Genetic Programming with Diverse Partner 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
Wellington, New Zealand
{meng.xu,yi.mei,fangfang.zhang,mengjie.zhang}@ecs.vuw.ac.nz

ABSTRACT

Dynamic flexible job shop scheduling (DFJSS) aims to make decisions for machine assignment and operation sequencing simultaneously to get an effective schedule under dynamic environments. Genetic programming hyper-heuristic (GPHH) has been successfully applied to evolve scheduling heuristics for the DFJSS problem. Parent selection plays an important role in GPHH for generating high-quality offspring. Traditional GPHHs select parents for crossover purely based on fitness (e.g., tournament selection). This might be too greedy to get good offspring and the selected parents might have similar structures/behaviours. In this paper, a GPHH method with a new *diverse partner selection* (DPS) scheme is proposed, namely GPDPS, for DFJSS. Specifically, we first define a new multi-case fitness to characterise the behaviour of each scheduling heuristic for DFJSS. Then, the newly proposed DPS method selects a pair of *complementary* high-quality parents for crossover to generate offspring. The experimental results show that GPDPS significantly outperforms the GPHH method on most of the DFJSS scenarios, in terms of both test performance and convergence speed.

CCS CONCEPTS

• **Computing methodologies** → **Planning under uncertainty**; **Heuristic function construction**.

KEYWORDS

dynamic flexible job shop scheduling, genetic programming, hyper-heuristic, diverse partner selection

ACM Reference Format:
Meng Xu, Yi Mei, Fangfang Zhang✉, Mengjie Zhang. 2022. Genetic Programming with Diverse Partner Selection for Dynamic Flexible Job Shop Scheduling. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3528920

1 INTRODUCTION

Dynamic flexible job shop scheduling (DFJSS) [13] aims to optimise the machine assignment and operation sequencing under dynamic environments, such as new job dynamic arrivals. Each job has a number of operations that need to be processed by a set of machines. DFJSS is a challenging optimisation problem. There are two kinds of decisions in DFJSS. One is the routing decision (allocate machine to a ready operation), and the other is the sequencing decision (select the operation for an idle machine to process next).

Scheduling heuristics are widely used to generate effective schedules in real-time [8]. In DFJSS, a scheduling heuristic has a routing rule and a sequencing rule. At each decision point, the routing rule/sequencing rule prioritises each candidate machine/operation, then the highest priority machine/operation is selected. However, manually designed scheduling heuristics highly rely on experts, and the designing process is time-consuming. Genetic programming hyper-heuristic (GPHH) [5] has been successfully applied for solving job shop scheduling problems automatically [1, 6, 7, 9, 12]

GPHH generates new individuals following a process of fitness evaluation, parent selection, crossover, mutation, and reproduction until the termination criteria is met, then retaining the best individual obtained as the final output. During the process, parent selection plays an important role in identifying promising individuals that carry good genes [4]. Tournament selection [3] is the classical parent selection method that selects parents based on their fitness. Tournament selection selects the parents independently, without considering the relationship between the selected parents. As a result, it may select high-quality parents with very similar structures/behaviours. Crossover is an important GP operator. To generate good offspring, the parents for crossover should not only have high fitness by themselves but also complement with each other. To consider the complementation of parents, diverse partner selection (DPS) is proposed to find more high-quality combinations of parent pairs and has shown success in solving some optimisation problems [2]. However, DPS cannot be directly applied to evolving scheduling heuristics for DFJSS.

In this paper, we aim to propose a novel GPHH method with a new DPS scheme (GPDPS) to select a pair of *complementary* high-quality parents for crossover to solve the DFJSS problem effectively.

2 PROPOSED APPROACH

2.1 The Overall Framework

The proposed GPDPS method takes the following steps to generate the scheduling heuristics. At first, a population of individuals is initialised by the ramped-half-and-half method. Then, at each generation, individuals are evaluated by the newly developed multi-case fitness evaluation. The multi-case fitness evaluation calculates both the standard fitness and a list of case-fitness. After the multi-case fitness evaluation, the case-fitness values in the list are normalised by the fitness normalisation step. Then, a number of elites with top standard fitness are selected to the next generation directly. Afterward, the breeding process is conducted to generate offspring

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3528920

through parent selection and genetic operators. Specifically, the parents for mutation and reproduction are selected by the traditional tournament selection, while the parents for crossover are selected by the newly developed DPS operator based on the normalised list of case-fitness. Note that the crossover and mutation rates are adjusted based on the number of times that the suitable second parent fails to be selected by DPS. The operator rate tuning strategy is the same as the original method [2]. Specifically, each time DPS fails to select suitable parents, the crossover (mutation) rate is decreased (increased) by $1/popsize$. After the entire breeding process, the crossover and mutation rates are reset to the default values. The above steps are repeated until a stopping criterion is met, and the best individual is returned.

