Genetic Programming with Cluster Selection for Dynamic Flexible Job Shop Scheduling

Meng Xu, Yi Mei, Fangfang Zhang✉, Mengjie Zhang
School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand
{meng.xu, yi.mei, fangfang.zhang, mengjie.zhang}@ecs.vuw.ac.nz

2022 IEEE Congress on Evolutionary Computation (CEC) | 978-1-6654-6708-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/CEC55065.2022.9870431

***Abstract*—Dynamic flexible job shop scheduling is a challenging combinatorial optimisation problem, that aims to optimise machine resources for producing jobs to meet some goals. There are two important kinds of decisions that the scheduling process needs to make under dynamic environments, i.e., the routing decision for machine assignment and the sequencing decision for operation ordering. Genetic programming hyper-heuristic has been successfully applied for solving the dynamic flexible job shop scheduling problem with the advantage of automatically evolving good scheduling heuristics. Parent selection is an important process for genetic programming, intending to select good individuals as parents to generate offspring for the next generation. Traditional genetic programming methods select parents for crossover based on only fitness (e.g., tournament selection). In this paper, a new parent selection (i.e., cluster selection) method is proposed to select parents not only with good fitness but also with different behaviours. The proposed cluster selection is combined with genetic programming hyper-heuristic to study whether considering different behaviours in parent selection will improve the effectiveness of the evolved scheduling heuristics. The experimental results show that increasing the number of unique behaviours in the population cannot help evolve effective scheduling heuristics. Further analysis shows that considering behaviour to select parents does increase the number of unique behaviours in the population. However, it gives individuals with poor fitness more probability to be selected to generate offspring. This might be the reason why the proposed method cannot outperform the baseline method.**

***Index Terms*—dynamic flexible job shop scheduling, genetic programming, cluster selection, diversity**

I. Introduction

Dynamic flexible job shop scheduling (DFJSS) [1] is an important combinatorial optimisation problem, that aims to allocate operations of jobs to a set of machines to optimise some goals, such as minimise mean flowtime, minimise max tardiness, and so on. There are two important kinds of decision points where DFJSS needs to make suitable decisions, that are the routing decision points and the sequencing decision points [2]. A routing decision point is defined as the moment once an operation becomes ready, and a machine is needed to be recommended to process the ready operation. A sequencing decision point represents the moment once a machine becomes idle, and an operation is needed to be selected from the waiting queue of this machine to be processed next. For the DFJSS problem, jobs arrive dynamically which means the system state is different for different decision points which makes the problem complex and difficult [3].

For solving the DFJSS problem, some early proposed methods for solving job shop scheduling problems, such as exact methods (e.g., mathematical programming [4] and branch-and-bound [5]) and some heuristic search methods (e.g., genetic algorithm [6] and tabu search [7]) are not applicable, because of the large scale and high dynamic particularity of DFJSS. Recently, scheduling heuristics are widely and popularly used to handle the DFJSS problem [8] with the advantages of reacting in real-time based on the latest information. However, designing good scheduling heuristics requires domain knowledge from expert and the design process is time-consuming.

Genetic programming hyper-heuristic (GPHH) [9] has been widely applied to automatically evolve scheduling heuristics for solving job shop scheduling problems [10]–[12] because of a number of advantages. Firstly, the flexible representation of GPHH gives more possibilities to explore scheduling heuristics with different structures. Secondly, GPHH does not require rich domain knowledge. Thirdly, scheduling heuristics evolved by GPHH have good interpretability. GPHH generates new individuals following a process of fitness evaluation, parent selection, genetic operators (i.e., reproduction, crossover, and mutation) until the termination criteria is reached, and then retains the best individuals obtained as the final output. In this process, parent selection performs an important role to identify promising individuals carrying good genes [13].

Currently, the widely used parent selection method for GPHH is tournament selection [14]. Tournament selection selects an individual as the parent depending on only one criterion, i.e., the fitness of the individual. On the other hand, GP tends to converge quickly, and the population tends to be filled with high-fitness individuals with very similar structures/behaviours. As a result, it is likely that individuals with similar structures/behaviours will be selected as the parents for crossover. Some papers study the selection pressure of tournament selection [15], [16] in order to get good results. However, still, only the fitness is used as the basis for comparison. A good parent selection method needs to support the evolutionary process of producing high-quality offspring but also ensure the diversity of the population. The phenotypic (behaviour) diversity is more widely studied than genotypic diversity because the former can show the characterisation of an individual (scheduling heuristic) more accurately than the latter. In DFJSS, the phenotypic diversity is usually denoted

978-1-6654-6708-7/22/$31.00 ©2022 IEEE

by the unique phenotypic characterisation (PC) [17] in the population. The PC of an individual is composed of a vector of decisions made by this individual on some decision points. Some studies increase the phenotypic diversity in the population by the niching technique which can clear individuals with similar PC but poor fitness [18]. Some studies increase the phenotypic diversity of the population by proposing new parent selection methods, such as batch tournament selection [19] and lexicase selection [20], [21]. These methods select parents based on different training subsets or different orders of training cases. However, the above methods cannot always ensure that the selected parents for crossover have dissimilar behaviours. For crossover, if the selected two parents carry genes that behave similarly, it is easy to produce offspring that can at best maintain a similar level to the parents and the search cannot progress sufficiently. Therefore, it is important to make sure that the two parents selected for crossover have dissimilar behaviours.

