Genetic Programming with Multi-case Fitness for Dynamic Flexible Job Shop Scheduling

Meng Xu, Fangfang Zhang✉, Yi Mei, Mengjie Zhang
School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington 6140, New Zealand
{meng.xu, fangfang.zhang, yi.mei, 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/CEC55065.2022.9870340

***Abstract*—Dynamic flexible job shop scheduling has attracted widespread interest from scholars and industries due to its practical value. Genetic programming hyper-heuristic has achieved great success in automatically evolving effective scheduling heuristics to make real-time decisions (i.e., operation ordering and machine assignment) for dynamic flexible job shop scheduling. The design of the training set and fitness evaluation play key roles in improving the generalisation of the evolved scheduling heuristics. The commonly used strategies for improving the generalisation of learned scheduling heuristics include using multiple instances for evaluation at each generation or using a single instance but changing the instance at each new generation of the training process of genetic programming. However, using multiple instances is time-consuming, while changing a single instance at each new generation, potentially promising individuals that happen to underperform in one particular generation might be lost. To address this issue, this paper develops a genetic programming method with a multi-case fitness evaluation strategy, which is named GPMF to evolve the scheduling heuristics with better generalisation ability for the dynamic flexible job shop scheduling problem. The proposed multi-case fitness evaluation strategy divides one instance into multiple cases and uses the average value of the multi-case objectives as the fitness. Experimental results show that the proposed GPMF algorithm is significantly better than the baseline method in all the tested scenarios.**

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

I. Introduction

Dynamic flexible job shop scheduling (DFJSS) [1] is an important task in the manufacturing system, where jobs are assigned to machines at specific times to optimise some objectives. In DFJSS, dynamic events and the flexibility of machines are two of the important challenges which make the problem complex and difficult. The dynamic arrival of new jobs means the information of a job is known once it arrives at the shop floor, which is a common dynamic event in reality. The flexibility of machines means that each operation can be processed by a set of candidate machines. In addition, since different machines have different processing rates, it is necessary to consider the heterogeneous machines in the job shop. Also, after an operation has been completed, if it needs to be processed by another machine, this operation should be transported from the current machine to the selected machine. Therefore, transportation time is also an important factor to be considered in practice [2]. In this paper, we will focus on the DFJSS problem with heterogeneous machines and transportation time.

Since the DFJSS problem has the above characteristics, the earlier methods proposed for the scheduling problem, such as branch-and-bound [3], tabu search [4]–[6] and ant colony algorithms [7], are not suitable for solving this problem, as they cannot make decisions quickly in dynamic environments. In this case, the scheduling heuristic is proposed to make real-time decisions under dynamic environments. In the DFJSS problem, the scheduling heuristic consists of two rules, a sequencing rule, and a routing rule. The sequencing rule is used to prioritise the operations when a machine becomes idle (the sequencing decision point). Then the operation with the highest priority will be selected. The routing rule is used to prioritise the machines when an operation is ready (the routing decision point). Then the machine with the highest priority will be selected. The scheduling heuristic enables fast decisions to be made based on the current system state when a decision point arrives, resulting in greater efficiency and better handling of dynamic events. However, the design of the scheduling heuristic requires extensive experimental studies and is time-consuming.

Genetic programming hyper-heuristic (GPHH) [8] has achieved great success in evolving effective scheduling heuristics to solve different kinds of JSS problems. During the GPHH process, a training set is used to evaluate the scheduling heuristics. The training set plays a key role in improving the quality of the learned scheduling heuristics. If the training set is too large, the training will be very time-consuming. On the other hand, if the training set is too small, then the learned scheduling heuristics might be overfitting to the small training set and cannot generalise well to the unseen test instances. To improve the generalisation of the evolved scheduling heuristics, a straightforward strategy is to increase the number of training instances. However, this strategy increases the training time, especially when the problem is large-scale. A smarter strategy uses only one instance for each generation but changes the instance (rotates the random seed of the DFJSS simulation) at each generation. This strategy is much more computationally efficient than the former strategy but has the limitation of losing some promising solutions at certain generations [9].

To address the above limitation, the overall goal of this

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

paper is to propose a novel multi-case fitness (MF) evaluation strategy for GPHH to automatically evolve the scheduling heuristic to solve the DFJSS problem. The proposed MF strategy divides one DFJSS instance into multiple cases and uses the average of the objectives of all cases as the fitness. To be specific, this paper has the following research objectives:

- Develop a new GPHH that uses the new MF strategy (named GPMF) to calculate the fitness of each individual.
- Conduct sensitivity analysis to investigate the effect of the number of cases (and the fidelity of each case) to the proposed MF strategy.
- Verify the effectiveness of the proposed GPMF in terms of the quality and size of the evolved rules.

II. Background

A. Dynamic Flexible Job Shop Scheduling

On the shop floor, there are a set of heterogeneous machines $\mathcal{M} = \{M_1, ..., M_p, ..., M_m\}$ which are located at different location $\mathcal{L} = \{L_1, ..., L_p, ..., L_m\}$ and own different processing rates. The processing rate of the machine M_p can be represented as $\gamma(M_p)$. In the DFJSS problem, the jobs $\mathcal{J} = \{J_1, J_2, ..., J_n\}$ arrive at the shop floor over time. Each job J_i has a sequence of operations $\mathcal{O}_i = \{O_{i1}, O_{i2}, ..., O_{ik}\}$. Each operation O_{ij} has a workload l_{ij}, and can only be processed by one of its optional machine $\pi(O_{ij}) \subseteq \mathcal{M}$. The processing time $\sigma(O_{ij})$ of the operation relies on the machine $M_p \in \pi(O_{ij})$ that processes it and its workload l_{ij}. Specifically, the processing time can be calculated by $\sigma(O_{ij}) = l_{ij}/\gamma(M_p)$. Once an operation becomes ready, a machine is determined to process this operation, then the operation should be transported to the location of the selected machine M_q from the location of the current machine M_p. The transportation time $T_{p,q}$ depends on the distance $dis_{p,q}$ between the location L_p of the current machine and the location of the selected machine L_q and the speed of the transport robot v. To be specific, the transportation time can be calculated based on $T_{p,q} = dis_{p,q}/v$.

