈)

TABLE VIII
THE MEAN (STANDARD DEVIATION) OF THE **TRAINING TIME** (IN MINUTES) OF GP, SGP_PC AND SGP_PCGC ACCORDING TO 30 INDEPENDENT RUNS IN SIX SCENARIOS.

Scenarios	GP	SGP_PC	SGP_PCGC
<Fmean, 0.75>	121(19)	43(10) (↑)	41(11)(↑)(≈)
<Fmean, 0.85>	127(24)	39(10)(↑)	37(6)(↑)(≈)
<Fmean, 0.95>	137(20)	39(8)(↑)	41(10)(↑)(≈)
<WTmean, 0.75>	127(19)	46(11)(↑)	46(11)(↑)(≈)
<WTmean, 0.85>	130(20)	40(8)(↑)	46(12)(↑)(≈)
<WTmean, 0.95>	131(19)	45(8)(↑)	45(8)(↑)(≈)

1) Quality of Learned Scheduling Heuristics with the Same Number of Generations: Table VII shows the objective values on test instances of GP, SGP_Ran, SGP_PC and SGP_PCGC with the same number of generations in six scenarios. The results show that there is no significant statistical difference in performance among GP, SGP_PC and SGP_PCGC. However, SGP_Ran performs significantly worse than its counterparts in all scenarios due to inaccurate estimations induced by random sample selection for building surrogates. This shows that the effectiveness of sample aware KNN based surrogate models. In addition, compared with SGP_PC, SGP_PCGC achieves better mean objective values in half of the examined scenarios (i.e., <Fmean, 0.95>, <WTmean, 0.85> and <WTmean, 0.95>).

2) Training Efficiency: An algorithm's efficiency is reflected in its training time. Since SGP_Ran performs worse than other algorithms, this section only chooses GP, SGP_PC and SGP_PCGC for investigations on training efficiency. Table VIII shows the training time (in minutes) of GP, SGP_PC and SGP_PCGC in six scenarios. Both algorithms, SGP_PC and SGP_PCGC, significantly reduce the training time required, to approximately 1/3 of that required by GP across all scenarios. The training time of SGP_PCGC is similar with SGP_PC, which indicates that the further sample selection based on the genotype of GP individuals does not affect the efficiency of SGP_PCGC.

The key of determining the training efficiency of GP algorithms including SGP_PCGC is the number of individuals with real evaluations. Fewer number of evaluated individuals leads to shorter training time. To further investigate the efficiency of SGP_PCGC, Fig. 9 shows the curves of the number of evaluated individuals of GP, SGP_PC, and SGP_PCGC. The curves of baseline GP are always a line at 500 which equals the population size, since there is no surrogate mechanism to reduce the efficiency of individual evaluations. We can observe that the number of evaluated individuals of SGP_PC and SGP_PCGC reduces rapidly in the first 15 generations, i.e., from 400 to 200 and then maintain at around 200. In addition, there is no significant statistical difference between SGP_PC and SGP_PCGC, which indicates their training time are similar. This conclusion is consistent with the finding in Table VIII.

Overall, it can be observed that with the same number of generations, the proposed algorithm can obtain competitive performance with a much short time, i.e., around 1/3 of the time needed for GP. This shows the efficiency and effectiveness of the proposed SGP_PCGC.

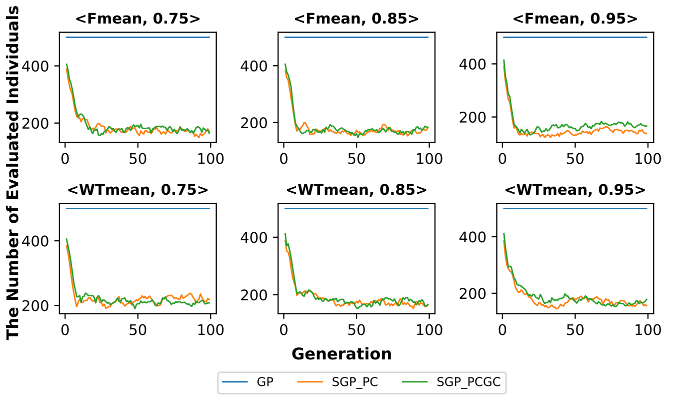

Fig. 9. The curves of the number of individuals with real evaluations of GP, SGP_PC and SGP_PCGC during the training process over 30 independent runs in six DFJSS scenarios.