Based on the above consideration, the overall goal of this paper is to propose a new parent selection method to increase the phenotypic diversity in the population. The new selection method is designed for crossover and is expected to select two parents with different behaviours. To be specific, this paper has the following research objectives:

- Design a new selection method for crossover that selects parents not only based on fitness but also based on behaviour.
- Develop a new GPHH framework with the newly proposed selection method to automatically evolve scheduling heuristics.
- Analyse whether considering different behaviours in parent selection will improve the effectiveness of the evolved scheduling heuristics?

II. Background

A. Dynamic Flexible Job Shop Scheduling

In the shop floor, there are a set of heterogeneous machines $\mathcal{M} = \{M_1, ..., M_p, ..., M_m\}$ which have different processing rates. The processing rate of the machine M_p is represented as $\gamma(M_p)$. These machines are placed in fixed locations $\mathcal{L} = \{L_1, ..., L_p, ..., L_m\}$. In a DFJSS problem, the jobs $\mathcal{J} = \{J_1, J_2, ..., J_n\}$ arrive at the shop floor dynamically. Each job J_i has a sequence of operations $\mathcal{O}_i = \{O_{i1}, O_{i2}, ..., O_{ik}\}$. Each operation O_{ij} can only be processed by one of its optional machine $\pi(O_{ij}) \subseteq \mathcal{M}$ and its processing time $\sigma_{M_p}(O_{ij}) = l_{ij}/\gamma(M_p)$ depends on the machine $M_p \in \pi(O_{ij})$ that processes it and the workload l_{ij} it has. The release time r_j, due date d_j, weight w_j of the job J_i is not known until the job arrives at the shop floor. In addition, the transportation time $T_{p,q} = dis_{p,q}/v$ between machines/entry/exit depends on the distance $dis_{p,q}$ between the location L_p of the current machine/entry/exit and the location L_q of the selected machine/entry/exit, and the transport speed v. DFJSS needs to decide which machine to use to process the ready operation and which operation to choose to process next by an idle machine.

In this paper, we consider three objectives which are max flowtime (Fmax), mean flowtime (Fmean), and mean weighted flowtime (WFmean). The calculations of the three objectives are shown as follows.

- Minimise $Fmax = \max\{C_1 - r_1, ..., C_n - r_n\}$
- Minimise $Fmean = \frac{\sum_{j=1}^{n}(C_j - r_j)}{n}$
- Minimise $WFmean = \frac{w_j \times \sum_{j=1}^{n}(C_j - r_j))}{n}$

where, C_j denotes the completion time of the job J_j.

Some main assumptions and constraints are described as follows.

- Scheduling is non-preemptive, in other words, once the processing of an operation has started it cannot be stopped or paused until it is completed.
- The machines will not break down and their processing rates and locations will not change during the entire scheduling process.
- One machine can process a maximum of one operation at a time.
- Once the jobs have arrived on the shop floor, the sequence of operations for each job is predetermined. An operation can only be processed once all previous operations have been completed and have arrived at its chosen machine.
- For each operation, only one of its candidate machines can process it.

B. Genetic Programming Hyper-heuristics for DFJSS

GPHH [9], with the advantages of flexible representations, has been successfully applied to evolve scheduling heuristics for solving DFJSS problems. GPHH can be divided into two parts, which are the training process and the test process. After the training process, the scheduling heuristic with the best fitness wins, and it is treated as the output of GPHH. Then, the test process measures the performance of the output scheduling heuristic.

GPHH starts with a population of randomly generated individuals (scheduling heuristics), and then determines the fitness of each individual based on its ability to perform the given task (e.g., the DFJSS problem), applying Darwinian natural selection (survival of the fittest) to determine the winning individual. The population also mimics the breeding process by selecting parents first and then combining or generating genes of individuals based on the reproduction, crossover, and mutation, until a predetermined stopping criterion is reached.

There have been many studies using GPHH for solving DFJSS problems. GPHH with multi-tree representation was proposed to evolve the sequencing rule and the routing rule for DFJSS simultaneously in [22]. In [23], GPHH with the surrogate was used to automatically evolve scheduling heuristics for DFJSS. Some works use GPHH to solve multi-objective DFJSS to meet some goals at the same time [24], [25].

C. Related Work

For the crossover operator, tournament selection is used two times to select two parents to generate offspring. However, as the evolutionary process goes on, the survival individuals

based on fitness would have similar structures or behaviours to each other. Even using different sampling strategies to select candidate individuals in tournament selection has little impact on selection [16]. In this case, it may cause the GPHH to converge to the local optimum so that no better solution can be found. In order to avoid this situation, some new techniques and selection methods are devoted to improving the diversity of parents selected, that is, increasing the diversity of the population.

In DFJSS, phenotypic diversity can reflect the characterise better than genotypic diversity. The phenotypic characterization (PC) [17] is usually used to represent the behaviour (phenotype) of a scheduling heuristic. The phenotypic representation is composed of a vector of numerical values, which denotes the decisions made by the scheduling heuristic on the selected decision points. Niching is a widely used technique to clear individuals with the same behaviour (PC) but poor fitness [18]. Niching can increase phenotypic diversity. However, it cannot always select parents with more diverse behaviours.