2.2 The Multi-case Fitness Evaluation

The newly proposed multi-case fitness evaluation strategy has two purposes. First, it calculates the standard fitness of the individual, which will be used for elitism selection and select parents for mutation and reproduction. Second, it calculates a list of case-fitness values to characterise the behaviour of the individual in different cases. Such a list will be used to select parents for crossover.

The calculation of the standard fitness is straightforward. On the other hand, the key issue of designing the list of case-fitness is the definition of *cases*. In DFJSS, an individual (scheduling heuristic) is evaluated by being applied to a DFJSS simulation to generate a schedule. Its fitness is typically set as the objective value (e.g., mean flowtime or makespan) of the generated schedule, which is a kind of aggregation (mean or max) over the jobs completed during the simulation. Considering that in classification and regression, the fitness (e.g., mean square error) is an aggregation over the training data points, and each data point is treated as a case. Analogously, we can consider the part of each job in the objective function as a case for DFJSS.

We design the multi-case fitness evaluation for DFJSS in Algorithm 1. Given an individual x, a DFJSS simulation sim, the number of jobs in the simulation n and the group size g, the multi-case fitness evaluation strategy first applies the individual x to the simulation to generate the corresponding schedule $sch(sim, x)$. Then, it is easy to obtain the completion time C_j for each job in the generated schedule. On the other hand, based on the number of jobs n and group size g (assuming that n is always divisible by g), we can obtain the number of cases $c = n/g$ and the jobs in each group (index from $g \times (i-1)+1$ to $g \times i$). Finally, we can calculate the standard fitness of x (line 6) and the fitness for each case (line 9).

2.3 Case-fitness Normalisation

To avoid biasing to any case during DPS, for each case, we normalise the case-fitness of all individuals in the population. Specifically, for each case $i = 1, \ldots, c$, the normalisation is done by

$$\mathbf{cf}_i(x) \leftarrow \frac{\mathbf{cf}_i(x) - \min\{\mathbf{cf}_i(x') | x' \in pop\}}{\max\{\mathbf{cf}_i(x') | x' \in pop\} - \min\{\mathbf{cf}_i(x') | x' \in pop\}}. \quad (1)$$

2.4 The New Diverse Partner Selection Method

The proposed new DPS first selects the first parent, i.e., the *recipient*, by tournament selection. Then, it keeps selecting the second parent by tournament selection and examines if the second parent has sufficient positive influence on the recipient. To this end, we define α as the actual influence of the second parent to the recipient, and β as the expected positive influence. If $\alpha > \beta$, then the second parent is confirmed as the *donor*, and the selection is terminated. Otherwise, the selection is continued. In the end, the operator rates are tuned accordingly.

Compared with the existing DPS, the main difference is on the calculation of the α and β parameters. Specifically, the new α parameter is defined as follows.

$$\alpha(x_1, x_2) = \sum_{i=1}^{c} \frac{\mathbf{cf}_i(x_1) - \mathbf{cf}_i(x_2)}{\max(\mathbf{cf}_i(x_1), \mathbf{cf}_i(x_2))}. \quad (2)$$

It calculates the total normalised advantage of the case-fitness $\mathbf{cf}_i(x_2)$ over $\mathbf{cf}_i(x_1)$ (the smaller the better, since the objective is minimised) over all the cases. The new β parameter is set to 0, which means that the crossover is allowed if the second parent has a positive influence on the recipient.

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: Standard fitness $fit(x)$; List of case-fitness $\mathbf{cf}(x)$.
1 Calculate the number of cases $c = n/g$;
2 Run the simulation sim with the scheduling heuristic x to obtain the corresponding schedule $sch(sim, x)$;
3 **for** $j = 1 \rightarrow n$ **do**
4 Obtain the job completion time C_j from $sch(sim, x)$;
5 **end**
6 Calculate $fit(x) = \texttt{obj}_{j=1,\ldots,n}(C_j)$;
7 **for** $i = 1 \rightarrow c$ **do**
8 Set the ith group of jobs $\mathcal{J}_i = \{g \times (i-1)+1, \ldots, g \times i\}$;
9 Calculate $\mathbf{cf}_i(x) = \texttt{obj}_{j \in \mathcal{J}_i}(C_j)$;
10 **end**
11 **return** $fit(x)$, $\mathbf{cf}(x)$;