C. The Number of Extra Evaluated Individuals of SGP_PCGC

We will conduct the results obtained with the same training time for more analyses in this section. SGP_PCGC selects more representative individuals in each PC group for real evaluations to build KNN based surrogates, while, the number of selected individuals for real evaluations of SGP_PC equals to the number of PC groups. This section will analyses the number of extra evaluated individuals by SGP_PCGC compared with the mechanism of SGP_PC. Specifically, we define that the number of extra evaluated individuals by SGP_PCGC equals the subtraction of the number of real evaluated individuals and the number of PC groups.

Fig. 10 shows the curves of the number of extra evaluated individuals by SGP_PCGC during the training process in six scenarios. Note that for convenience, we still use generations as x-axis, and the number of generations can slightly differ among different algorithms or scenarios. The results show that the number of extra evaluated individuals is larger in the early stages of evolutionary process of SGP_PCGC than the ones in the later evolution stages in all scenarios. Along with generations, the number of extra individual evaluations has dropped from around 30 to a few. A reason is that the genotypes of individuals become more similar along with generations, and there are fewer individuals selected to be samples for KNN based surrogate. This indicates that the proposed phenotype and genotype based sample aware surrogate-assisted GP has a big/small influence in the early/later stage during the evolutionary process.

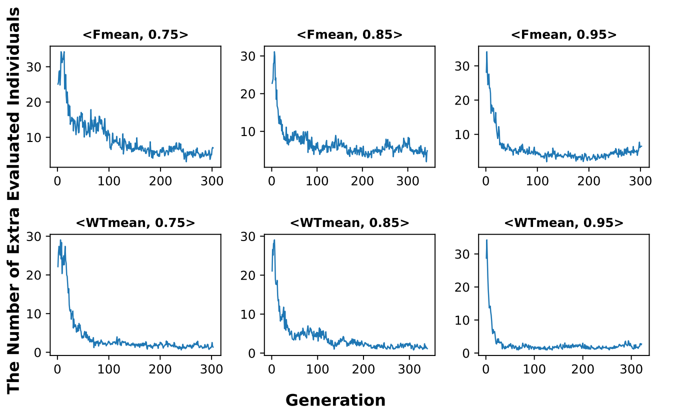

Fig. 10. The curves of the number of extra evaluated individuals by SGP_PCGC compared with the mechanism of SGP_PC during the training process over 30 independent runs in six DFJSS scenarios.

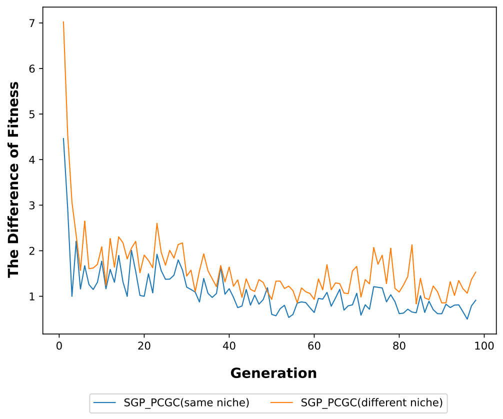

Fig. 12. The fitness difference between individuals in the same and different niche(s) of SGP_PCGC in the scenario <Fmean, 0.85> of 30 runs.

