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 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 algorithms 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.**

***Impact Statement*—Genetic programming is a widely used approach for learning scheduling heuristics in scheduling problems. However, the training process is often time-consuming due to extensive workshop simulations. Although phenotype-based surrogate models have been proposed to estimate the quality of heuristics and reduce training time, the samples used in these models may lack representativeness, leading to reduced prediction accuracy. This paper proposes a surrogate sample selection method that considers both the phenotype and genotype of scheduling heuristics. By applying this method, the training time can be reduced by approximately 30–35%, enabling industrial decision-makers to obtain effective scheduling solutions more efficiently and enhance production performance. Furthermore, the method is generic and applicable to a wide range of problems.**

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

This work was supported in part by the National Natural Science Foundation of China (62206255), Natural Science Foundation of Henan (252300421501), Young Talents Lifting Project of Henan Association for Science and Technology (2024HYTP023), Frontier Exploration Projects of Longmen Laboratory (LMQYTSKT031). (*Corresponding author: Ke Chen.*)

Luyao Zhu was with the School of Electrical and Information Engineering, Zhengzhou University, Zhengzhou 450001, China, and is with the Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail:luyao.zhu@vuw.ac.nz).

Xiaodong Zhu and Ke Chen are with the School of Electrical and Information Engineering, Zhengzhou University, Zhengzhou 450001, China and also with Longmen Laboratory, Luoyang 471000, China (e-mail: Zhu_xd@zzu.edu.cn, chenkezixf@zzu.edu.cn).

Fangfang Zhang, Mengjie Zhang are with the Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand (e-mail: fangfang.zhang@vuw.ac.nz, mengjie.zhang@vuw.ac.nz).

I. Introduction

Dynamic flexible job shop scheduling (DFJSS) [1] is an important combinatorial optimisation problem, which aims to solve challenging and practical scheduling tasks, such as resource allocation in grid computing [2] and operation processing in manufacturing [3]. In DFJSS, some jobs need to be effectively processed by several machines in dynamic situations. Compared to traditional job shop scheduling (JSS) [4], DFJSS is more complex due to interactions among multiple decision-making processes in dynamic environments, such as the arrival of new jobs [5]. Two decisions need be made simultaneously: *machine assignment*, which assigns a ready operation to a machine, and *operation sequencing*, which selects the next operation for an idle machine.

DFJSS is an NP-hard problem [6]. *Exact optimisation approaches* [7], [8] and *Approximate solution optimisation approaches* [9], [10], perform poorly in DFJSS since they are not able to handle dynamic events efficiently. *Scheduling heuristics* [11], e.g., dispatching rules [12], [13], utilise 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 for an idle machine and a *routing rule* to assign operations to different machines. However, the design of scheduling heuristics normally needs domain expertise, prior experience, and an expensive trial and error process. Moreover, manually designed scheduling heuristics are often effective only in specific scenarios. GP [14] has been widely used for automatic generation of scheduling heuristics in DFJSS [15]. However, the evaluations of GP individuals for DFJSS are normally based on simulations, which are time-consuming, especially in large scale scenarios.

Surrogate models [16] have been extensively applied to reduce computation cost in evolutionary algorithms including

GP in DFJSS [17]. Unlike general evolutionary algorithms with vector-based representation, GP has a tree-based representation which makes it not straightforward to build surrogates. There are two commonly used surrogates in JSS. One is simulation simplification based, where the surrogate is built by a simplified simulation with a smaller number of jobs and machines than the original simulation. The other is K-nearest neighbour (KNN) based surrogates with phenotypic characterisation (PC), where KNN is used to find the most similar sample in the surrogate by calculating the difference of PCs. In this paper, we choose KNN as the surrogate model due to three reasons. Firstly, KNN-based surrogate is easy to build. Secondly, KNN is more efficient than the simplified simulation, and the prediction of fitness is less time-consuming. Lastly, the sample in the KNN-based surrogate is based purely on phenotype, and we found only considering phenotype in the KNN-based surrogate has limitations for fitness prediction. This is the main reason for choosing KNN, and we aim to investigate whether further combining genotype information can help improve the surrogate accuracy. In KNN-based surrogate, each GP individual's PC is a vector of ranks of machines and operations in fixed decision scenarios, which connects closely to its behaviour for decision making. 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, KNN will find the most similar sample based on PC, and this sample's fitness will be the estimated fitness of the GP individual.

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. Zhu et al. [18] proposed to group GP individuals based on unique PCs, and then selected the smallest individual in each group for real evaluations as samples to build a surrogate. Then, this surrogate will be used to estimate fitness of the remaining individuals in the current generation. In this way, the training time can be significantly reduced by decreasing the number of truly evaluated individuals. However, it relies on the assumption that individuals with the same phenotype exhibit similar fitness. Our observations, however, indicate that individuals with identical phenotypes may still have notably different fitness values. This discrepancy arises because the decision scenarios used to obtain PCs are often fewer and distinct from the scenarios used in simulations for actual fitness evaluations. Consequently, a single individual may not effectively represent with the same PC, potentially impacting surrogate accuracy. Additionally, using a large number of decision scenarios to calculate PCs for GP individuals in KNN surrogates is impractical, as it would considerably reduce KNN surrogate efficiency due to the computationally expensive distance calculations required for high-dimensional PC vectors. Therefore, exploring a more advanced, sample-aware surrogate-assisted GP approach for DFJSS is a promising direction for future work.

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 PC in DFJSS [21], [22]. The genotype of a GP individual normally refers to its structure including original genetic materials [23]. If the phenotype and genotype of two individuals are similar, their fitness have more potential to be closer. Thus, we consider adding the genotype factor to select more comprehensive surrogate sample.