Some studies propose new parent selection methods to improve the phenotypic diversity. Batch tournament selection is such a method that selects parents based on different training subsets [19]. However, in batch tournament selection, there is no criterion to completely ensure that the two individuals selected must be dissimilar in behaviour. Lexicase selection [26] was proposed to select parents by following different orders of training cases which can increase the phenotypic diversity of the selected parents. It has been proven that lexicase selection can obtain better performance than tournament selection on some problems [20], [21]. However, using multiple training cases in DFJSS to do evaluation is time-consuming. Additionally, if the order of training cases used in the two parent selections for crossover is the same, the same individual will be selected as parents.

In [15], a clustering tournament selection method was proposed for tuning the selection pressure dynamically and automatically along with evolution. In this method, individuals are divided into different groups, and each group is assigned a fitness. The group with the best fitness wins the tournament, and an individual in this group is randomly selected as a parent to produce offspring. The method proposed in [15] obtained good results than traditional tournament selection. However, sometimes the selected group may have few individuals which might lead GPHH to converge to a local optimum.

Based on the above limitations, a new parent selection method is proposed to help GPHH select parents with good quality and dissimilar behaviours for crossover.

III. The New Approach

A. The Overall Framework

The overall framework of the proposed GPCS for DFJSS is shown in Fig. 1. As with traditional GPHH algorithms, population initialisation, fitness evaluation, elitism selection, parent selection, reproduction, crossover, and mutation are all main processes. The main innovation in this paper is to use

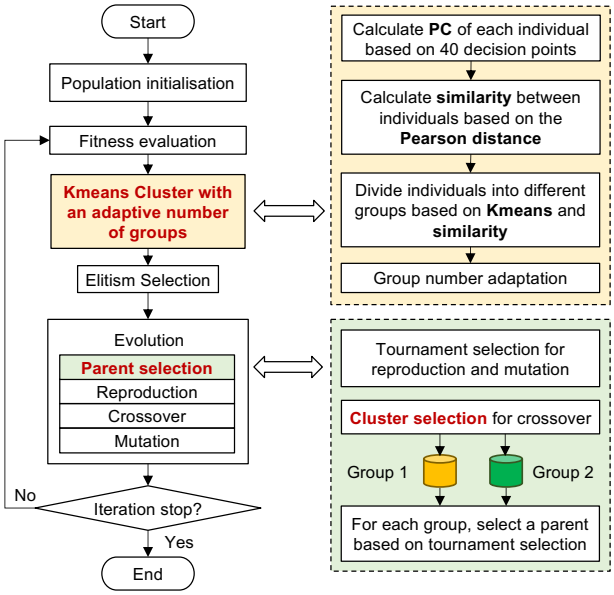

Fig. 1. The flowchart of the proposed GPCS method.

cluster selection to replace tournament selection to select a pair of more diverse parents in the evolution process for crossover.

In the cluster selection, the Kmeans cluster strategy with an adaptive number of groups is proposed and used after fitness evaluation to divide individuals in the population into different groups based on their behaviour similarity. Each group contains a number of individuals with similar behaviours. Then, when two parents are needed for crossover, the cluster selection randomly selects two different groups, we call them *preliminary winners one* and *preliminary winners two*. After that, tournament selection is used to select two individuals from *preliminary winners one* and *preliminary winners two*, respectively. These two individuals are the parents used for crossover.

B. Behaviour Similarity Estimation

In this paper, the *decision priorities* on 20 sequencing decision points and 20 routing decision points [17] in a fixed instance is used to estimate how similar two individuals behave with each other. The *decision priority* denotes the index sequencing of the candidate operations/machines made by the individual (scheduling heuristic) on the decision point. For example, if we have 3 sequencing decision points and 3 routing decision points, each with 7 candidate operations/machines, and the index of candidate operation/machine is between 1 and 7. Then the *decision priorities* for an individual x on these decision points can be represented as Fig. 2.

It can be seen from Fig. 2, each row represents the *decision priority* on each decision point which is the priority order of candidate operations/machines. The whole table denotes the *decision priorities* of the individual x. In addition, the first column shows the final decisions on these decision points, which represents the PC of the individual x. We can estimate the behaviour similarity between two individuals based on the

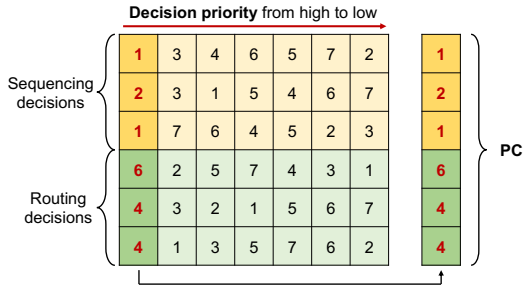

Fig. 2. An example of the *decision priorities* on 3 sequencing decision points and 3 routing decision points.

mean *Pearson distances* between *decision priorities* of the two individuals x, y. The *Pearson distance* on the decision point d_i is defined as the $dis(x,y)_i = 1 - \rho(x,y)_i$, where $\rho(x,y)_i$ represents the *Pearson correlation* which can be calculated by Eq. (1).

$$\rho(x,y)_i = \frac{\sum_{j=1}^{n}(X_{ij} - \mu_{X_i})(Y_{ij} - \mu_{Y_i})}{\sqrt{\sum_{j=1}^{n}(X_{ij} - \mu_{X_i})^2}\sqrt{\sum_{j=1}^{n}(Y_{ij} - \mu_{Y_i})^2}} \quad (1)$$

where, n is the number of candidate machines/operations. $X_i = [X_{i1}, \cdots, X_{in}]$ and $Y_i = [Y_{i1}, \cdots, Y_{in}]$ represent the *decision priority* on the decision point d_i (each row in Fig. 2) by two individuals, respectively. μ_{X_i} and μ_{Y_i} are the average value of X_i and Y_i.