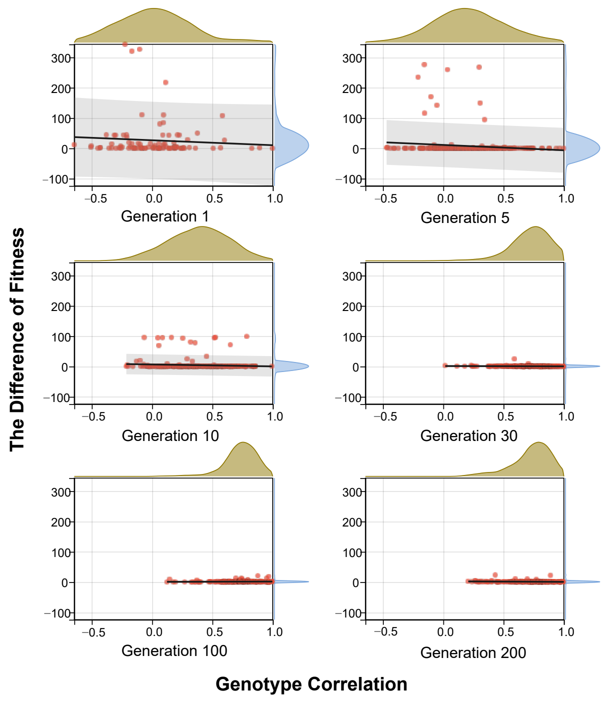

Fig. 11. Genotype correlation and the difference of fitness in the scenario <Fmean, 0.85> of one GP run.

D. Genotype Correlation and Fitness Gap of GP Individuals

This section analyses the relation between genotype correlation and fitness difference for individuals with the same PC during the evolutionary process. We choose the smallest rule in each PC group as the base individual, and the calculations of genotype correlation and fitness difference are between individuals and the base individual within the same PC group.

Fig. 11 shows the plots of genotype correlation and fitness difference in the scenario <Fmean, 0.85> in one run across several generations (i.e., generations 1, 5, 10, 30, 100 and 200). It is clear that individuals with the same PC could have different fitness. This observation shows that among the individuals with the same PC, it is necessary to further select individuals/samples to build KNN based surrogates which is the main focus of SGP_PCGC. By observing the density plots above each subgraph, the results show that the genotype correlations of individuals with the same PC increase along with the evolutionary process. The most frequent phenotype correlation at generation 1 is about 0, 0.2 at generation 5, 0.4 at generation 10, and 0.8 at generation 30, 100 and 200. This indicates that the genotypes of individuals with the same PC have become similar and similar during the evolutionary process. In addition, the density plots on the right side of each subplot illustrate that the difference in fitness values between individuals with the same PC is getting smaller along with generations. This indicates that SGP_PCGC has more effect on the early stage of evolutionary process, which aligns with our findings in Section V-C.

E. Fitness Difference in the Same and Different Niches

For each PC group, the proposed algorithm SGP_PCGC aims to group individuals with similar genotypes in one niche. The differences in fitness between individuals occupying the same niche are expected to be smaller than the fitness differences of individuals between different niches. This paper defines the fitness difference among one niche as the average fitness difference of all individuals with the smallest individuals in each same niche of each PC group. The fitness difference among different niches is the average fitness difference among the center individuals from each niche by taking the center individual of the first niche as the base for calculations in each PC group.

Fig. 12 shows the curves of mean fitness difference between individuals in the same and different niche(s) of SGP_PCGC in the scenario <Fmean, 0.85>. Since the number of extra evaluated individuals after generation 100 is small according to Fig. 10, which indicates there are few niches after generation 100, resulting in less available data on fitness differences in different niches. Thus, we will only collect data from the first 100 generations for analyses. It is clear that the fitness differences in the same niches are always smaller than those in different niches. In addition, the fitness difference among same niche becomes smaller and smaller along with evolutionary process. This shows the effectiveness of SGP_PCGC to put individuals with similar fitness into the same group/niche,

TABLE IX
THE MEAN (STANDARD DEVIATION) OF THE SIZES EVOLVED THE BEST RULE OF GP, SGP_RAN, SGP_PC AND SGP_PCGC WITH **THE SAME TRAINING TIME** ACCORDING TO 30 INDEPENDENT RUNS IN SIX SCENARIOS.