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) We have proposed a new effective surrogate-assisted GP algorithm. Specifically, instead of utilising phenotype information of GP individuals only [18], we have developed a novel sample selection strategy for building KNN-based surrogates in DFJSS by considering both the phenotypes and genotypes of GP individuals. The results show that the proposed strategy can distinguish representative samples better to improve the surrogate accuracy. This is the first time to design a surrogate strategy that considers both the phenotype and genotype of a GP individual, and we have confirmed that using both phenotype and genotype is helpful for distinguishing GP individuals with flexible representation.
2) We have designed an effective way to represent the genotype characterisation of a GP individual by considering the frequency of features. The results show that the proposed novel way of representing genotype of variable-length GP programs can successfully further distinguish GP individuals with the same phenotype. In addition, The idea of designing genotype characterisation is generic, and is possible to be applied to other combinatorial optimisation problems.
3) The advantage of our proposed algorithm over the other algorithms on unseen test scenarios demonstrates the effectiveness and efficiency of the designed sample selection strategy used in GP, the potential to achieve smaller rules, and the ability to maintain a proper population diversity. In addition, these analyses have shown how the phenotype and genotype change along with generations, which provides a better understanding of GP Individuals and an available guidance for designing genotype and phenotype related strategies.

II. Background

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 $M(O_{ji}) \subseteq \pi(O_{ji})$. Machine assignment and

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2025.3562161

operation sequencing are required to be made concurrently under dynamic situations. This paper focuses on a common dynamic event, i.e., new job arrival. The information regarding a new job remains unknown until it is delivered to the shop floor. In this paper, we aim to minimise two commonly used objectives, respectively, which are actually the fitness of GP individuals. A smaller fitness indicates a better individual. Their calculations are demonstrated as follows.

Mean-flowtime (Fmean):

$$\frac{1}{n}\sum_{j=1}^{n}(C_j - r_j) \tag{1}$$

Mean-weighted-tardiness (WTmean):

$$\frac{1}{n}\sum_{j=1}^{n}\max\{0, C_j - d_j\}\cdot w_j \tag{2}$$

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.
- w_j is the weight of J_j.

The mathematical model of the DFJSS problem is shown as follows:

$$\min\ Fmean, WTmean \tag{3}$$

The principal constraints in DFJSS are formulated as follows:

1. Precedence Constraint: An operation O_{ji} cannot start until its preceding operation $O_{j(i-1)}$ is completed.

$$S_{ji} \geq C_{j(i-1)}, \quad \forall j, i > 1 \tag{4}$$

where S_{ji} is the start time of O_{ji} and $C_{j(i-1)}$ is the completion time of $O_{j(i-1)}$.

2. Exclusive Assignment Constraint: Each operation O_{ji} must be assigned to only one machine M_k from the set of candidate machines.

$$x_{jik} \in \{0,1\}, \quad \sum_{k\in M(O_{ji})} x_{jik} = 1, \quad \forall j, i \tag{5}$$

where $x_{jik} = 1$ if operation O_{ji} is assigned to machine M_k; otherwise, $x_{jik} = 0$.

3. Non-overlapping Constraint: A machine M_k can only process one operation at a time.

$$S_{ji} \geq C_{pq} \text{ or } S_{pq} \geq C_{ji}, \quad \forall (j,i) \neq (p,q),\ \forall k \tag{6}$$

where S_{ji} and C_{ji} represent the start and completion times of O_{ji}, and S_{pq} and C_{pq} represent the start and completion times of O_{pq} if both are assigned to machine M_k.

4. Non-interruptible Constraint: Once an operation O_{ji} starts on a machine M_k, it cannot be interrupted until its completion.

$$C_{ji} = S_{ji} + p_{ji}, \quad \forall j, i \tag{7}$$

where p_{ji} is the processing time of operation O_{ji}.

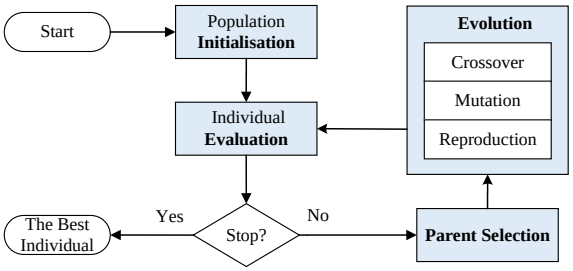

Fig. 1. The flowchart of GP.

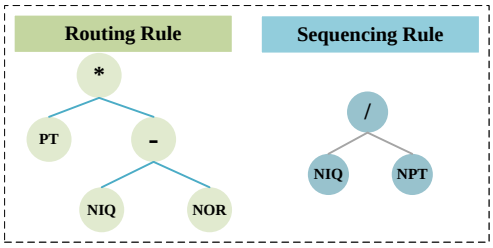

Fig. 2. An example of a GP individual with multi-tree representation for DFJSS.

B. Genetic Programming for DFJSS

GP has been successfully used to learn scheduling heuristics for JSS problems as a hyper-heuristic approach [24]–[29]. 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 [30]–[32] 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 [33]. 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.

1) Representation: This paper applies multi-tree representation to learn scheduling heuristics for DFJSS [33]. 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 represents a priority function of PT * (NIQ - NOR), where PT refers to the processing time required to operate on the specified machine, NIQ is the number of operations in the queue, and NOR denotes the number of operations remaining for a job. The sequencing rule is defined as NIQ / NPT, with NPT representing the processing

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	Feature (PT	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**	

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 choose the operation with the highest priority for processing on an idle machine.