In summary, the process of calculating the behaviour similarity between individuals is as Algorithm 1. Firstly, for each individual and each decision point, we calculate the priority of each candidate, to form the *decision priority* (line 2). Secondly, for each decision point, calculating the *Pearson correlation* between the *decision priority* of the two individuals (line 3). At the same time, we can calculate the *Pearson distance* on each decision point (line 4). Finally, the behaviour similarity is calculated as the mean of the *Pearson distance* over all the decision points (line 6).

The *Pearson correlation* has been used in DFJSS to measure the importance of each subtree for an individual and has shown success to improve the effectiveness of the scheduling heuristics by swapping the unimportant subtree with the important subtree to do crossover [27]. Therefore, we use the mean *Pearson distances* to estimate the similarity between individuals. Based on the Eq. (1), the *Pearson correlation* $\rho(x,y)$ is always in the range of $[-1, 1]$, so the mean *Pearson distance* is always between 0 and 2. To be specific, if two individuals have the same *decision priority* on all the decision points, the mean *Pearson distances* between them is 0, on the other hand, if they have very different *decision priority* on all the decision points, the mean *Pearson distance* between them is near 2.

C. Kmeans Cluster

This paper uses the Kmeans method to divide the population into different groups. Different from the traditional Kmeans method which uses the Euclidean distance to estimate the similarity between two lists of values, we use the mean *Pearson distance* to estimate how similar two individuals behave with each other, which is shown in Section III-B. The center for each group is selected based on fitness. That is, the *decision priorities* of the individual with the best fitness in a group is set as the center.

For the Kmeans method, the group size needs to be set in advance which might lead to different groups containing very non-uniform numbers of individuals, for example, 100 individuals in one group and only 5 individuals in another group. In this case, if the group with only 5 individuals is selected to select a parent, then the individual with good fitness will be selected many times which might lose the diversity of selected parents. Therefore, we design an adapt cluster number strategy to avoid this situation. That is after the Kmeans method has divided the population into different groups, we check the number of individuals in each group, if a group has less than m individuals, the individuals in this group will be moved to the other group with the most similar behaviour by calculating the mean *Pearson distance* between each individual with the center individual in the other group. Finally, the groups without individuals will be removed.

Algorithm 1: Behaviour similarity estimation

Input: The two individuals: x and y; 40 DFJSS decision points: $D = \{d_1, \ldots, d_{40}\}$.
Output: The behaviour similarity between the two individuals: $sim(x,y)$.

1 **for** $i = 1 \rightarrow 40$ **do**
2 Apply the two individual x, y on the decision point d_i to get the *decision priority* X_i and Y_i, respectively;
3 Calculate the *Pearson correlation* $\rho(x,y)_i$ between the *decision priorities* X_i and Y_i of two individuals;
4 Calculate the *Pearson distance* $dis(x,y)_i = 1 \quad \rho(x,y)_i$ between the two individuals;
5 **end**
6 $sim(x,y) = \sum_{i=1}^{40} dis(x,y)_i / 40$;
7 **return** $sim(x,y)$;

D. Cluster Selection

The cluster selection is used to select parents only for crossover. Different from the traditional tournament selection, it has two steps, the first step is to select two groups randomly. Then, in each group, the tournament selection is applied to select a parent with the best fitness in the second step. In this way, we can select two parents with promising fitness and perform dissimilar with each other.

IV. Experiment Design

To investigate the effectiveness (objectives on unseen test instances) of the proposed GPCS method in different scenarios, this section lists the parameter setting and performs the simulation model. In addition to the GPHH algorithm, an algorithm with the opposite principle, GPHH with the same cluster selection (GPSCS), is also used as a comparison algorithm. This algorithm selects parents from the same group for crossover, i.e., it selects two parents that perform similarly for crossover.

TABLE I
THE TERMINAL SET.

Notation	Description
NIQ	the number of operations in the queue
WIQ	the remaining processing time in the queue
MWT	the machine's waiting time since it gets ready
PT	the processing time of the operation
NPT	the median processing time for next operation
OWT	the operation's waiting time since it gets ready
WKR	the work remaining
NOR	the number of operations remaining of job
W	the job's weight
TIS	the time in system since it is released
TRANT	the transportation time

TABLE II
THE PARAMETER SETTINGS OF GP.