Scenarios	GP	SGP_Ran	SGP_PC	SGP_PCGC
<Fmean, 0.75>	85.67(17.07)	92.80(23.72)(≈)	85.33(22.01)(≈)(≈)	90.67(27.17)(≈)(≈)(≈)
<Fmean, 0.85>	85.20(19.07)	100.27(26.91)(↓)	96.00(22.47)(≈)(≈)	87.53(21.99)(≈)(≈)(≈)
<Fmean, 0.95>	87.47(20.01)	104.80(26.47)(↓)	92.80(21.70)(≈)(≈)	98.60(21.85)(↓)(≈)(≈)
<WTmean, 0.75>	92.80(24.65)	103.67(21.83)(↓)	89.93(25.72)(≈)(↑)	87.6(22.63)(≈)(↑)(≈)
<WTmean, 0.85>	95.13(21.32)	105.33(19.29)(≈)	80.73(21.46)(↑)(↑)	90.67(19.50)(↑)(↑)(≈)
<WTmean, 0.95>	94.67(20.38)	108.13(24.19)(↓)	84.27(19.02)(≈)(↑)	88.80(20.26)(≈)(↑)(≈)

which further helps the sample selection for building KNN based surrogates.

VI. FURTHER ANALYSES

A. Sizes of the best Learned Scheduling Heuristics

This section will investigate the impact of our proposed algorithm on the sizes of the evolved best rules in the experiments with the same generations and training time, respectively.

*a) Sizes of Learned Scheduling Heuristics **with the Same Training Time**:* Table IX shows the sizes of evolved the best rule of GP, SGP_Ran and SGP_PC, SGP_PCGC with the same training time. The results show that SGP_Ran learns larger rules than GP. On the contrary, SGP_PC and SGP_PCGC can achieve similar rule sizes with GP but with better performance as shown in Section V-A and V-B. This shows the effectiveness of choosing the smallest individual as samples in the individual group with the same PC of SGP_PC for rule size control. This can also verify the effectiveness of taking the smallest rule in the individual group with the same PC as the base to calculate the genotype correlation. Compared with SGP_PC, SGP_PCGC does not significantly increase the rule size in all examined scenarios.

*b) Sizes of Learned Scheduling Heuristics **with the Same Number of Generations**:* Section V-B shows that SGP_PC and SGP_PCGC obtain comparable performance with the baseline GP but with lower computational cost. It is interesting to observe the difference in sizes of evolved rules with GP, SGP_PC and SGP_PCGC. Note that since SGP_Ran performs much worse than other algorithms, we do not include its rule size information here. Table X shows the sizes of evolved the best rule of GP, SGP_PC and SGP_PCGC with the same number of generations. SGP_PC and SGP_PCGC can obtain significantly smaller rules than GP in half of the examined scenarios, which means SGP_PC and SGP_PCGC can achieve similar performance with GP by using much smaller rules. The rule sizes of SGP_PC and SGP_PCGC do not significantly differ.

TABLE X
THE MEAN (STANDARD DEVIATION) OF THE SIZES OF EVOLVED THE BEST RULE OF GP, SGP_PC AND SGP_PCGC WITH **THE SAME NUMBER OF GENERATIONS** ACCORDING TO 30 INDEPENDENT RUNS IN SIX SCENARIOS.