III. RELATED WORK

A. Surrogate-Assisted GP for Job Shop Scheduling

Surrogate-assisted evolutionary algorithm has been widely studied to deal with expensive fitness evaluations [34]–[37] in the past decades. The fundamental idea of surrogates is to design a simple model to predict the fitness of individuals instead of using the expensive real evaluations.

a) Building Surrogates 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 [38]. The other is to develop a simplified simulation model (i.e., the simulation situation with few machines and jobs) as a surrogate model to predict the fitness of individuals [17], [39], [40]. 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.

b) Usages of Surrogates: There are two commonly used ways. First, most existing studies [39], [41], [42] 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 [18], [40]. Various surrogates with different accuracy were designed and used at different generations to reduce the training time without sacrificing the performance of GP algorithm [40]. A KNN-based sample aware algorithm was studied in [18] to not only reduce the training time but also improve the effectiveness of learned scheduling heuristics. However, [18] 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.

TABLE II
AN EXAMPLE OF THE CALCULATION OF THE PHENOTYPIC CHARACTERISATION OF A ROUTING RULE ACROSS THREE DECISION SCENARIOS AND EACH INVOLVING 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**
$3(M_3)$	$\underline{1}$	$\underline{1}$	

B. 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. The PC of a GP individual is a numeric vector that contains the rankings of machines and operations in various routing and sequencing decision scenarios, representing the decision behaviour of a pair of scheduling rules. Specifically, an examined routing or sequencing rule selects the highest-priority machine or operation, and each dimension of the PC represents the corresponding rank of the selected machine or operation according to a fixed reference rule (i.e., a manual rule). Table II illustrates the calculation of a routing rule's PC using three decision scenarios, each consisting of three machines. For example, in the first scenario, M_1 is assigned the highest priority according to the examined routing rule. Then, the rank of M_1 based on the reference rule (i.e., 2) is set as the PC value in the first scenario. Similarly, the PC values for other scenarios can be determined in the same way. Finally, PC of the examined routing rule is $[2, 3, 1]$. The PC for a sequencing rule can be determined similarly, but based on sequencing decision scenarios. In this way, each individual can obtain a PC to reflect its behaviours for decision marking. Note that 20 routing decision situations and 20 sequencing decision situations are involved to construct a PC as a 40-dimensional numerical vector. The decision situations are fixed to calculate the PCs of all individuals. More details can be found in [42].

b) Fitness Prediction: The KNN-based surrogate samples are actually from individuals in the previous generation. One sample consists of a PC vector and a fitness value of an individual. Individuals with close PC vectors show similar decision behaviour, which leads to comparable fitness. Fig. 3 illustrates how the KNN-based surrogate works. When using this surrogate to predict the fitness of a new individual ind whose PC is $[3, 1, 3, 2, 1, 1, 4, 2]$, first, we calculate the Euclidean distances between ind's PC and all samples' PCs in the surrogate, and then the KNN algorithm is applied to find the most similar sample based on the distances. It's obvious that the distance between ind and sample$_2$ is the smallest (i.e.,

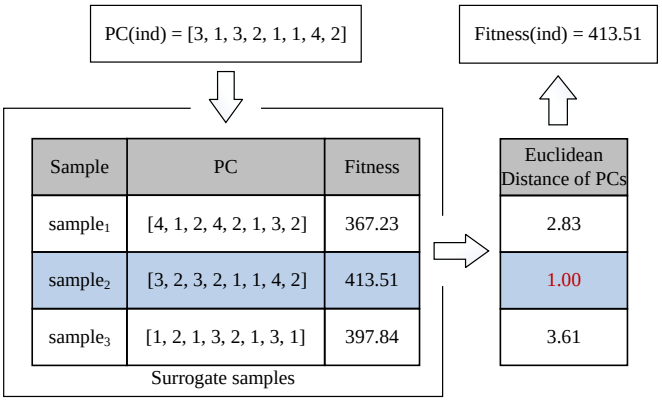

Fig. 3. An illustration of using KNN-based surrogate with PC for fitness prediction for GP individuals.

1), thus, sample_2 is the sample most similar to ind in the surrogate. Finally, the fitness of the most similar sample (i.e., 413.51) will be the estimated fitness of ind.

C. Sample Selection in KNN-based Surrogate

From the descriptions in Section III-B, it is evident that samples play a crucial role in KNN-based surrogate models. These samples are expected to represent diverse behaviours within the population to ensure accurate fitness estimation. However, research on KNN-based surrogate sample for DFJSS remains in its early stages, and selecting appropriate samples to construct the surrogate model is non-trivial. The original KNN-based surrogate simply selects all individuals from the previous generation as samples, without considering the characteristics of GP individuals [41]. In terms of improving sample selection, Zhang et al. [42] proposed incorporating more samples across generations. However, their method only focuses on the quantity of samples, without considering their quality. To address this, Zhu et al. [18] proposed grouping GP individuals based on unique PCs, then selecting one representative individual (typically the smallest) from each group for real evaluation. A surrogate model is then built using these selected samples to estimate the fitness of the remaining individuals in the current generation. This significantly reduces training time by reducing the number of individuals for real evaluation. However, this method relies on the assumption that individuals with the same phenotype share similar fitness, which may not always hold.

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 are different from the decision situations in the simulation for real fitness evaluations. Thus, an individual cannot 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.