In this paper, we consider three objectives which are max-flowtime, max-tardiness, and max-weighted-tardiness. The definition of the three considered objectives is as follows.

- Minimise $Fmax = \max\{C_1 - r_1, ..., C_n - r_n\}$
- Minimise $Tmax = \max\left\{\begin{matrix}\max\{C_1 - d_1, 0\}\\ \vdots \\ \max\{C_n - d_n, 0\}\end{matrix}\right\}$
- Minimise $WTmax = \max\left\{\begin{matrix}\max\{w_1(C_1 - d_1), 0\}\\ \vdots \\ \max\{w_n(C_n - d_n), 0\}\end{matrix}\right\}$

where n represents the number of jobs, r_i, C_i, d_i and w_i represent the release time, completion time, due date and weight of the job J_i, respectively.

In addition, the following constraints are considered in this paper.

- An operation can be processed only after all the previous operations have been completed.
- Scheduling is non-preemptive, that is, the processing of an operation cannot be stopped or suspended once it is started until it is completed.
- Each operation can only be processed by one of its candidate machines.
- Each machine can process up to one operation at a time.
- The attribute (i.e. processing rate and location) of each machine will not change throughout the whole work and the machine will not break down.

B. Genetic Programming Hyper-heuristic

GPHH involves two main processes, which are a training process and a test process. During the training process, a training set is used to evolve the scheduling heuristics. Then the scheduling heuristic output from the training process is applied to an unseen test set during the test process to assess the out-of-sample performance of the trained scheduling heuristic and the effectiveness of the GPHH.

The training process of GPHH consists of four main parts, namely population initialisation, fitness evaluation, selection, and breeding (crossover, mutation, and reproduction). In the beginning, a group of individuals is initialised as a population, each individual denotes a solution with a designed representation. Then the fitness of each individual is evaluated on the training set. After fitness evaluation, the selection process determines individuals as parents for crossover, mutation, and reproduction to generate offspring to inherit to the next generation. The above process is cyclical until a stopping condition is reached. The best individual obtained will be output as the final solution.

C. Related Work

An important type of common method for scheduling is the exact methods, such as mathematical programming [10] and branch-and-bound [3], which can obtain the optimal solution. However, exact methods are only suitable for solving static and small-scale JSS problems, but not suitable for solving dynamic or large-scale JSS problems because of their high computational cost and inability to handle dynamic events efficiently. Some heuristic methods, such as genetic algorithm [11]–[13] and tabu search [4], [5], can solve large-scale JSS problems, but still are not suitable for solving DJSS problems because of their limitation of being unable to handle dynamic events efficiently. In contrast, scheduling heuristics are much more suitable for solving DFJSS. Scheduling heuristics have many advantages over the other existing approaches. Firstly, they can react to dynamic events such as new job arrivals in real-time. Secondly, the implementation of scheduling heuristics is simple. Finally, they can be reused based on the latest information of the environment. However, there are still some limitations of scheduling heuristics, that is, the design of scheduling heuristics needs much domain knowledge and it is time-consuming to design an effective scheduling heuristic. GPHH has been widely used to solve different kinds of JSS problems and has received great success in evolving effective scheduling heuristics. GPHH has several advantages that make

it stand out from many optimisation algorithms. Firstly, GPHH has a flexible representation that allows it to explore different structures. Secondly, it has good interpretability as the user can see and analyse the learned rules that have been evolved. Finally, the evolved rule by GPHH has good generalisation capability, which makes it possible to reuse the rules in different situations. However, its generalisation capability is strongly related to the design of the training set and has a great impact on its performance on the test set.

DFJSS has attracted widespread interest from scholars and industries due to its practical applied value. To obtain effective scheduling heuristics, some existing methods improve the generalisation capability of evolved scheduling heuristics by using a set of instances to do evaluation for each generation [14]–[20] or using a single instance but generating new training instance by rotating the random seed of the simulation to evaluate individuals in each new generation [21]–[27]. However, these two types of methods both have their drawbacks. Using a set of instances to do evaluation for each generation is time-consuming because of the increasing time for evaluation. Using a single instance but generating a new training instance to evaluate individuals in each new generation might eliminate individuals that perform well in subsequent generations [9]. Additionally, surrogate models are also used to approximate the fitness. For example, a *HalfShop* surrogate model is used to evaluate the fitness of each individual by using a simplified simulation model of the original shop [28]. The surrogate model is mainly used to reduce the training time while not sacrificing the performance of evolved rules. To improve the generalisation capability of the evolved scheduling heuristics, taking advantage of individuals already generated in the process of breeding while not increasing the training time, an MF evaluation strategy is proposed in this paper to help GP evolve effective scheduling heuristics.

III. The New Approach

A. The Overall Framework

The overall framework of the proposed GPMF is shown in Fig. 1. The main difference from the existing approaches (e.g., the multi-tree baseline GP [29]) is that GPMF uses the proposed novel MF evaluation to calculate the fitness of each individual which will affect the elitism selection and parent selection, thus influencing the evolution process.