Scenarios	GP	SGP_PC	SGP_PCGC
<Fmean, 0.75>	79.53(23.39)	76.13(27.73)(≈)	69.73(22.17)(≈)(≈)
<Fmean, 0.85>	79.33(21.31)	73.40(20.04)(≈)	78.40(27.45)(≈)(≈)
<Fmean, 0.95>	91.33(21.16)	82.33(22.85)(↑)	81.13(26.43)(↑)(≈)
<WTmean, 0.75>	92.27(26.77)	82.67(20.04)(≈)	74.33(23.73)(↑)(≈)
<WTmean, 0.85>	94.53(18.93)	70.33(23.12)(↑)	82.60(29.26)(↑)(≈)
<WTmean, 0.95>	92.27(23.25)	71.00(22.26)(↑)	79.87(15.42)(≈)(≈)

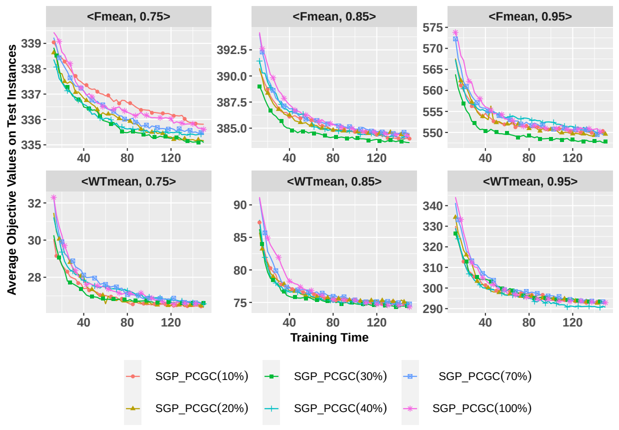

Fig. 13. The curves of average objective values on test instances of SGP_PCGC with different top individuals ratios over 30 independent runs in six DFJSS scenarios.

B. Parameter Sensitivity Analyses

1) Top Individuals Ratio: The top individuals ratio controls the proportion of PC groups for niching. A small ratio may reduce the number of evaluated individuals due to the less PC groups for further subdividing to get more samples. A large ratio will bring more individuals for evaluations, which is time-consuming. This section investigates the effect of different top individuals ratios on the performance of SGP_PCGC. Fig. 13 shows the curves of average objective values on test instances of SGP_PCGC with different top individuals ratios. The performance of SFP_PCGC with the ratio of 30% is better than its counterparts in scenarios <Fmean, 0.75>, <Fmean, 0.85>and <Fmean, 0.95>. In the remaining scenarios, although all algorithms obtain similar final performance, SGP_PCGC with the ratio of 30% can find better scheduling heuristics faster in the early stage. Therefore, top individuals ratio of 30% is used in this paper.

2) Genotype Correlation Threshold in SGP_PCGC: The genotype correlation threshold *gct* in SGP_PCGC determines the degree of genotype relatedness of individuals to enter the same niche. Fig. 14 presents the curves of average objective

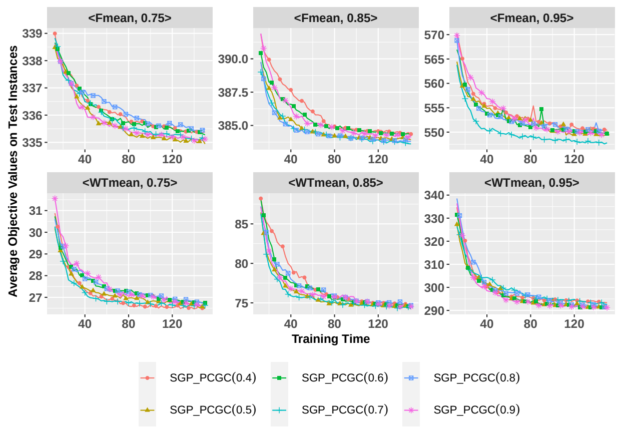

Fig. 14. The curves of average objective values on test instances of SGP_PCGC with different correlation thresholds in niching over 30 independent runs in six DFJSS scenarios.