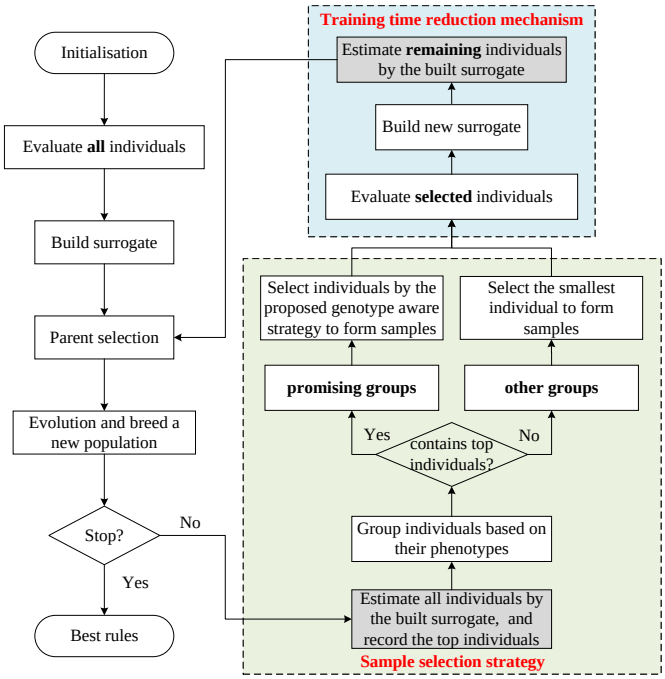

Fig. 4. Flowchart of the proposed algorithm.

A more advanced sample-aware surrogate-assisted GP for DFJSS is worth investigating. To this end, this paper aims to select more representative samples for surrogate-assisted GP in DFJSS. This will be the first time to select samples by considering both the phenotype and genotype of individuals.

IV. 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 training time reduction mechanism is highlighted in blue, which is to only evaluate some selected individuals instead of all individuals as samples to build surrogates and estimate the remaining individuals' fitness in the current generation. In the beginning, at *initialisation* stage, a population is randomly generated, and we evaluate all individuals at the *evaluation* stage and use them to build surrogate. Then, a new population will be bred by genetic operators with selected parents during *evolution* stage. If the stop condition is not achieved, the algorithm will move to the next generation, and the main steps of the evaluation of the newly generated population are as follows.

- First, the built surrogate in the previous generation will estimate the fitness of all the newly generated individuals, and some top individuals $topIndividuals$ according to their estimated 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 a group contains any individuals

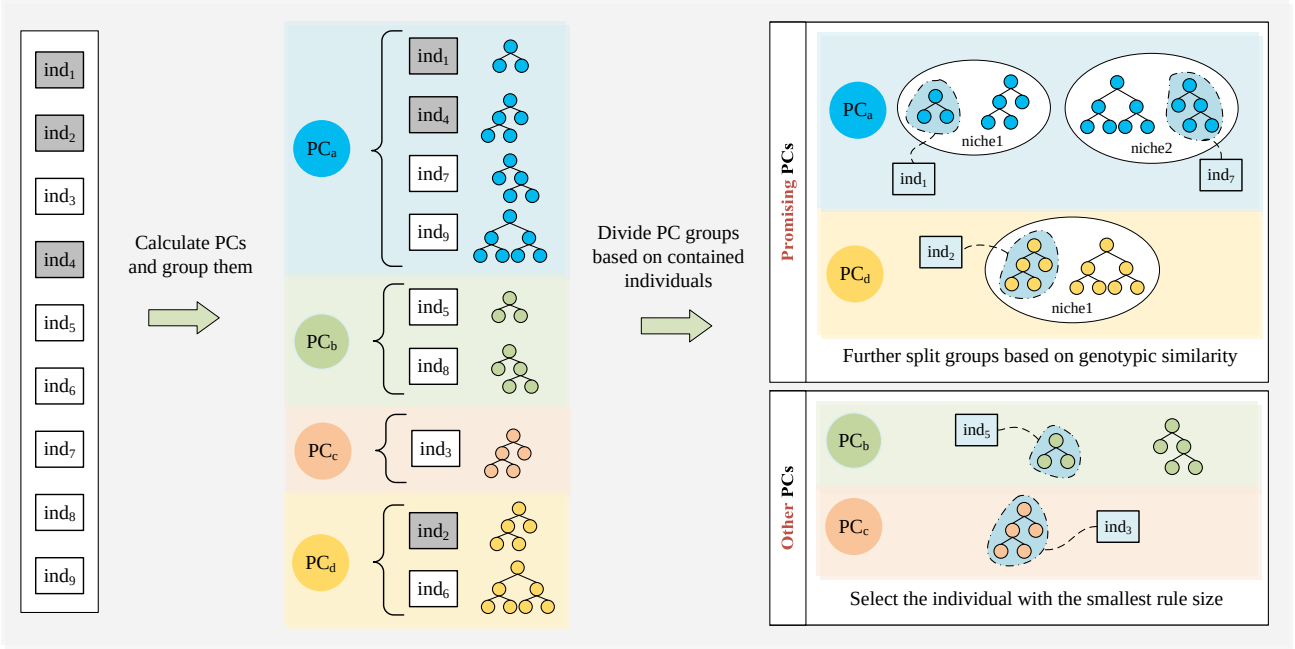

Fig. 5. An example of the work procedure of the proposed phenotype and genotype aware sample selection strategy.

in the $topIndividuals$ set, we define this group as a promising group. Thus, we name the groups as *promising groups* and *other groups*.

- Third, we use different strategies to select individuals/samples from promising groups and other groups for real evaluation. For promising groups, we select samples based on proposed phenotype and genotype based sample selection strategy with niching techniques from each group. For other groups, we will directly select the smallest individual from each group.
- Finally, these selected individuals/samples from promising groups and other groups will be truly evaluated, and then their information will be used to built a new surrogate to predict the remaining unevaluated individuals' fitness in the current generation.