As with traditional GPHH, population initialisation, fitness evaluation, elitism selection, parent selection, reproduction, crossover, and mutation are all necessary components. The proposed novel MF evaluation strategy is used to divide one instance into multiple cases and get a new fitness to replace the original one by calculating the mean value of objectives of multiple cases. Details about the proposed MF evaluation strategy and how to use it will be described in the following subsection.

B. The Multi-case Fitness Evaluation

The newly proposed MF evaluation strategy calculates the fitness of the individual in a new way and the fitness will be used for elitism selection and parent selection.

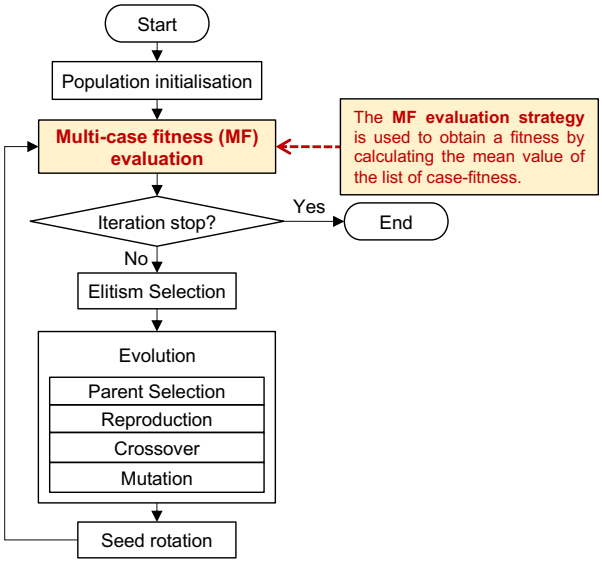

Fig. 1. The flowchart of the proposed GPMF method.

Traditional GPHHs evaluate an individual (scheduling heuristic) by applying the individual to a DFJSS simulation to generate a schedule. Then the fitness of the individual is set to the objective value of the generated schedule (e.g. max flowtime or mean flowtime), which denotes an aggregation (max or mean) of the jobs done during the simulation. Different from the traditional evaluation in GPHHs, in the design of the multi-case fitness, a critical issue is the definition of *cases*. Consider that in regression and classification, fitness (e.g. mean squared error) is an aggregation of the objectives of all the cases in the training set. Similarly, in DFJSS, we can consider each part of the jobs as a case. That is, we can divide the jobs completed during the DFJSS simulation into groups and treat each group as a case to calculate the objective.

Due to the above considerations, the MF evaluation for DFJSS is proposed, as shown in Algorithm 1. The inputs of this algorithm include an individual (scheduling heuristic) x, a DFJSS simulation sim, the number of jobs n, and the group size g. Firstly, depending on the number of jobs n and the size of the group g, we can calculate the number of cases $c = n/g$ (line 1, assuming that n is always divisible by g). Secondly, the MF evaluation strategy runs the simulation with the individual x to get the schedule $sch(sim, x)$. Then, the completion time of each job can be obtained from the generated schedule. Thirdly, the MF evaluation strategy calculates the case-fitness, which is the objective (line 7) of each case. Finally, we can calculate the fitness of the individual x on simulation sim (line 10) by calculating the mean of all the objectives for each case.

In summary, the difference between the proposed MF evaluation and the existing fitness evaluation functions are as follows:

- Compared with the existing multi-instance fitness, the MF evaluation is much faster, as it only runs one instance.

Algorithm 1: Multi-case fitness evaluation for DFJSS.

Input: The individual to be evaluated: x; DFJSS simulation: sim; number of jobs in the simulation: n; group size: g.
Output: Fitness $fit(x)$.

```
Calculate the number of cases c = n/g;
Run the simulation sim with the scheduling heuristic x to obtain
  the corresponding schedule sch(sim, x);
Calculate a list of case-fitness cf(x) and get the mean value of them
  as fitness fit(x);
sum ← 0;
for i = 1 → c do
    Set the ith group of jobs J_i = {g × (i − 1) + 1, ..., g × i};
    Calculate cf_i(x) = obj_{j∈J_i}(C_j);
    sum = sum + cf_i(x);
end
fit(x) = sum/c;
return fit(x);
```

- Compared with the existing single-instance fitness, it is the same if the objective is to calculate the mean objective of the jobs, such as the mean flowtime and mean tardiness since the average of the objectives of the cases is equivalent to the average of all the jobs. However, it will be different if the objective is based on the maximum objective of the jobs, such as max tardiness and max flowtime.
- If there is only one case, it becomes the single instance fitness, and the proposed GPMF is equivalent to the traditional GP. If the number of cases equals the number of jobs in the simulation, it is the other extreme that considers each job independently, and always calculates the mean objective even for the max objectives. So the best balance should be somewhere in between.

IV. Experimental Design

A. Parameter Setting

In this paper, the terminals and functions used to compose the individuals (scheduling heuristics) are shown in Table I. The terminals represent features that are relevant to operations (e.g., NPT, OWT, and PT), jobs (e.g., NOR, W, WKR, and TIS), machines (e.g., WIQ, MWT, and NIQ), and transport (e.g., TRANT). The function "max/min" takes two arguments and returns the maximum/minimum of the arguments. The functions "$+$", "$-$", "$\times$", and "$\div$" also take two arguments, where "$\div$" is protected and gives 1 if it is divided by zero.

Table II gives the parameters related to GP and GPMP. All the parameters are fixed except the number of cases, when the number of cases is 1, the proposed GPMF reduces to GP.