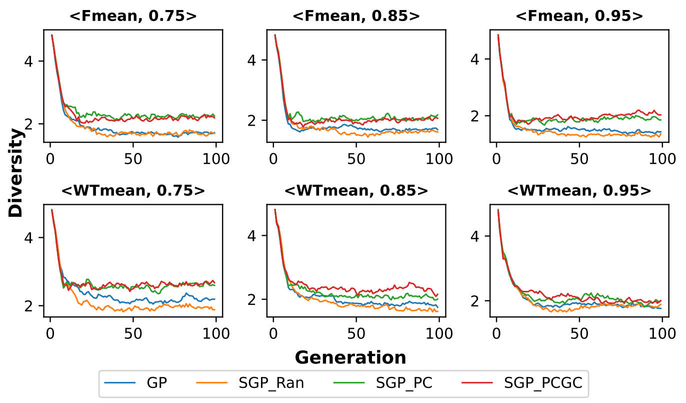

Fig. 15. The curves of phenotypic diversity values in every generation of GP, SGP_Ran, SGP_PC and SGP_PCGC according to 30 independent runs in six scenarios.

values on test instances of SGP_PCGC with different correlation thresholds for SGP_PCGC in six DFJSS scenarios. In general, the quality of learned scheduling heuristics of SGP_PCGC with different correlation thresholds are similar in all scenarios. However, by looking at the curves, SGP_PCGC with the correlation threshold of 0.7 is a good value to start for all scenarios. In scenarios <Fmean, 0.75>, there are some minor differences among the algorithms, and the correlation threshold of 0.5 seems slightly better than other algorithms. This means fine-tuning on specific scenarios may further improve the performance of SGP_PCGC. Without loss of generalisation, this paper chooses 0.7 as the genotype correlation threshold.

C. Population Diversity

Diversity is an important criterion for tracking the population status during the evolutionary process. Throughout the GP literature, diversity is consistently emphasized as crucial for avoiding premature convergence to local optima [23], [57], [58]. This paper uses entropy to measure population diversity, which is calculated as

$$entropy = -\sum_{c \epsilon C}(\frac{|c|}{|inds|})log(\frac{|c|}{|inds|}) \qquad (2)$$

where C is the set of clusters obtained using the DBScan clustering algorithm [59] with the phenotypic distance measure and a cluster radius of 0 [6]. A larger entropy means a higher diversity of the population.

Since the number of generations can differ among different algorithms or scenarios if using a fixed training time as the stopping criterion, we use the results with the same number of generations as the stopping criterion to analyze the population diversity. Fig. 15 shows the curves of diversity values in every generation of GP, SGP_Ran, SGP_PC and SGP_PCGC. we find that the diversity of all algorithms reduces along with the generations and decreases rapidly in around the first 10 generations. This phenomenon is commonly observed in GP [60]. After around generation 10, SGP_PC and SGP_PCGC have a higher diversity value than GP and SGP_PC in all scenarios. Since the main selection pressure control mechanism is tournament selection in GP, this suggests that more diverse individuals are chosen to generate offspring in SGP_PC and SGP_PCGC. This also means that promising and diverse individuals are well recognised and kept in the population. On the contrary, SGP_Ran loses population diversity quickly, and gets worse diversity than baseline GP. The reason is that the samples/individuals for building surrogates of SGP_Ran are randomly selected, and the KNN based surrogate with randomly selected individuals are not representative and cannot keep diverse individuals in the population.

VII. Conclusions

The goal of this article was to develop an effective sample selection strategy to select representative samples for KNN based surrogate-assisted GP to evolve promising scheduling heuristics in DFJSS efficiently. To achieve this goal, we proposed an effective phenotype and genotype based sample selection strategy with niching technique.

The results showed that with the same training time, the proposed SGP_PCGC can obtain better performance while converging faster than the other algorithms in most scenarios. With the same number of generations, the proposed algorithm SGP_PCGC can achieve comparable scheduling heuristics in all the examined scenarios with only about one third training time of the baseline GP algorithm. The effectiveness of the proposed algorithm is also verified by the analyses of the number of extra evaluated individuals by SGP_PCGC, the relation of genotype correlation and fitness difference during the evolutionary process, and population diversity along with generations. In terms of rule size, with the same training time, SGP_PCGC can obtain better performance without increasing the rule size. With the same number of generations, compared with baseline GP, SGP_PCGC can achieve comparable performance but with smaller rules. In addition, parameter sensitivity analyses were also carried out to investigate the robustness of the proposed algorithms.