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

Note that we use real evaluations for all individuals for the initialised population (the first generation) to get a good start point. We can also see that there are two built surrogates for different purposes, which is highlighted in grey. First, the built surrogate in the previous generation is used to estimate the fitness of all newly generated individuals to determine whether the individual's PC group is promising or not. Second, the built surrogate in the current generation will be used to estimate the fitness of the remaining individuals in the current generation for the following evolution process such as parent selection.

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 Aware Sample Selection Strategy for Promising Groups: Niching techniques [43] have been widely used to address multi-objective [44] and multi-modal [45] optimisation problems. These techniques divide the population into multiple niches, with each niche expected to contain similar individuals. This paper applies niching to separate individuals within the same promising group based on genotype for sample selection. The technique involves two parameters: capacity, which determines the number of individuals per niche, and radius, which controls niche coverage. Here, the capacity is set to infinity, and the radius is determined by genotype correlations, is explained in the next subsection.

An example of the proposed genotype based sample selection strategy with niching to further split individuals in promising groups into niches, is shown in the upper right corner of Fig. 5. In group PC_a, two niches are generated based on the genotypic similarity, and the genotypes of individuals in one niche are similar. The smallest rule in each niche (i.e., ind_1, ind_7) will be selected as a 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. The surrogate will assign identical fitness to individuals within the same niche, which helps increase population diversity. In this paper, tournament

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 IV-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** $samples$

selection is employed as the parent selection method, choosing the individual with the best fitness among a group of candidates. Since the surrogate assigns identical fitness to individuals within the same niche, more candidates in the tournament selection process may have the same fitness. This can promote a more diverse parent selection process since multiple candidates can have the same fitness, i.e., indicating the same chance to be selected, thereby enhancing the diversity of the generated population. Note that niching is applied only to promising PC groups, ensuring that only individuals within the same high-quality groups share identical fitness. This approach maintains population diversity while simultaneously facilitating the selection of high-quality parents.

Algorithm 1 shows the proposed genotype aware sample selection strategy for promising groups. 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 grouped together (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 centre, then the genotype correlation between the centre 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 centre point, the remaining individual with the smallest rule size outside the niche will be considered as a new centre 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 centres 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 centre individual of each niche will be the smallest individual in that niche.

2) Sample Selection for Other Groups: Since the individuals in other groups are less effective than the individuals in promising groups, which have less potential to be parent in the subsequent stage (i.e., parent selection), further splitting them to select samples will not improve 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 [24]. Moreover, it has been demonstrated that terminal information is useful to guide initialisation and search in GP [46], [47]. Thus, designed genotype has potential to further divide individuals with same PC. 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). MWT refers to a machine's waiting time. W and PT are the weight of a job and the processing time of an operation on a designated machine. For example, the frequency of MWT in the routing rule is calculated by $10/(10+2+6) = 0.56$. Similarly, the frequency

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

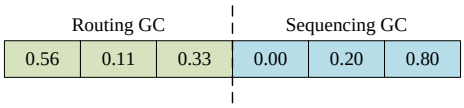

Fig. 6. Example of designed genotypic characterisation of an individual.

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 [48] to measure the genotypic similarity 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 two main advantages. One is that proposed algorithm considers both phenotype and genotype, which can make the selected samples more representative. The other is making the centre of each niche to be the smallest rule as a reference for GC calculations or sample selections can bring three benefits. Firstly, small rules have the potential to be more interpretable and preferable in practical situations. Secondly, small rules might have better generalisation ability to avoid overfitting issues. Thirdly, small rules tend to take less time for priority calculations, which can make decisions efficiently.

V. EXPERIMENT DESIGN

A. Simulation Model

Simulation is an effective technique to study complex dynamic JSS problems [49]. Referring to widely used DFJSS simulations [50], [51], 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 [52], 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.

$$\lambda = \mu * P_M / p \tag{8}$$

where λ defines the expected utilisation 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 algorithms will change the training instance in each generation to enhance the generalisation capabilities of the evolved scheduling rules [31], [53]. To verify the robustness of proposed algorithm, 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. In addition, our proposed algorithm is also compared with manually designed rules that are popularly used for scenarios. It is important to note that due to the dynamic and large-scale nature, the common exact methods and heuristic methods cannot be directly used for solving dynamic and/or large-scale job shop scheduling problems. The details are shown as follows.

1) Manual Rules: For all scenarios, we use WIQ to allocate an operation to a machine with smallest workload. For scenarios with objective mean-flowtime, we use shortest processing time (SPT) as the sequencing rule. For scenarios with objective mean-weighted-tardiness, we use WATC rule, i.e., weighted apparent tardiness cost, as the sequencing rule [54].

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

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

2) GP: The GP algorithm with multi-tree representation [33] is selected as a baseline algorithm to learn two rules simultaneously for DFJSS for comparison.
3) SGP_Ran: SGP_Ran simply selects samples *randomly* from the population is included for comparison to show the importance of sample aware surrogate building.
4) SGP_PC [18]: 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.
5) 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 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 [32], [55], 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, more details can be found in supplementary materials.

VI. 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 and robustness 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 "↑", "↓", "≈" 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 [17]. Borrowing the idea in [17], 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 of GP, SGP_Ran, SGP_PC and SGP_PCGC with the same training time and Manual Rules in six scenarios. Clearly, the scheduling rules learned by GP based algorithms are significantly better than manually designed rules in all scenarios. This indicates the effectiveness of learning scheduling heuristics by GP based algorithms. Overall, the results show that the phenotype

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2025.3562161

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) AND MANUAL RULES ACCORDING TO 30 INDEPENDENT RUNS IN SIX SCENARIOS.