Parameter	Value
Population size	1024
Maximal generations	51
Method to initialise population	Ramped-half-and-half
Initial minimum/maximum depth	2 / 6
Elitism	10
Maximal depth	8
Parent selection	Tournament selection Cluster selection
Mutation rate	0.15
Crossover rate	0.80
Reproduction rate	0.05
Terminal/non-terminal selection rate	10% / 90%
Group number	5
Minimum number of individuals in a group	20

A. Parameter Setting for Genetic Programming

In our experiments, the terminal set of GP is shown in Table I. Terminals are set to represent features related to jobs (e.g., WKR, NOR, W, and TIS), operations (e.g., PT, NPT, and OWT), machines (e.g., NIQ, WIQ, and MWT), and transportation (e.g., TRANT).

The function set has six elements $\{+, -, \times, \div, max, min\}$. The arithmetic operators take two arguments. The "$\div$" operator is protected and returns 1 if divided by zero. The max and min functions take two arguments and return the maximum and minimum of their arguments, respectively. The other parameters about GP are shown in Table II. Note that the *Group number* and *Minimum number of individuals in a group* are two new parameters. The former parameter is set as 5 because a small value might not distinguish individuals with different behaviours well, while a large value would result in having too few individuals in each group, leading to an increased frequency of some individuals being selected. The latter parameter is set as 20 to avoid there being few individuals in a group.

B. Simulation model

In this paper, the simulation environment is used as the experimental model to study the influencing factors of DFJSS. During the scheduling process, jobs arrive at the shop floor dynamically based on a Poisson process with the rate λ. Each job has a different number of operations that are randomly generated from the range $[2, 10]$. These jobs are needed to be processed by 10 heterogeneous machines. These machines have different processing rates which are randomly generated by a uniform discrete distribution between 10 and 15. The distance between machines and the distance between entry/exit point and each machine is assigned with the range $[35, 500]$. The transport speed is set as 5. In addition, the importance of jobs is different, which is represented by weights. The weights of 20%, 60%, and 20% of jobs are set as one, two, and four, respectively. The due date factor is set to 1.5.

The utilisation level denotes how busy the shop floor is. A higher utilisation level represents a busier shop floor. In this paper, we consider six scenarios based on two utilisation levels (0.85 and 0.95) and three objectives. To obtain stable data, warm-up jobs (the first 1000 jobs) are used to get typical scenarios that occur in long-term simulation. Data are collected for the next 5000 jobs and the simulation stops after the 6000th job is completed. Additionally, we generate a single replication but rotate the random seed during each generation of GP to improve the generalisability of the scheduling heuristics.

TABLE III
THE MEAN (STANDARD DEVIATION) OF TEST PERFORMANCE OF 30 INDEPENDENT RUNS OF THE COMPARED METHODS FOR SIX SCENARIOS.

S*	GPHH	GPCS	GPSCS
1*	1291.40(12.62)	1297.73(19.28)(=)	1292.97(7.29)(=)
2*	1375.33(16.77)	1380.39(19.94)(=)	1377.30(14.44)(=)
3*	524.54(2.93)	524.39(3.14)(=)	524.40(2.75)(=)
4*	566.69(3.23)	568.50(4.52)(+)	568.08(3.95)(=)
5*	**1141.09(4.63)**	1145.77(8.80)(+)	1144.63(6.95)(+)
6*	1230.70(8.91)	1234.57(9.42)(=)	1232.55(9.49)(=)

* S: Scenarios, 1: <Fmax, 0.85>, 2: <Fmax, 0.95>, 3: <Fmean, 0.85>, 4: <Fmean, 0.95>, 5: <WFmean, 0.85>, 6: <WFmean, 0.95>.

V. RESULTS AND DISCUSSIONS

A. Test Performance

Table III shows the mean (standard deviation) results of the test performance of 30 independent runs of GPCS, GPSCS, and GPHH on six scenarios. GPCS and CPSCS are compared with the GPHH by the Wilcoxon rank-sum test with a significance level of 0.05. The "$-/+/=$" indicates that the corresponding results are significantly better than, worse than, or similar to the results of GPHH, respectively. We can see that GPCS gives significantly worse results than GPHH on two scenarios (<Fmean, 0.95>, <WFmean, 0.85>) and GPSCS shows significantly worse results than GPHH on one scenario (<WFmean, 0.85>). Fig. 3 gives the convergence curves on the test performance of the compared methods on the six scenarios. We can see that GPCS and GPSCS tend to converge slower than GPHH on almost all the scenarios. We further analyse the reasons in the following Section why the proposed methods that select behaviour different parents for crossover to generate offspring are not effective.

B. Parents Pearson Distance

Fig. 4 shows the histograms of the mean *parents Pearson distance* on generation 1, 25 and 45 of GPCS and GPSCS

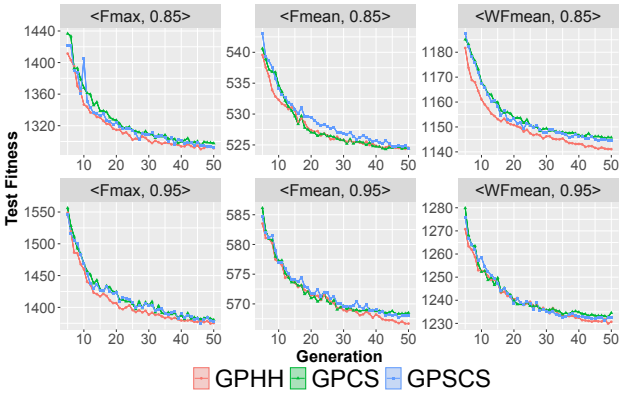

Fig. 3. Convergence curves of test objective values in six scenarios.