3 EXPERIMENT DESIGN

In the experiments, we test our algorithms on eight DFJSS scenarios, which are characterised by (1) the objective to be optimised and (2) utilisation level (how busy the shop floor is). We consider the four objectives (max flowtime, mean flowtime, max tardiness, and mean weighted tardiness) and two utilisation levels of 0.85 (less busy) and 0.95 (busier). Each scenario is denoted as <obj, ul>, where obj represents the objective and ul denotes the utilisation level. For each scenario, we train the scheduling heuristics by using a set of simulations. Then, the trained scheduling heuristics are tested on an unseen set of simulations.

Each simulation contains 5000 jobs that need to be processed by 10 heterogeneous machines, whose processing rates are randomly generated within the range $[10, 15]$. The distances between machines and between the entry/exit point and each machine are sampled from a uniform discrete distribution between 35 and 500. The transport speed is set as 5. New jobs arrive over time according to a Poisson process. The number of operations of each job is randomly generated by a uniform discrete distribution between 2 and 10. The weights of 20%, 60%, and 20% of jobs are set as one, two, and four. The workload of each operation is assigned by uniform discrete distribution with the range $[100, 1000]$. The due date factor

Table 1: The mean (standard deviation) of test performance of 30 independent runs of the compared methods for eight scenarios.

Scenarios	GPHH	GPDPS*	GPDPS
<Fmax, 0.85>	1291.40(12.62)	**1284.45(12.46)(-)**	**1284.86(19.02)(-)(=)**
<Fmax, 0.95>	1375.33(16.77)	**1368.16(23.17)(-)**	**1366.72(13.97)(-)(=)**
<Fmean, 0.85>	524.54(2.93)	525.54(4.86)(=)	524.27(4.52)(=)(=)
<Fmean, 0.95>	566.69(3.23)	566.83(3.20)(=)	567.18(3.73)(=)(=)
<Tmax, 0.85>	733.27(14.69)	**719.52(11.28)(-)**	728.25(14.86)(=)(+)
<Tmax, 0.95>	860.42(19.09)	**847.18(18.69)(-)**	**841.51(19.82)(-)(=)**
<WFmean, 0.85>	1141.09(4.63)	1141.83(4.58)(=)	1146.52(12.33)(=)(=)
<WFmean, 0.95>	1230.70(8.91)	1232.42(11.57)(=)	1230.99(8.72)(=)(=)

is set to 1.5. The simulation uses 1000 jobs to warm up the shop floor, and collect the information of the next 5000 jobs [10].

In our experiments, the function set is as $\{+, -, \times, \div, max, min\}$. The terminal set includes the features related to the transportation time (TRANT), machines, operations, and jobs which can be seen in [10]. For GP, the population size is set to 1024, the maximal depth of the evolved rule is set to 8. The crossover, mutation, and reproduction rates are 0.8, 0.15, and 0.05. For DPS, each group is designed with 160 jobs, to find a pair of parents, the number of times used to find a donor in DPS is set to 20. The process stops after 51 generations. For each DFJSS instance, the proposed method and baseline methods are run 30 times independently.

4 EXPERIMENTAL RESULTS

Two algorithms are taken into comparison in this paper. The first is the traditional genetic programming (GPHH) [11]. The second is a variant of GPDPS that excludes the operator rate tuning, named GPDPS*. The comparison between GPDPS (GPDPS*) and GPHH verifies the effectiveness of the DPS crossover parent selection. The comparison between GPDPS and GPDPS* shows the influence of the operator rate tuning strategy.

4.1 Test Performance

Table 1 gives the mean (standard deviation) results of the test performance of 30 independent runs of GPDPS and baseline methods for eight scenarios. We compare each algorithm with the algorithms on its left 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 with the results of the algorithms on its left. From the table, it can be seen that GPDPS* can obtain significantly better performance than GPHH on four scenarios (<Fmax, 0.85>, <Fmax, 0.95>, <Tmax, 0.85>, <Tmax, 0.95>), and GPDPS can get significantly better performance than GPHH on three scenarios (<Fmax, 0.85>, <Fmax, 0.95>, <Tmax, 0.95>). Fig. 1 shows the convergence curves on the test performance of the compared methods on the eight scenarios. We can see that GPDPS and GPDPS* tend to converge faster than GPHH on almost all the scenarios. This verifies the effectiveness of the new DPS method that selects a pair of complementary parents to generate offspring. It is noted that GPDPS and GPDPS* can obtain significantly better performance than GPHH on most scenarios with maximum objectives while showing similar performance with GPHH on all the scenarios with mean objectives. This may be because mean objectives are not easy to be optimised due to their stability.