Scenarios	Manual Rules	GP	SGP_Ran	SGP_PC	SGP_PCGC
<Fmean, 0.75>	436.46	336.57(1.61)(↑)	335.52(1.0)(↑)(↑)	335.37(0.71)(↑)(↑)(≈)	**335.09(0.55)**(↑)(↑)(≈)(≈)
<Fmean, 0.85>	502.30	385.47(2.29)(↑)	385.82(3.00)(↑)(≈)	384.10(0.98)(↑)(↑)(↑)	**383.62(0.79)**(↑)(↑)(↑)(↑)
<Fmean, 0.95>	763.85	554.93(7.55)(↑)	552.58(3.85)(↑)(≈)	549.11(3.32)(↑)(↑)(↑)	**547.81(1.67)**(↑)(↑)(↑)(↑)
<WTmean, 0.75>	121.32	27.24(1.41)(↑)	27.03(0.92)(↑)(≈)	26.97(1.75)(↑)(↑)(≈)	**26.63(1.05)**(↑)(↑)(↑)(≈)
<WTmean, 0.85>	220.09	75.61(2.41)(↑)	75.91(2.39)(↑)(≈)	74.46(1.93)(↑)(↑)(↑)	**74.24(1.63)**(↑)(↑)(↑)(≈)
<WTmean, 0.95>	572.37	297.94(11.53)(↑)	300.39(12.00)(↑)(≈)	**291.84(5.11)**(↑)(↑)(↑)	292.59(8.10)(↑)(↑)(↑)(≈)
Win / Draw / Lose	6 / 0 / 0	6 / 0 / 0	5 / 1 / 0	2 / 4 / 0	N/A
Average Rank	5	3.17	2.86	2.12	**1.85**

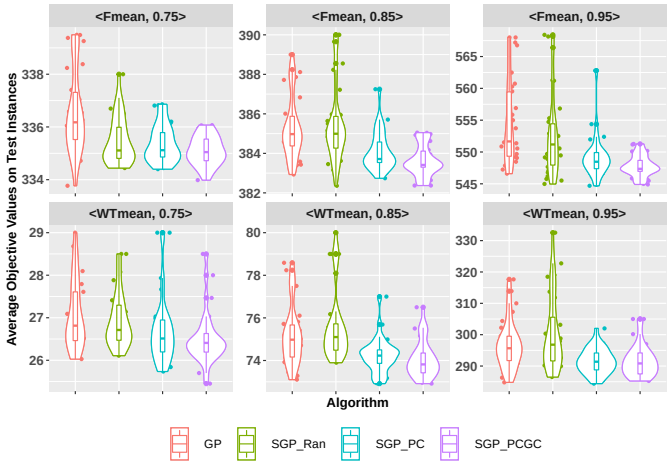

Fig. 7. The violin plots of the average objective values on test instances of GP, SGP_Ran, SGP_PC and SGP_PCGC with the same training time (in minutes) according to 30 independent runs in six scenarios.

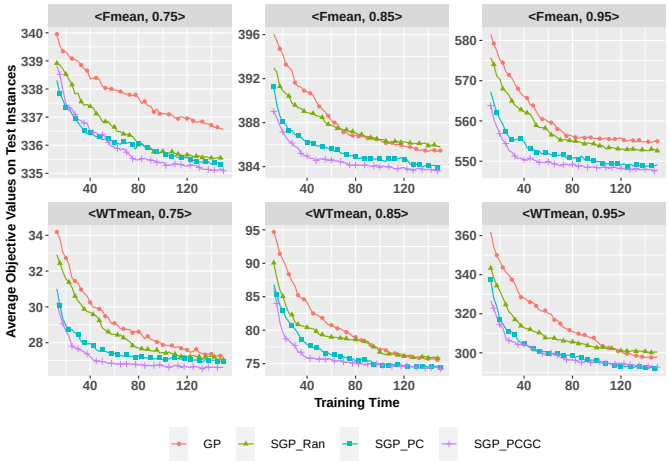

Fig. 8. The curves of average objective values according to 30 independent runs on test instances of GP, SGP_Ran, SGP_PC and SGP_PCGC with the same training time (in minutes) in six different scenarios.

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 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>). In addition, SGP_PCGC shows the smallest distribution spread, indicating higher stability and lower variance in objective values compared to other algorithms. Overall, the results demonstrate that proposed sample selection strategy in SGP_PCGC can distinguish representative samples to build surrogate with high accuracy.

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>). This is attributed to the surrogate model built using the new sample selection strategy, which effectively identifies promising individuals and maintains population diversity. In addition, we can see that SGP_PCGC can sustain the advantage into the late 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

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)(↑)(≈)

heuristics and achieve better scheduling heuristics faster than its counterparts in most scenarios with the same 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.

1) Quality of Learned Scheduling Heuristics with the Same Number of Generations: Table VII shows the objective values 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. For GP in DFJSS, the most time-consuming part is the fitness evaluations of individuals. Thus, $O(\text{time}) = \text{evals} * O(\text{eval}) = \text{evals} * \text{ops} * O(\text{ds}) = \text{evals} * \text{ops} * O((M + 100) * \text{rulesize}) = \text{evals} * \text{ops} * O(\text{rulesize})$, where evals, ops, and ds are the number of evaluations, operations, and decision situations, respectively. M (i.e., 10 in this paper) is the number of machines, and 100 is the maximal number of operations in a queue of a machine (the simulation will terminate if the queue size exceeds 100). Each operation will have routing decisions to be allocated to a machine, and sequencing decisions to be selected for processing next. rulesize is the number of nodes of a GP individual.