Fig. 4. The histograms of the mean *parents Pearson distance* on generation 1, 25 and 45 of GPCS and GPSCS on six scenarios.

on six scenarios. It can be seen that GPSCS gives a smaller mean *Pearson distance* of the selected parents for crossover than GPCS which is expected because GPSCS selects parents from the same group. From Fig. 4, we can see that, in the early stage of evolution, GPCS and GPSCS both obtain large Pearson distance of the selected parents. In the middle and late stages of evolution, the *Pearson distance* of the selected parents by the two methods decreases. It is noted that GPSCS selects parents with about half of the *Pearson distance* than GPCS in the middle and late stages of evolution. This verifies the effectiveness of the proposed cluster selection method that can place individuals with similar behaviours into the same group and select parents with different behaviours.

C. Groups

Fig. 5 gives the points of the group versus fitness of individuals on generations 1, 25, and 45 of GPCS on two scenarios. It is noted that the *bad individuals*, that is the individuals that cannot complete the scheduling (fitness $\rightarrow \infty$) are not shown in the figure. It can be seen that, at generations 1, 25 and 45 of the two scenarios (<Fmax, 0.85>, <Fmean, 0.85>), the individuals are divided into 5 groups. In each group, there are individuals with good fitness and poor fitness. That is, not all individuals with similar behaviours have similar fitness. Table IV shows the number of individuals in each

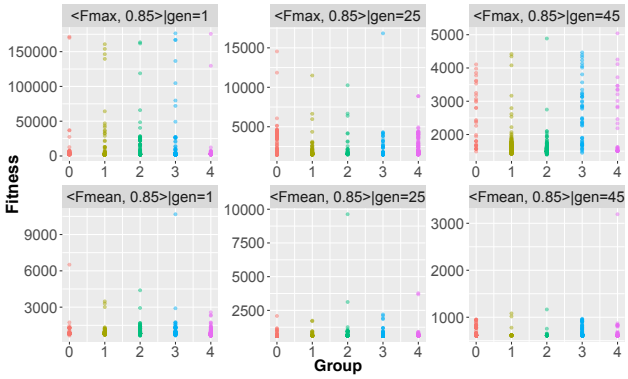

Fig. 5. The points of the group versus fitness of individuals on generations 1, 25 and 45 of GPCS on two scenarios.

TABLE IV
THE NUMBER OF INDIVIDUALS IN EACH GROUP EXCEPT THE *bad individuals* AT GENERATIONS 1, 25 AND 45 OF THE TWO SCENARIOS.

Scenarios-Gen	0	1	2	3	4
<Fmax, 0.85>-gen=1	100	255	255	197	132
<Fmax, 0.85>-gen=25	277	139	52	91	410
<Fmax, 0.85>-gen=45	**34**	**536**	308	52	34
<Fmean, 0.85>-gen=1	84	160	256	168	271
<Fmean, 0.85>-gen=25	148	45	174	325	275
<Fmean, 0.85>-gen=45	381	30	33	430	126

group except the *bad individuals* at generations 1, 25 and 45 of the two scenarios (<Fmax, 0.85>, <Fmean, 0.85>). The distribution of the number of individuals in groups is not uniform. Sometimes, one group may contain 10 times more individuals than the other (e.g., $536 > 34 \times 10$). This non-uniform distribution may cause one individual to be selected many times as a parent when the group it belongs to has a small number of individuals. In this case, the proposed method might lose the population diversity. This might be one of the reasons why the proposed method cannot evolve better scheduling heuristics than the baseline method.

VI. Further Analyses

A. Parents Selection

Parents selection denotes the number of unique individuals selected as parents for crossover, mutation, and reproduction among the population. Fig. 6 shows the convergence curves on *the number of unique parents* selected by GPCS, GPSCS and GPHH on six scenarios. We can see that the proposed two methods have fewer unique parents than GPHH on all the scenarios. At the beginning of the breeding process, the number of unique parents of the proposed methods is big (about 330), as the breeding process continues, the number of unique parents decreases (to about 310) first, then increases and convergences (at about 320) gradually. GPCS and GPSCS give a similar number of unique parents at all generations. The difference on the number of unique parents between the proposed methods and GPHH is about 15 in all the scenarios. This may be one of the reasons why the proposed methods get significantly worse performance than GPHH.

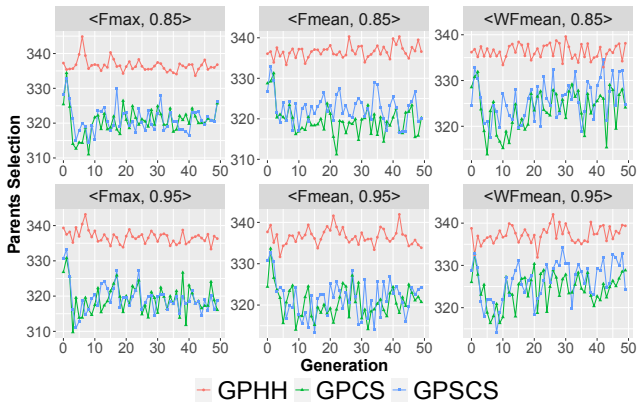

Fig. 6. The convergence curves on the number of *unique parents* selected by the compared methods on six scenarios.