There are several interesting directions that could be explored in the future. We plan to extend the proposed algorithm to other dynamic combinatorial optimisation problems such as vehicle routing, cloud scheduling and online packing. In addition, we will take the whole structure of GP individuals as their genotype to further investigate more effective strategies to select more representative individuals to build surrogate models.

References

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

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

[3] 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.

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

[5] F. Zhang, "Genetic programming hyper-heuristics for dynamic flexible job shop scheduling," Ph.D. dissertation, Open Access Te Herenga Waka-Victoria University of Wellington, 2021.

[6] 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*, 2022, Doi: 10.1109/TCYB.2022.3196887.

[7] 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*, 2022, Doi: 10.1109/TEVC.2022.3199783.

[8] 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.

[9] 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.

[10] H. Chen, C. Chu, and J.-M. 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.

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

[12] F. Pezzella, G. Morganti, and G. Ciaschetti, "A genetic algorithm for the flexible job-shop scheduling problem," *Computers & Operations Research*, vol. 35, no. 10, pp. 3202–3212, 2008.

[13] A. Muhlemann, A. Lockett, and C. K. Farn, "Job shop scheduling heuristics and frequency of scheduling," *The International Journal of Production Research*, vol. 20, no. 2, pp. 227–241, 1982.

[14] X. Li, L. Gao, X. Li, and L. Gao, "Gep-based reactive scheduling policies for dynamic fjsp with job release dates," *Effective Methods for Integrated Process Planning and Scheduling*, pp. 405–428, 2020.

[15] 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.

[16] J. R. Koza, *Genetic programming: A paradigm for genetically breeding populations of computer programs to solve problems*. Stanford University, Department of Computer Science Stanford, CA, 1990, vol. 34.

[17] W. B. Langdon, "Genetic programming and data structures: genetic programming+ data structures= automatic programming!" 1998.

[18] 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*, 2023, Doi: 10.1109/TEVC.2023.3255246.

[19] W. Banzhaf, "Genotype-phenotype-mapping and neutral variation-a case study in genetic programming," in *PPSN*, vol. 3, 1994, pp. 322–332.

[20] T. Hu, M. Tomassini, and W. Banzhaf, "A network perspective on genotype–phenotype mapping in genetic programming," *Genetic Programming and Evolvable Machines*, vol. 21, pp. 375–397, 2020.

[21] Z. Vašíček and K. Slanỳ, "Efficient phenotype evaluation in cartesian genetic programming," in *Genetic Programming: 15th European Conference, EuroGP 2012, Málaga, Spain, April 11-13, 2012. Proceedings 15*. Springer, 2012, pp. 266–278.

[22] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Phenotype based surrogate-assisted multi-objective genetic programming with brood recombination for dynamic flexible job shop scheduling," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*, 2022, pp. 1218–1225.

[23] E. K. Burke, S. Gustafson, and G. Kendall, "Diversity in genetic programming: An analysis of measures and correlation with fitness," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 47–62, 2004.

[24] Y. Jin, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[25] Y. Jin, H. Wang, T. Chugh, D. Guo, and K. Miettinen, "Data-driven evolutionary optimization: An overview and case studies," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 3, pp. 442–458, 2018.

[26] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling," *IEEE Transactions on Cybernetics*, vol. 52, no. 8, pp. 8142–8156, 2021.

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

[28] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Instance rotation based surrogate in genetic programming with brood recombination for dynamic job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2022. Doi: 10.1109/TEVC.2022.3180693.