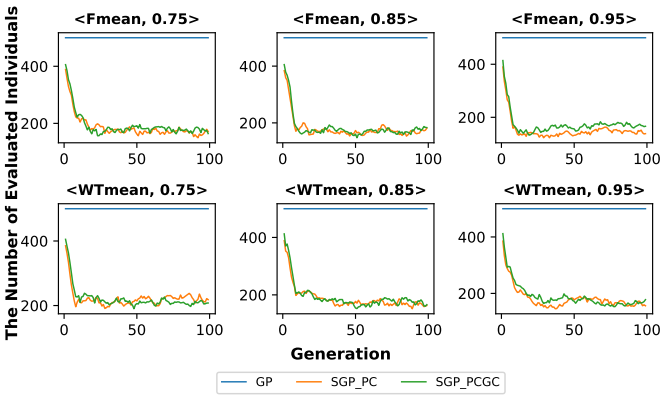

Fig. 9. The curves of the number of individuals with real evaluations of GP, SGP_PC and SGP_PCGC during the training process over 30 independent runs in six DFJSS scenarios.

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 factor in determining the training efficiency of GP algorithms 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.

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

This article has been accepted for publication in IEEE Transactions on Artificial Intelligence. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TAI.2025.3562161

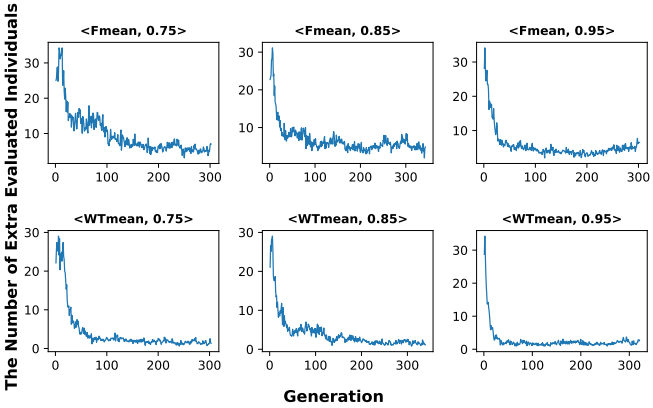

Fig. 10. The curves of the number of extra evaluated individuals by SGP_PCGC compared with the mechanism of SGP_PC during the training process over 30 independent runs in six DFJSS scenarios.

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. The reason is that 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.

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).

The black line represents the fitting curve of the data, showing the overall trend between genotype correlation and fitness differences in different generations. 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. In the early stages (i.e., Generation 1 and 5), from the black fitting curve, we can find the lower the genotype correlation, the bigger differences between the fitness values, which shows the effectiveness of designed genotype. Observing the density

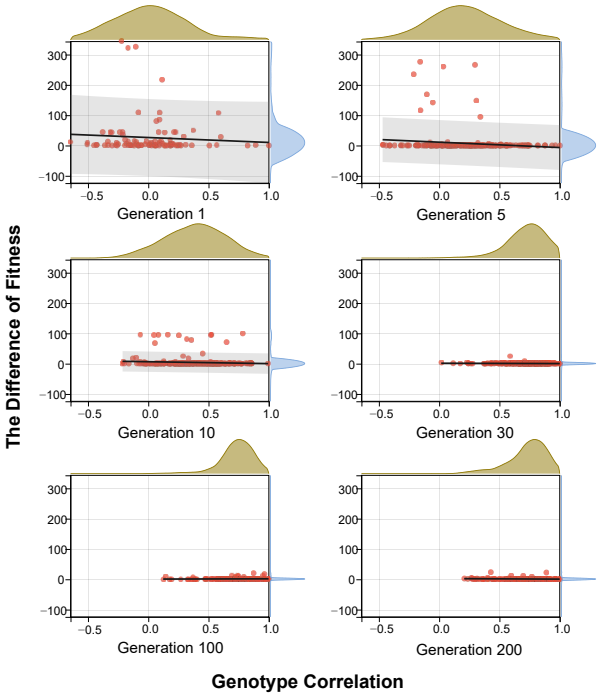

Fig. 11. Genotype correlation and the difference of fitness in the scenario <Fmean, 0.85> of one GP run.

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 more similar during the evolutionary process. This is because individuals tend to use similar terminals along with the evolution. In addition, the density plots on the right side of each subplot illustrate that the fitness differences between individuals with the same PC is getting smaller along with generations. Compared with the plots in the early stages, we can find the consistent pattern, the higher the genotype correlation, the less different fitness values. Overall, the results show that designed genotype can distinguish individuals with different fitness in the same PC group.

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 centre individuals from each niche by taking the centre 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 the different niche(s)

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)(≈)(↑)(≈)

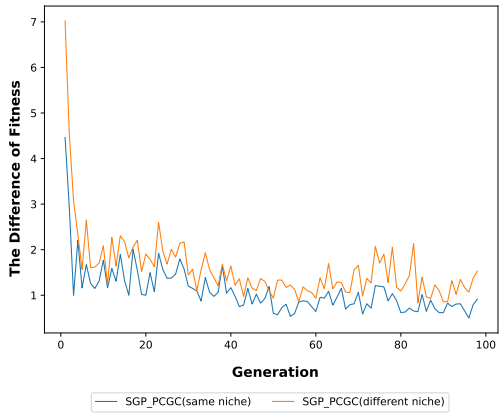

Fig. 12. The fitness difference between individuals in the same and different niche(s) of SGP_PCGC in the scenario <Fmean, 0.85> of 30 runs.

of SGP_PCGC in the scenario <Fmean, 0.85>. According to our proposed algorithm, individuals with similar genotype will be moved in the same niche, whereas individuals with different genotypes will be in different niches. 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, which also demonstrates the effectiveness of designed genotype to distinguish GP individuals.