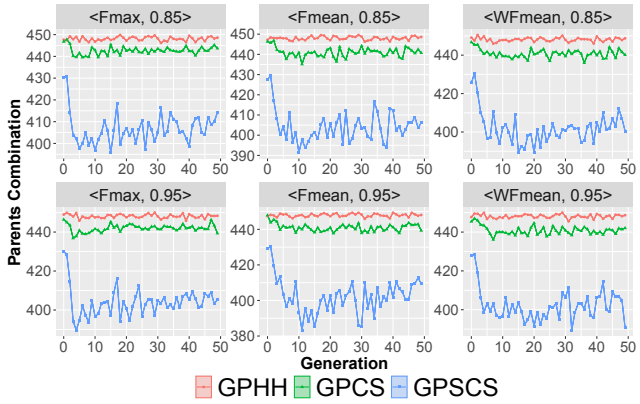

Fig. 7. The curves on the number of *parent combinations* of GPCS and baseline methods on six scenarios.

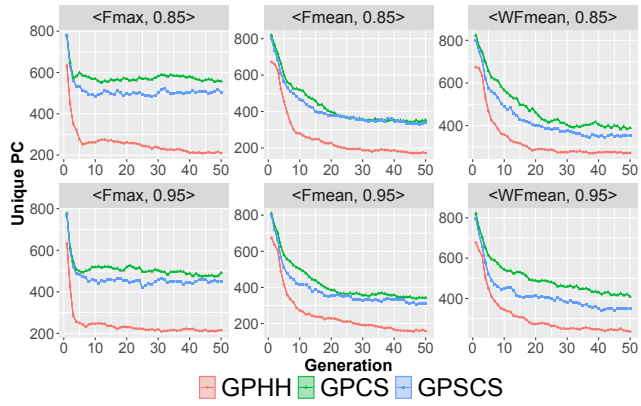

Fig. 8. The convergence curves on the number of *unique PC* of each generation by the compared methods on six scenarios.

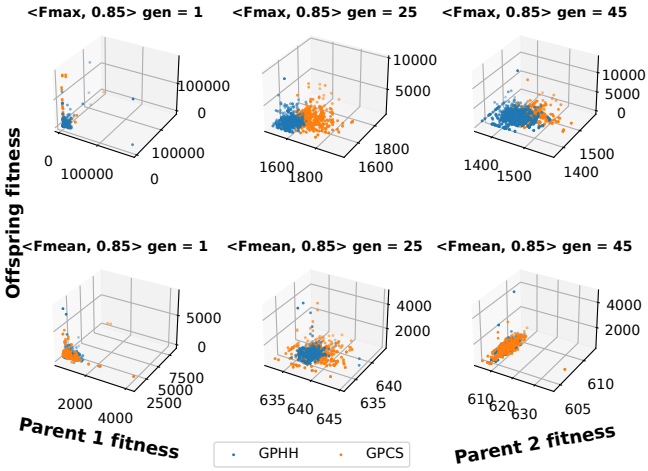

Fig. 9. The parent fitness versus offspring fitness of GPCS, GPSCS and GPHH on two scenarios of a single run on generation 1, 25 and 45.

B. Parent Combination

A *parent combination* means a unique pair of parents for crossover on each generation. Fig. 7 gives the curves on the number of unique parent pairs selected by the compared methods. We can see that GPHH has about 450 unique parent pairs. GPCS selects slightly fewer unique parent pairs (around 440) than GPHH on all the scenarios and GPSCS gives even fewer unique parent pairs (about 400). Because GPSCS limits the candidate individuals to a group to select two parents, which is fewer than the number of individuals in the population. In the beginning, the number of unique parent pairs selected by the proposed methods is large. As the breeding process goes on, the number of unique parent pairs of the proposed methods decreases and reaches a stability value. Combined with the analysis of *parents selection*, we can see that the proposed method not only reduces the number of unique parents but also reduce the number of unique parent pairs. This may be one of the reasons why the proposed methods get significantly worse performance than GPHH on some tested scenarios.

C. Unique PC

Fig. 8 shows the convergence curves on the number of *unique PC* of each generation by the compared methods on six scenarios. In this paper, the *unique PC* means the unique number of the list of decisions made by individuals in the population on 20 sequencing decision points and 20 routing decision points. From Fig. 8, we can see that GPCS and GPSCS both increase the number of *unique PC* on almost all the generations. Compared with GPSCS, GPCS gives a higher *unique PC*. There is a strange phenomenon that GPSCS increases the number of *unique PC*, because the parents selected by GPSCS for crossover come from the same group in which individuals have similar behaviours. Thus GPSCS is expected to decrease the number of *unique PC*. Combined with the analysis of *parents Pearson distance*, we can see that two parents with similar behaviours still can generate offspring with more diverse behaviours. This may be because crossover will affect the behaviours a lot no matter the two parents have similar or different behaviours. This may be one of the reasons why the proposed methods could not get good results.