B. Simulation model

In this paper, the simulation model is used to study the influencing factors of GPMF. We assume that there are 10 heterogeneous machines with different processing rates which are randomly generated with the range $[10, 15]$ on the shop floor. The distance between machines and the distance between the entry/exit point and each machine is assigned by a uniform discrete distribution between 35 and 500. The transport robot has a speed of 5. When the scheduling process starts, jobs arrive on the shop floor over time following a Poisson process with a rate of λ. Each job has varying amounts of operations that are randomly generated by a uniform discrete distribution within the range of $[2, 10]$. Different jobs are set to have different importance and these are represented by weights (i.e., 20%, 60%, and 20% of the jobs are set to 1, 2, and 4, respectively). The workload for each operation is assigned according to a uniform discrete distribution over the range $[100, 1000]$. The due date factor for the simulation is 1.5.

The utilisation level is a key parameter representing the different simulation scenarios. The higher the utilisation level, the busier the job shop system will be. This paper considers six scenarios based on two utilisation levels (0.85 and 0.95) and three objectives. In the long-term simulation, warm-up jobs (the first 1000 jobs) are used to obtain a stable job shop scenario and guarantee the accuracy of the collected data. Data is then collected for the next 5,000 jobs. The simulation is stopped at the completion of the 6000th job.

TABLE I
THE TERMINALS AND FUNCTIONS.

Notation	Description
NPT	the next operation's median processing time
PT	the operation's processing time.
OWT	the waiting time of the operation since it gets ready
NOR	the remaining number of operations of the job
WKR	the remaining processing time of the job
W	the weight of the job
TIS	the time in the system since release
WIQ	the remaining processing time in the queue of machine
NIQ	the number of operations in the queue of machine.
MWT	the waiting time of the machine since it gets ready
TRANT	the transportation time between machines/entry/exit
Functions	max, min, $+$, $-$, $\times$, $\div$

TABLE II
THE PARAMETER SETTINGS OF GP.

Parameter	Value
The size of population	1024
The maximal generations	51
Initialisation method	Ramped-half-and-half
Initial minimum/maximum depth	2 / 6
Number of elitism	10
The maximal depth of tree	8
Parent selection	Tournament selection
The reproduction rate	0.05
The crossover rate	0.80
The mutation rate	0.15
Selection rate of terminal/non-terminal	10% / 90%
Cases	1,2,5,10,20,25,40,50,100,200, 250,500,1000,2500,5000

V. Results

A. Parameter Sensitivity Analysis of Multi-Case Strategy

This section analyses the sensitivity of the number of cases to the multi-case strategy. 30 independent runs are performed for each scenario, and 50 instances are used as the test set for measuring the performance of the evolved scheduling heuristics. The Wilcoxon rank sum test at the 0.05 significance level was then used to validate the performance

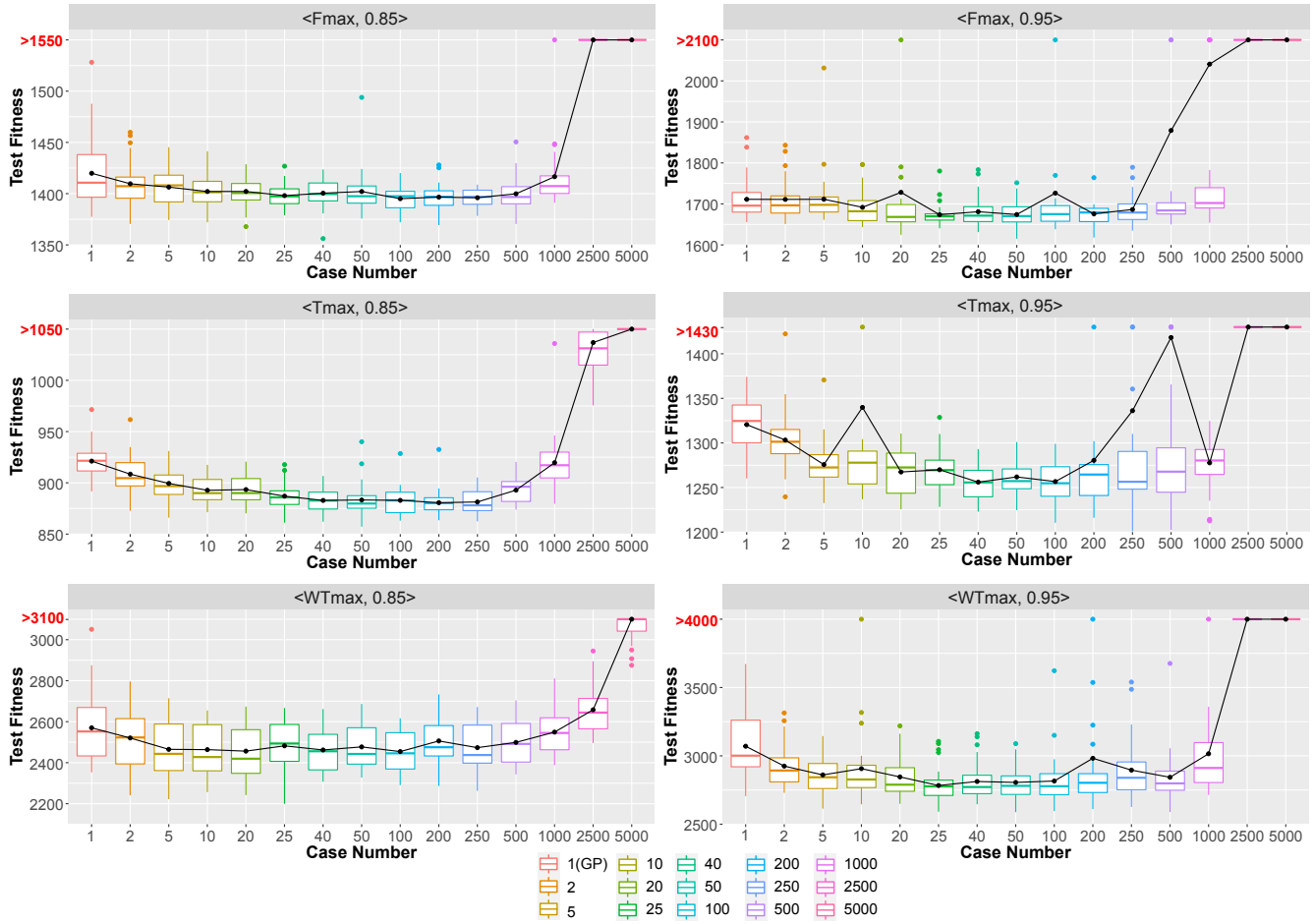

Fig. 2. Box plots of test fitness on one scenario with different numbers of cases.