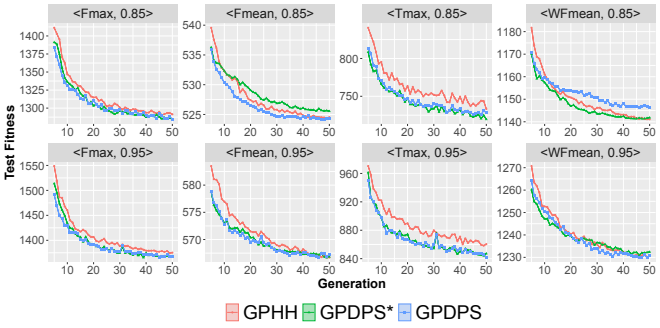

Figure 1: The convergence curves on test fitness of GPDPS and baseline methods on eight scenarios.

Figure 2: The curves on the crossover rate of the end of each generation of GPDPS on eight scenarios.

4.2 The Effect of Operator Rate Tuning

Fig. 2 gives the curves on the crossover rate at the end of each generation of GPDPS on the eight scenarios. It can be seen that the crossover rate is reduced during the breeding process at each generation but would not be reduced lower than about 0.7. Based on the test results in Table 1, we can see that GPDPS performs significantly worse than GPDPS* on one scenario (<Tmax, 0.85>), and on the other scenarios, GPDPS performs similarly with GPDPS*. The operator rate tuning is used in the traditional DPS in [2], but no specific analysis is done to show the effect of the operator rate tuning strategy. However, based on the above results, in DFJSS the operator rate tuning strategy has no positive effect on the test performance. It also shows that the decrease of the crossover rate is less than 0.08, which means that less than 10% (80 out of 800, each time decreased by 0.001) of the crossover fail to find suitable parents.

5 FURTHER ANALYSIS

5.1 Parents Selection

Parents selection means the number of unique individuals selected as parents for crossover, mutation, and reproduction among the population. Fig. 3 shows the convergence curves on *the number of unique parents* selected by GPDPS and baseline methods on eight scenarios. It can be seen 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

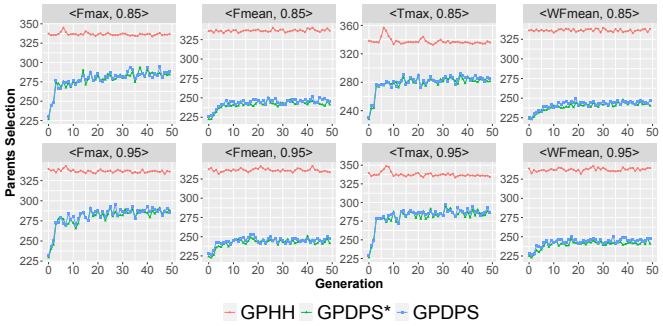

Figure 3: The convergence curves on the number of *unique parents* selected by the compared methods on eight scenarios.

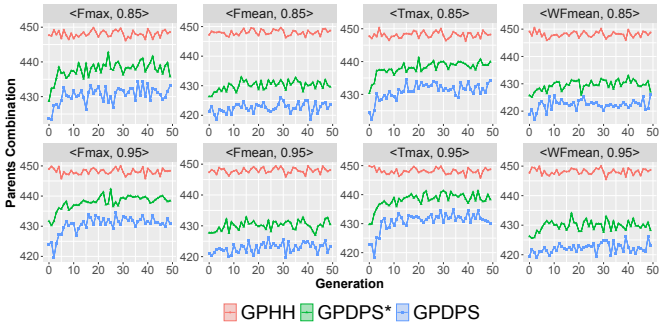

Figure 4: The curves on the number of *parent combinations* of GPDPS and baseline methods on eight scenarios.

methods is small, as the breeding process continues, the number of unique parents increases and convergences gradually, and after about 5 generations, there are no obvious changes, but only small fluctuations. GPDPS and GPDPS* give a similar number of unique parents on all generations. It is noted that for maximum objectives and mean objectives the numerical range of the final convergence value of the number of unique parents is different. The difference on the number of unique parents between the proposed methods and GPHH is about 50 on the maximum objectives, and about 100 on the mean objectives. This may be one of the reasons why the proposed methods can get significantly better performance than GPHH on the maximum objectives, while only similar performance on the mean objectives (lose too much parent diversity).