[29] L. Zhu, F. Zhang, X. Zhu, K. Chen, and M. Zhang, "Sample-aware surrogate-assisted genetic programming for scheduling heuristics learning in dynamic flexible job shop scheduling," in *Proceedings of the genetic and evolutionary computation conference*. ACM, 2023, Doi: 10.1145/3583131.3590440.

[30] W. W. Daniel *et al.*, *Applied nonparametric statistics*. Houghton Mifflin, 1978.

[31] 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.

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

[33] F. Zhang, Y. Mei, and M. Zhang, "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.

[34] ——, "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.

[35] S. Nguyen, D. Thiruvady, M. Zhang, and K. C. Tan, "A genetic programming approach for evolving variable selectors in constraint programming," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 3, pp. 492–507, 2021.

[36] M. Durasevic, D. Jakobovic, and K. Knezevic, "Adaptive scheduling on unrelated machines with genetic programming," *Applied Soft Computing*, vol. 48, pp. 419–430, 2016.

[37] K. Jaklinovic, M. Dhurasevic, and D. Jakobovic, "Designing dispatching rules with genetic programming for the unrelated machines environment with constraints," *Expert Systems with Applications*, vol. 172, p. 114548, 2021.

[38] 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.

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

[40] 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.

[41] X. Sun, D. Gong, Y. Jin, and S. Chen, "A new surrogate-assisted interactive genetic algorithm with weighted semisupervised learning," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 685–698, 2013.

[42] Y. Sun, H. Wang, B. Xue, Y. Jin, G. G. Yen, and M. Zhang, "Surrogate-assisted evolutionary deep learning using an end-to-end random forest-based performance predictor," *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 2, pp. 350–364, 2019.

[43] K. Chen, B. Xue, M. Zhang, and F. Zhou, "Correlation-guided updating strategy for feature selection in classification with surrogate-assisted particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 26, no. 5, pp. 1015–1029, 2021.

[44] J. Branke, T. Hildebrandt, and B. Scholz-Reiter, "Hyper-heuristic evolution of dispatching rules: a comparison of rule representations," *Evolutionary Computation*, vol. 23, no. 2, pp. 249–277, 2015.

[45] 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.

[46] 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.

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

[48] S. W. Mahfoud, *Niching methods for genetic algorithms*. University of Illinois at Urbana-Champaign, 1995.

[49] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *Proceedings of the first IEEE conference on evolutionary computation. IEEE world congress on computational intelligence*. Ieee, 1994, pp. 82–87.

[50] B. Y. Qu, J. J. Liang, and P. N. Suganthan, "Niching particle swarm optimization with local search for multi-modal optimization," *Information Sciences*, vol. 197, pp. 131–143, 2012.

[51] A. Baykasoğlu, M. Göçken, and L. Özbakir, "Genetic programming based data mining approach to dispatching rule selection in a simulated job shop," *Simulation*, vol. 86, no. 12, pp. 715–728, 2010.

[52] 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.

[53] 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.

[54] 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.

[55] T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach," in *Proceedings of the Conference on Genetic and Evolutionary Computation*. ACM, 2010, pp. 257–264.

[56] F. Zhang, S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: An evolutionary learning approach," in *Machine Learning: Foundations, Methodologies, and Applications*. Springer, 2021, DOI: 10.1007/978-981-16-4859-5, pp. XXXIII+338 pages.

[57] N. F. McPhee, N. J. Hopper *et al.*, "Analysis of genetic diversity through population history," in *Proceedings of the Genetic and Evolutionary Computation Conference*, vol. 2. Citeseer, 1999, pp. 1112–1120.

[58] A. Ekárt and S. Z. Németh, "A metric for genetic programs and fitness sharing," in *Proceddings of the European Conference on Genetic Programming*. Springer, 2000, pp. 259–270.

[59] 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 *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[60] S. M. Gustafson, "An analysis of diversity in genetic programming," Ph.D. dissertation, University of Nottingham, 2004.