of the proposed GPMF method. The "−/+" denotes that the corresponding results are significantly better/worse than that of the baseline GP method. The "=" represents that there is no statistical difference between the results of the two methods.

The number of cases of GPMF is set as 1, 2, 5, 10, 20, 25, 40, 50, 100, 200, 250, 500, 1000, 2500 and 5000 (the largest number of jobs). When the number of cases is 1, the GPMF is reduced to baseline GP. Fig. 2 shows the box plot of test fitness on six scenarios of GPMF with different numbers of cases, and the baseline GP, where the black curves represent the mean test fitness of 30 runs as the number of cases increases. Additionally, Table III gives the mean (standard deviation) test fitness of 30 independent runs of baseline GP, and GPMF with different numbers of cases for all the six scenarios. It is apparent from the tables and Fig. 2 that the proposed GPMF with 10, 20, 25, 40, 50, 100 and 250 cases perform significantly better than baseline GP on all the scenarios, GPMF with 200 cases performs significantly better than baseline GP on five scenarios (<Fmax, 0.85>, <Tmax, 0.85>, <Fmax, 0.95>, <Tmax, 0.95> and <WTmax, 0.95>), GPMF with 5 cases performs significantly better than baseline GP on four scenarios (<Tmax, 0.85>, <WTmax, 0.85>, <Tmax, 0.95> and <WTmax, 0.95>), GPMF with 500 cases performs significantly better than baseline GP on four scenarios (<Fmax, 0.85>, <Tmax, 0.85>, <Tmax, 0.95> and <WTmax, 0.95>) too, but unlike GPMF with 5 cases, GPMF with 2 cases performs significantly better than baseline GP on three scenarios (<Tmax, 0.85>, <Tmax, 0.95> and <WTmax, 0.95>), GPMF with 1000 cases performs significantly better than baseline GP on two scenarios (<Tmax, 0.95> and <WTmax, 0.95>), while GPMF with 2500 and 5000 cases perform significantly worse than baseline GP on all scenarios.

Based on the above results, we can see that the number of cases plays a really important role in the proposed method. When the number of cases is less than 1000, the proposed method can obtain good results on most of the scenarios. When the number of cases is too large, the results become very poor, especially when the number of cases is equal to the number of jobs. Additionally, for different scenarios, the number of cases still plays a key role. For example, for scenario <Tmax, 0.85>, the proposed method with 200 cases obtains the best results among all the settings, while for scenario <WTmax, 0.95>, the best number of cases is 50.

Table. IV shows the number of significantly better results between GPMFs with different numbers of cases that outper-

TABLE III
THE MEAN (STANDARD DEVIATION) TEST FITNESS OF 30 INDEPENDENT RUNS OF BASELINE GP, AND GPMF WITH DIFFERENT NUMBERS OF CASES FOR SIX SCENARIOS.

Cases	<Fmax, 0.85>	<Fmax, 0.95>	<Tmax, 0.85>	<Tmax, 0.95>	<WTmax, 0.85>	<WTmax, 0.95>	-/=/+
1(GP)	1419.89(35.04)	1711.16(49.52)	921.17(16.61)	1320.53(26.35)	2570.14(169.36)	3070.49(218.33)	-
2	1409.62(22.14)(=)	1710.99(48.29)(=)	**908.47(19.33)(-)**	**1303.24(35.85)(-)**	2520.58(152.56)(=)	**2925.12(151.49)(-)**	3/3/0
5	1406.37(17.26)(=)	1711.30(68.02)(=)	**899.40(15.28)(-)**	**1275.69(26.87)(-)**	**2465.13(134.00)(-)**	**2860.57(151.00)(-)**	4/2/0
10	**1402.08(16.65)(-)**	**1691.66(41.03)(-)**	**892.71(11.95)(-)**	**1339.86(369.90)(-)**	**2463.75(123.36)(-)**	**2906.41(325.46)(-)**	**6/0/0**
20	**1402.13(14.70)(-)**	**1728.16(278.46)(-)**	**893.40(14.13)(-)**	**1267.40(26.82)(-)**	**2456.40(122.44)(-)**	**2845.91(160.42)(-)**	**6/0/0**
25	**1398.08(12.39)(-)**	**1673.64(26.55)(-)**	**886.98(12.76)(-)**	**1269.91(23.37)(-)**	**2482.25(118.16)(-)**	**2783.25(138.10)(-)**	**6/0/0**
40	**1400.46(13.71)(-)**	**1680.92(38.82)(-)**	**882.84(10.53)(-)**	**1255.97(19.61)(-)**	**2462.13(114.41)(-)**	**2812.48(135.27)(-)**	**6/0/0**
50	**1402.09(21.03)(-)**	**1674.11(31.68)(-)**	**883.27(15.65)(-)**	**1261.84(20.54)(-)**	**2477.04(106.89)(-)**	**2805.70(128.13)(-)**	**6/0/0**
100	**1395.09(11.28)(-)**	**1726.30(266.69)(-)**	**882.90(13.55)(-)**	**1256.66(21.82)(-)**	**2454.22(99.67)(-)**	**2815.71(193.63)(-)**	**6/0/0**
200	**1396.61(12.80)(-)**	**1675.60(26.27)(-)**	**880.62(12.77)(-)**	**1280.45(112.44)(-)**	2506.06(111.83)(=)	**2982.34(826.88)(-)**	5/1/0
250	**1396.00(8.62)(-)**	**1686.43(35.09)(-)**	**881.31(12.47)(-)**	**1336.14(317.27)(-)**	**2473.37(115.29)(-)**	**2895.63(211.79)(-)**	**6/0/0**
500	**1399.86(17.04)(-)**	1879.05(756.64)(=)	**892.85(12.33)(-)**	**1418.29(504.18)(-)**	2499.05(105.62)(=)	**2843.24(195.49)(-)**	4/2/0
1000	1416.52(36.37)(=)	2040.93(809.57)(=)	919.61(28.85)(=)	**1277.64(27.22)(-)**	2549.20(97.32)(=)	**3015.48(448.54)(-)**	2/4/0
2500	1627.10(37.21)(+)	4628.60(709.25)(+)	1036.85(37.38)(+)	4209.50(743.46)(+)	2658.14(112.27)(+)	6412.27(1062.19)(+)	0/0/6
5000	2206.17(49.93)(+)	4920.53(499.40)(+)	1588.27(59.02)(+)	4446.50(645.05)(+)	3115.47(112.69)(+)	6716.63(934.35)(+)	0/0/6