5.2 Parent Combination

A *parent combination* denotes a unique pair of parents for crossover on each generation. Fig. 4 gives the curves on the number of unique parent pairs selected by the compared methods. It can be seen that GPHH has about 450 unique parent pairs. In the beginning, the number of unique parent pairs selected by the proposed methods is small. As the breeding process goes on, the number of unique parent pairs of the proposed methods increases and reaches a stability value (around 430). Combined with the analysis of *parents selection*, we can see that although the number of unique parents is really small at the beginning of the breeding process, the number of unique parent pairs is not that small. This may be because the proposed method can select a smaller number of individuals as parents than GPHH but gives them many different combinations to generate offspring. This may be one of the reasons why the proposed method can get significantly better performance than GPHH on most tested scenarios.

6 CONCLUSIONS

The goal of this paper is to design a new parent selection method to select a pair of parents with high quality and complement with each other for crossover and help GPHH evolve effective scheduling heuristics for solving the DFJSS problem. This goal has been successfully achieved by the newly proposed diverse partner selection (DPS) method and the resultant GPDPS algorithm. The GPDPS algorithm is examined and compared with the baseline GPHH on eight scenarios. The results show that the GPDPS method can outperform the baseline GPHH on all the maximum objectives and achieve similar performance on all the mean objectives. Additionally, the further analysis gives a general understanding of how the *parents selection* and *parent combinations* change by the newly proposed method. Overall, the newly designed DPS method can contribute to GPHH to generate effective scheduling heuristics.

In the future, new strategies will be proposed to help the GPDPS method to evolve effective scheduling heuristics not only on maximum objectives but also on mean objectives.

REFERENCES

[1] Harith Al-Sahaf, Ying Bi, Qi Chen, Andrew Lensen, Yi Mei, Yanan Sun, Binh Tran, Bing Xue, and Mengjie Zhang. 2019. A survey on evolutionary machine learning. *Journal of the Royal Society of New Zealand* 49, 2 (2019), 205–228.
[2] Muhammad Waqar Aslam, Zhechen Zhu, and Asoke Kumar Nandi. 2018. Diverse partner selection with brood recombination in genetic programming. *Applied Soft Computing* 67 (2018), 558–566.
[3] Yongsheng Fang and Jun Li. 2010. A review of tournament selection in genetic programming. In *International Symposium on Intelligence Computation and Applications*. 181–192.
[4] Thomas Helmuth and Amr Abdelhady. 2020. Benchmarking parent selection for program synthesis by genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 237–238.
[5] John R Koza et al. 1994. *Genetic programming II*. Vol. 17. MIT press Cambridge, MA.
[6] Kazuo Miyashita. 2000. Job-shop scheduling with genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 505–512.
[7] Su Nguyen, Yi Mei, and Mengjie Zhang. 2017. Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems* 3, 1 (2017), 41–66.
[8] Li Nie, Liang Gao, Peigen Li, and Xinyu Li. 2013. A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. *Journal of Intelligent Manufacturing* 24, 4 (2013), 763–774.
[9] Joc Cing Tay and Nhu Binh Ho. 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering* 54, 3 (2008), 453–473.
[10] Meng Xu, Fangfang Zhang, Yi Mei, and Mengjie Zhang. 2021. Genetic Programming with Archive for Dynamic Flexible Job Shop Scheduling. In *Proceedings of the IEEE Congress on Evolutionary Computation*. 2117–2124.
[11] Fangfang Zhang, Yi Mei, and Mengjie Zhang. 2018. Genetic programming with multi-tree representation for dynamic flexible job shop scheduling. In *Proceedings of the Australasian Joint Conference on Artificial Intelligence*. 472–484.
[12] Fangfang Zhang, Su Nguyen, Yi Mei, and Mengjie Zhang. 2021. DOI: 10.1007/978-981-16-4859-5. Genetic Programming for Production Scheduling: An Evolutionary Learning Approach. In *Machine Learning: Foundations, Methodologies, and Applications*. Springer, XXXIII+338.
[13] Yong Zhou, Jianjun Yang, and Zhuang Huang. 2020. Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming. *International Journal of Production Research* 58, 9 (2020), 2561–2580.