D. The Selected Parents

To further analyse the performance of the selected parents and their offspring, we select two scenarios as examples. Fig. 9 plots the parent fitness versus offspring fitness of GPCS

and GPHH on two scenarios of a single run on generations 1, 25, and 45. For the selected two scenarios (<Fmax, 0.85>, <Fmean, 0.85>), at the beginning of the breeding process, the fitness of the selected parents and their offspring by the proposed methods are similar to that by the GPHH method. As the breeding proceeds, the fitness of the parents selected by the proposed methods has a more wide range than GPHH at the intermediate and late stages. For the fitness of the generated offspring, sometimes GPCS gives a larger value, while sometimes GPCS gives a smaller value. Taking the analysis of the *unique PC* into consideration, we can see that although the proposed GPCS can select parents with more diverse PC than GPHH, at the same time the selected parents will have poor fitness. This may be one of the reasons why the proposed methods cannot get good performance.

VII. Conclusions

In this paper, a cluster selection scheme is proposed to select two parents with different behaviours for crossover. This method is used to study whether considering different behaviours in parent selection will improve the effectiveness of the evolved scheduling heuristics for DFJSS. The results show that the proposed method increases the number of unique behaviours in the population but does not help to generate effective scheduling heuristics. The analyses of the selected parents, the *parent combination*, the mean *Pearson distance* of the selected parents, and the *unique PC* in the population show that beyond considering fitness, considering behaviour as a further basis for selection will give individuals with poor fitness more probability to be selected which might be the possible reason why the proposed method cannot get effective results. Although the proposed method does not obtain better results than baseline methods, this paper gives an interesting phenomenon that crossover between parents with dissimilar behaviours does increase the number of unique behaviours in the population. In the future, we will propose a new strategy to select parents with different behaviours and good fitness to do crossover to generate offspring which is expected to help GPHH evolve effective scheduling heuristics.

References

[1] A. Rajabinasab and S. Mansour, "Dynamic flexible job shop scheduling with alternative process plans: an agent-based approach," *The International Journal of Advanced Manufacturing Technology*, vol. 54, no. 9, pp. 1091–1107, 2011.

[2] 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*, 2021.

[3] 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, 2020.

[4] K. E. Stecke, "Formulation and solution of nonlinear integer production planning problems for flexible manufacturing systems," *Management Science*, vol. 29, no. 3, pp. 273–288, 1983.

[5] M. Zhou, H. S. Chiu, and H. H. Xiong, "Petri net scheduling of fms using branch and bound method," in *Proceedings of the Conference on IEEE Industrial Electronics*, vol. 1, pp. 211–216, 1995.

[6] J. F. Gonçalves, J. J. de Magalhães Mendes, and M. G. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *European Journal of Operational Research*, vol. 167, no. 1, pp. 77–95, 2005.

[7] M. Saidi-Mehrabad and P. Fattahi, "Flexible job shop scheduling with tabu search algorithms," *International Journal of Advanced Manufacturing Technology*, vol. 32, no. 5, pp. 563–570, 2007.

[8] A. Teymourifar, G. Ozturk, Z. K. Ozturk, and O. Bahadir, "Extracting new dispatching rules for multi-objective dynamic flexible job shop scheduling with limited buffer spaces," *Cognitive Computation*, vol. 12, no. 1, pp. 195–205, 2020.

[9] J. R. Koza *et al.*, *Genetic programming II*, vol. 17. MIT press Cambridge, MA, 1994.

[10] K. Miyashita, "Job-shop scheduling with genetic programming," in *Proceedings of the Conference on Genetic and Evolutionary Computation*, pp. 505–512, 2000.

[11] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Genetic programming for job shop scheduling," in *Evolutionary and Swarm Intelligence Algorithms*, pp. 143–167, Springer, 2019.

[12] 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.

[13] T. Helmuth and A. Abdelhady, "Benchmarking parent selection for program synthesis by genetic programming," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pp. 237–238, 2020.

[14] T. Blickle, "Tournament selection," *Evolutionary Computation*, vol. 1, pp. 181–186, 2000.

[15] H. Xie and M. Zhang, "Parent selection pressure auto-tuning for tournament selection in genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 1–19, 2012.

[16] H. Xie and M. Zhang, "Impacts of sampling strategies in tournament selection for genetic programming," *Soft Computing*, vol. 16, no. 4, pp. 615–633, 2012.

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

[18] J. Park, Y. Mei, G. Chen, and M. Zhang, "Niching genetic programming based hyper-heuristic approach to dynamic job shop scheduling: an investigation into distance metrics," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 109–110, 2016.

[19] V. V. De Melo, D. V. Vargas, and W. Banzhaf, "Batch tournament selection for genetic programming: the quality of lexicase, the speed of tournament," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 994–1002, 2019.

[20] S. Aenugu and L. Spector, "Lexicase selection in learning classifier systems," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 356–364, 2019.

[21] W. La Cava, L. Spector, and K. Danai, "Epsilon-lexicase selection for regression," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 741–748, 2016.

[22] 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*, pp. 472–484, 2018.

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

[24] 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*, pp. 1366–1373, 2019.

[25] Y. Zhou and J. Yang, "Automatic design of scheduling policies for dynamic flexible job shop scheduling by multi-objective genetic programming based hyper-heuristic," *Procedia CIRP*, vol. 79, pp. 439–444, 2019.

[26] B. Metevier, A. K. Saini, and L. Spector, "Lexicase selection beyond genetic programming," in *Genetic Programming Theory and Practice XVI*, pp. 123–136, Springer, 2019.

[27] 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.