TABLE IV
THE NUMBER OF SIGNIFICANTLY BETTER RESULTS BETWEEN GPMFS WITH DIFFERENT NUMBERS OF CASES FOR SIX SCENARIOS.

Cases	10(-)	20(-)	25(-)	40(-)	50(-)	100(-)	250(-)
10	-	0	3	2	2	**3**	1
20	0	-	1	2	1	**2**	2
25	0	0	-	1	1	**1**	0
40	0	0	0	-	0	**1**	1
50	0	0	0	0	-	**0**	0
100	0	0	0	0	0	-	0
250	0	0	1	1	1	**1**	-
Sum	28	32	39	39	38	**45**	36

form GP on all the tested scenarios based on the pairwise Wilcoxon rank-sum test results. Based on the results, we can see that GPMF with 100 cases wins in all algorithms with 45 significantly better results (i.e., 3 significantly better results than GPMF with 10 cases $+\cdots+$ 1 significantly better result than GPMF with 250 cases). Therefore, the effectiveness of the proposed GPMF with 100 cases is selected to do further analysis by comparing it with the baseline GP on six scenarios.

B. Overall Results

Based on the above sensitivity analysis, the effectiveness of the proposed GPMF with 100 cases is selected to do further analysis by comparing it with the baseline GP on six scenarios. The convergence curves are shown in Fig. 3. Based on Table III, we can see that the GPMF with 100 cases performs significantly better than baseline GP on all six scenarios. Based on Fig. 3, we can see that the method proposed in this paper has a faster convergence speed and better convergence results. Additionally, based on Fig. 3, for half of the scenarios (<Tmax, 0.85>, <Fmax, 0.85>, and <WTmax, 0.85>), the scheduling heuristics evolved by GPMF give smoother convergence curves on the test set, while the curves obtained by baseline GP are relatively more fluctuating which indicates that the scheduling heuristics evolved by GPMF have better generalisation ability. Overall, these results indicate that the proposed GPMF could obtain significantly better results than the baseline GP method.

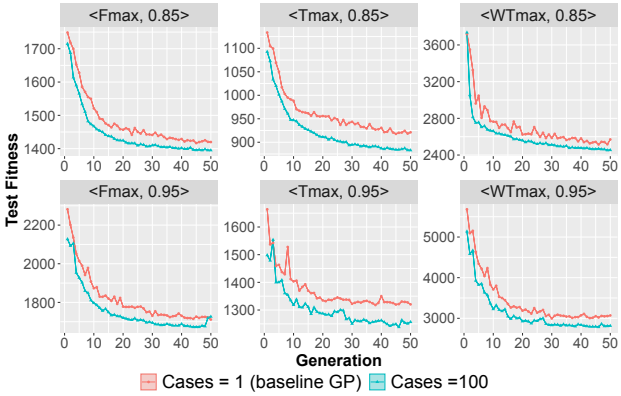

Fig. 3. Convergence curves of the test fitness on six scenarios.

VI. FURTHER ANALYSIS

A. Diversity Analysis

In this section, we investigate whether the proposed method influences the diversity of the individuals under the phenotypic characterisation (PC) [30]. The PC is a vector of numerical values, which represents the behaviour of each individual. In this paper, we use a fixed instance to obtain 20 sequencing decision points and 20 routing decision points [9], and if two scheduling heuristics have the same decisions for all the 40 decision points, we consider that they have the same PC. For each generation, the percentage of unique PC among the population is used to represent the PC diversity. It is noted that the same PC of two scheduling heuristics does not definitely mean they have the same decisions on all the decisions during the whole simulation because for a simulation, there are thousands of decision points. However, it indicates that the two rules could make similar decisions on most decision points. Fig. 4 shows the convergence curves of PC diversity of three algorithms on six scenarios. We can see that as the evolution process continues, the PC diversity decreases with the proposed GPMF methods. Overall, we can see that the proposed method can reduce the PC diversity, which might be one of the reasons it gets good performance.