VII. 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, SGP_PC and 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 VI-A and VI-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.

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)(≈)(≈)

*b) Sizes of Learned Scheduling Heuristics **with the Same Number of Generations**:* Section VI-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 it 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.

B. Population Diversity

Diversity is an important criterion for tracking the population status during the evolutionary process. Throughout the GP literature, diversity is consistently emphasised as crucial for avoiding premature convergence to local optima [23], [56], [57]. This paper uses entropy to measure population diversity, which is calculated as

$$entropy = -\sum_{c \epsilon C} \left(\frac{|c|}{|inds|}\right) log\left(\frac{|c|}{|inds|}\right) \quad (9)$$

where C is the set of clusters obtained using the DBScan clustering algorithm [58] with the phenotypic distance measure and a cluster radius of 0. A larger entropy means a higher diversity of the population.

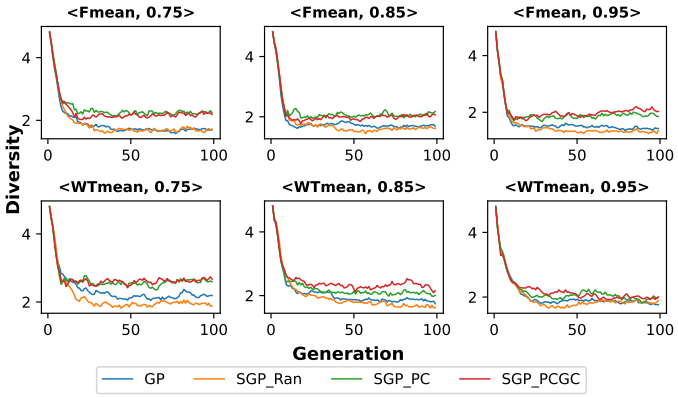

Fig. 13. The curves of phenotypic diversity values in every generation of GP, SGP_Ran, SGP_PC and SGP_PCGC according to 30 independent runs in six scenarios.

Since the number of generations may 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 analyse the population diversity. Fig. 13 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 [59]. After around generation 10, SGP_PC and SGP_PCGC have a higher diversity value than GP and SGP_Ran in all scenarios. Since tournament selection is the primary mechanism for controlling selection pressure in GP, it implies that more diverse individuals are selected to produce 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 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.

VIII. 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 cloud scheduling and online packing. In addition, the proposed genotype of individuals becomes more similar along with the generations, which leads to more effect in the early stage than the later stage. We'll further investigate more advanced strategies to select representative individuals for surrogate building during in the whole evolutionary process.

References

[1] F. Zhang, G. Shi, Y. Mei, and M. Zhang, "Multiobjective dynamic flexible job shop scheduling with biased objectives via multitask genetic programming," *IEEE Transactions on Artificial Intelligence*, vol. 6, no. 1, pp. 169–183, 2025.

[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] D. Djurdjanovic, L. Mears, F. Akhavan Niaki, A. Ul Haq, and L. Li, "Process and operations control in modern manufacturing," in *International Manufacturing Science and Engineering Conference*, vol. 50749. American Society of Mechanical Engineers, 2017, p. V003T04A057.

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

[5] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Multitask multiobjective genetic programming for automated scheduling heuristic learning in dynamic flexible job-shop scheduling," *IEEE Transactions on Cybernetics*, vol. 53, no. 7, pp. 4473–4486, 2023.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Survey on genetic programming and machine learning techniques for heuristic design in job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 28, no. 1, pp. 147–167, 2024.

[16] 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.

[17] 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.

[18] 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*, 2023, pp. 384–392.

[19] W. Banzhaf, "Genotype-phenotype-mapping and neutral variation—a case study in genetic programming," in *International Conference on Parallel Problem Solving from Nature*. Springer, 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 *Proceedings of the European Conference on Genetic Programming*. 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] 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.

[25] 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.

[26] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Task relatedness-based multitask genetic programming for dynamic flexible job shop scheduling," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 6, pp. 1705–1719, 2023.

[27] 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.

[28] L. Zhu, F. Zhang, M. Feng, K. Chen, X. Zhu, and M. Zhang, "Crossover operators between multiple scheduling heuristics with genetic programming for dynamic flexible job shop scheduling," in *Proceedings of IEEE Congress on Evolutionary Computation*. IEEE, 2024, pp. 1–8.

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

[30] 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.

[31] 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.

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

[33] 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.

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

[35] 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.

[36] 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.

[37] 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.

[38] 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.

[39] 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.

[40] 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.

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

[42] 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*, vol. 27, no. 5, pp. 1192–1206, 2022.

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

[44] 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.

[45] 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.

[46] M. urasević, F. J. Gil-Gala, and D. Jakobović, "To bias or not to bias: Probabilistic initialisation for evolving dispatching rules," in *European Conference on Genetic Programming (Part of EvoStar)*. Springer, 2023, pp. 308–323.

[47] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Genetic programming with adaptive search based on the frequency of features for dynamic flexible job shop scheduling," in *European Conference on Evolutionary Computation in Combinatorial Optimization (Part of EvoStar)*. Springer, 2020, pp. 214–230.

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

[49] 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.

[50] 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.

[51] 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.

[52] 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.

[53] 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.

[54] B. Chen and T. I. Matis, "A flexible dispatching rule for minimizing tardiness in job shop scheduling," *International Journal of Production Economics*, vol. 141, no. 1, pp. 360–365, 2013.

[55] 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.

[56] 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.

[57] 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.

[58] 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.

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