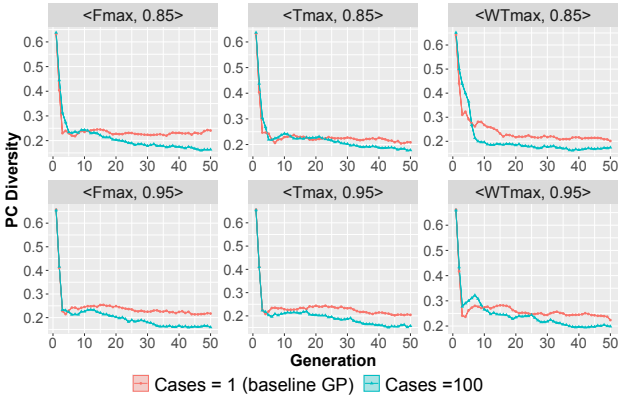

Fig. 4. Convergence curves of PC diversity on six scenarios.

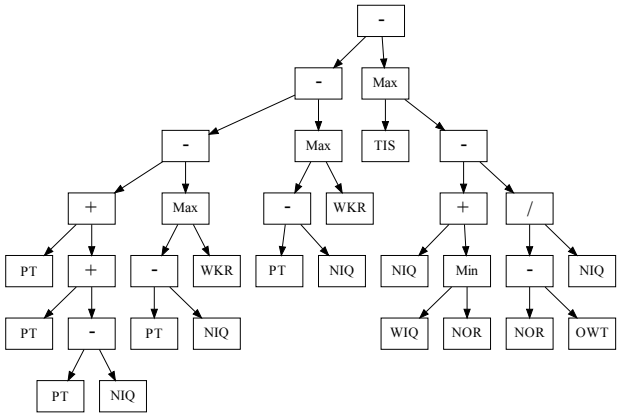

Fig. 5. An example of sequencing rule evolved by the proposed method.

TABLE V
THE MEAN (STANDARD DEVIATION) OF ROUTING AND SEQUENCING RULE SIZE OF GP AND GPMF WITH 100 CASES FOR SIX SCENARIOS.

S*	sequencing rule		routing rule	
	GP	GPMF	GP	GPMF
1*	45.93(17.55)	49.27(19.81)(=)	**36.73(13.39)**	46.33(13.82)(+)
2*	**40.20(15.52)**	52.20(18.36)(+)	**45.53(21.37)**	51.47(21.22)(+)
3*	45.93(14.49)	51.73(22.99)(=)	**37.20(18.97)**	47.47(20.02)(+)
4*	46.67(16.30)	47.53(14.09)(=)	**40.67(16.98)**	51.47(17.47)(+)
5*	41.27(15.92)	44.40(21.11)(=)	**33.60(16.22)**	44.20(12.84)(+)
6*	48.47(22.69)	42.87(9.61)(=)	**42.13(19.17)**	51.07(17.25)(+)

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

B. Rule Size

Table V shows the mean (standard deviation) results of the sequencing and routing rule size of 30 independent runs. As the population evolves, the size of both the sequencing rule and routing rule tends to increase. Additionally, we can see that for all scenarios, the routing rule size of the best rule evolved by GPMF is larger than baseline GP, and on these scenarios, GPMF can get significantly better performance than baseline GP. For the sequencing rule size, we can see similar phenomena for one of the scenarios (<Fmax, 0.95>), that is, the sequencing rule size of the best rule evolved by GPMF is significantly larger than baseline GP. Based on the above analysis, we can see that larger sequencing rule and routing rule sizes, especially larger routing rule sizes tend to lead to better performance, which might be one of the reasons GPMF obtains good performance.

C. Structure Analysis of Evolved Scheduling Heuristics

To gain further understanding of the behaviour of the scheduling heuristic evolved by the proposed method, an evolved scheduling heuristic is selected to be analysed. Figs. 5 and 6 show the sequencing rule and routing rule from the selected scheduling heuristic evolved by GPMF on scenario <Fmax, 0.85>. The selected scheduling heuristic has a promising test performance.

From Figs. 5 and 6, it is observed that the sequencing rule is a combination of seven terminals (NIQ, PT, NOR, TIS, WKR, OWT, and WIQ), where NIQ and PT are the most frequently used terminals in this rule. The sequencing rule can be simplified as S below.

$$S = 3PT - NIQ - 2\max\{PT - NIQ, WKR\} - \max\{TIS, NIQ + \min\{WIQ, NOR\} - \frac{NOR - OWT}{NIQ}\}$$

Additionally, the routing rule is a combination of nine terminals (TRANT, OWT, WIQ, WKR, PT, NPT, TIS, MWT, and NIQ), where TRANT and OWT are the most frequently used terminals in this rule. The routing rule can be simplified as R below.

$$R = 2OWT + 2PT + 2TRANT + WIQ - \frac{NPT}{TIS} - \frac{(MWT + NIQ)NPT}{OWT \times WKR + TRANT} - \frac{WIQ}{WKR}$$

The terminal TRANT is not used by the sequencing rule, while it is used three times by the routing rule. The terminal TRANT is mainly designed for the transportation time and the routing rule uses it to select a machine to process the operation. When one machine is selected, the operation should be transported to the selected machine, so TRANT should be considered and this result is what we expect. Additionally, the subtree $\max\{PT - NIQ, WKR\}$ is used twice in the sequencing rule and the subtree $OWT + PT$ is used twice in the routing rule, which might mean that these two subtrees can play a key role in the selection of operation and machine, respectively.

VII. CONCLUSIONS

This paper sets out to improve the effectiveness of the evolved scheduling heuristics by improving the generalisation ability of the evolved scheduling heuristics. This goal has been successfully achieved by proposing a new GPMF method. The main novelty is the MF strategy that uses the multi-case fitness to replace the original fitness.

The experimental results and analysis have shown that the proposed method performs significantly better than the baseline GP on all the scenarios. The evidence from the experimental results suggests that the proposed MF strategy can improve the generalisation capability and effectiveness of

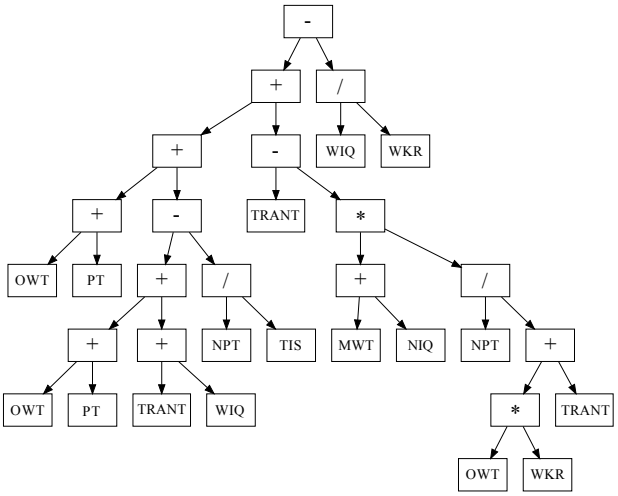

Fig. 6. An example of routing rule evolved by the proposed method.

the evolved scheduling heuristics. The present findings have important implications on the fitness evaluation process for solving DFJSS or other combinatorial optimisation problems with evolutionary algorithms. This new fitness is significantly better than the baseline GP on the maximum objectives while being equivalent to the baseline GP for the mean objectives. Future work will explore this MF strategy for solving other combinatorial optimisation problems.

REFERENCES

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

[2] G. Zhang, J. Sun, X. Liu, G. Wang, and Y. Yang, "Solving flexible job shop scheduling problems with transportation time based on improved genetic algorithm," *Mathematical Biosciences and Engineering*, vol. 16, no. 3, pp. 1334–1347, 2019.

[3] 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, IEEE, 1995.

[4] 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.

[5] G. Vilcot and J.-C. Billaut, "A tabu search algorithm for solving a multicriteria flexible job shop scheduling problem," *International Journal of Production Research*, vol. 49, no. 23, pp. 6963–6980, 2011.

[6] P. Fattahi, M. Messi Bidgoli, and P. Samouei, "An improved tabu search algorithm for job shop scheduling problem trough hybrid solution representations," *Journal of Quality Engineering and Production Optimization*, vol. 3, no. 1, pp. 13–26, 2018.

[7] K.-L. Huang and C.-J. Liao, "Ant colony optimization combined with taboo search for the job shop scheduling problem," *Computers & operations research*, vol. 35, no. 4, pp. 1030–1046, 2008.

[8] J. Park, S. Nguyen, M. Zhang, and M. Johnston, "A single population genetic programming based ensemble learning approach to job shop scheduling," in *Proceedings of the Conference on Genetic and Evolutionary Computation*, pp. 1451–1452, 2015.

[9] M. Xu, F. Zhang, Y. Mei, and M. Zhang, "Genetic programming with archive for dynamic flexible job shop scheduling," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 2117–2124, IEEE, 2021.

[10] 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.

[11] 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.

[12] D. C. Mattfeld and C. Bierwirth, "An efficient genetic algorithm for job shop scheduling with tardiness objectives," *European Journal of Operational Research*, vol. 155, no. 3, pp. 616–630, 2004.

[13] 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.

[14] Y. Zhou, J. Yang, and L. Zheng, "Hyper-heuristic coevolution of machine assignment and job sequencing rules for multi-objective dynamic flexible job shop scheduling," *IEEE Access*, vol. 7, pp. 68–88, 2018.

[15] 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.

[16] M. Gholami and M. Zandieh, "Integrating simulation and genetic algorithm to schedule a dynamic flexible job shop," *Journal of Intelligent Manufacturing*, vol. 20, no. 4, pp. 481–498, 2009.

[17] K. Li, Q. Deng, L. Zhang, Q. Fan, G. Gong, and S. Ding, "An effective mcts-based algorithm for minimizing makespan in dynamic flexible job shop scheduling problem," *Computers & Industrial Engineering*, vol. 155, p. 107211, 2021.

[18] 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.

[19] F. Geyik and A. T. Dosdoğru, "Process plan and part routing optimization in a dynamic flexible job shop scheduling environment: an optimization via simulation approach," *Neural Computing and Applications*, vol. 23, no. 6, pp. 1631–1641, 2013.

[20] C. Dimopoulos and A. M. Zalzala, "Investigating the use of genetic programming for a classic one-machine scheduling problem," *Advances in Engineering Software*, vol. 32, no. 6, pp. 489–498, 2001.

[21] Y. Mei and M. Zhang, "A comprehensive analysis on reusability of gp-evolved job shop dispatching rules," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 3590–3597, IEEE, 2016.

[22] 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*, pp. 33–49, Springer, 2019.

[23] S. Nguyen, Y. Mei, B. Xue, and M. Zhang, "A hybrid genetic programming algorithm for automated design of dispatching rules," *Evolutionary Computation*, vol. 27, no. 3, pp. 467–496, 2019.

[24] Y. Mei, S. Nguyen, and M. Zhang, "Constrained dimensionally aware genetic programming for evolving interpretable dispatching rules in dynamic job shop scheduling," in *Proceedings of the Asia-Pacific Conference on Simulated Evolution and Learning*, pp. 435–447, Springer, 2017.

[25] 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.

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

[27] 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.

[28] 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, 2016.

[29] 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.

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