Importance-Aware Genetic Programming for Automated Scheduling Heuristics Learning in Dynamic Flexible Job Shop Scheduling

Fangfang Zhang[1(✉)], Yi Mei[1], Su Nguyen[2], and Mengjie Zhang[1]

[1] School of Engineering and Computer Science, Victoria University of Wellington, PO BOX 600, Wellington 6140, New Zealand
{fangfang.zhang,yi.mei,mengjie.zhang}@ecs.vuw.ac.nz
[2] Centre for Data Analytics and Cognition, La Trobe University, Bundoora, Australia
P.Nguyen4@latrobe.edu.au

Abstract. Dynamic flexible job shop scheduling (DFJSS) is a critical and challenging problem in production scheduling such as order picking in the warehouse. Given a set of machines and a number of jobs with a sequence of operations, DFJSS aims to generate schedules for completing jobs to minimise total costs while reacting effectively to dynamic changes. Genetic programming, as a hyper-heuristic approach, has been widely used to learn scheduling heuristics for DFJSS automatically. A scheduling heuristic in DFJSS includes a routing rule for machine assignment and a sequencing rule for operation sequencing. However, existing studies assume that the routing and sequencing are equally important, which may not be true in real-world applications. This paper aims to propose an importance-aware GP algorithm for automated scheduling heuristics learning in DFJSS. Specifically, we first design a rule importance measure based on the fitness improvement achieved by the routing rule and the sequencing rule across generations. Then, we develop an adaptive resource allocation strategy to give more resources for learning the more important rules. The results show that the proposed importance-aware GP algorithm can learn significantly better scheduling heuristics than the compared algorithms. The effectiveness of the proposed algorithm is realised by the proposed strategies for detecting rule importance and allocating resources. Particularly, the routing rules play a more important role than the sequencing rules in the examined DFJSS scenarios.

Keywords: Importance-aware scheduling heuristics learning · Genetic programming · Hyper-heuristic · Dynamic flexible job shop scheduling

1 Introduction

Dynamic flexible job shop scheduling (DFJSS) [1,2] is an important combinatorial optimisation problem which is valuable in real-world applications such as production scheduling in manufacturing and processing industries [3,4].

G. Rudolph et al. (Eds.): PPSN 2022, LNCS 13399, pp. 48–62, 2022.
https://doi.org/10.1007/978-3-031-14721-0_4

The goal of DFJSS is to find effective schedules to process a number of jobs by a set of machines [5]. In DFJSS, each job consists of a number of operations, and each operation can be processed by more than one machine. Two decisions, i.e., machine assignment to allocate operations to machines and operation sequencing to select an operation to be processed next by an idle machine, need to be made simultaneously. In addition, the decision marking has to be made under dynamic environments such as continuously job arrival [6,7].

Genetic programming (GP) [8], as a hyper-heuristic approach [9–12], has been successfully used to learn scheduling heuristics for DFJSS [13,14]. For GP in DFJSS, a scheduling heuristic consists of a routing rule and a sequencing rule which are used to make decisions on machine assignment and operation sequencing, respectively. The quality of schedules depends on the interaction of the routing rule and the sequencing rule. These two rules in previous studies are normally regarded as being equally important, and are given the same amount of computational resources to evolve. However, this is not necessarily the case in real world applications, and giving too many computational resources to less important rules may lead to a waste of resources and negatively affect the quality of schedules.

To this end, this paper aims to propose an effective importance-aware scheduling heuristics learning GP approach for DFJSS. The developed rule importance measure reflects the significance of the routing rule and the sequencing rule in DFJSS. Inspired by the computation resource allocation strategy that is widely used to allocate resources to sub-problems [15–19], an adaptive computational resource allocation strategy is designed to give more resources for learning the more important rules. In this paper, we use the number of individuals to represent the magnitude of the resources. The proposed algorithm aims to help GP find better scheduling heuristics by allocating proper number of individuals between learning the routing and sequencing rules in DFJSS. Specifically, this paper has the following research objectives.

- Develop an effective strategy to measure the importance of the routing rule and the sequencing rule in the decision making of DFJSS.
- Propose an adaptive computational resource allocation strategy based on the rule importance.
- Analyse the effectiveness of the proposed algorithm in terms of the performance of learned rules.
- Analyse how the proposed algorithm affects GP's behaviour in terms of the number and the ratio of individuals assigned, and the reward for each rule.
- Analyse the effect of the proposed algorithm on the sizes of the learned scheduling heuristics.

2 Background

2.1 Dynamic Flexible Job Shop Scheduling

In DFJSS, m machines $\mathcal{M} = \{M_1, M_2, ..., M_m\}$ are required to process n jobs $\mathcal{J} = \{J_1, J_2, ..., J_n\}$. Each job J_j has a sequence of operations $\mathcal{O}_j = \{O_{j1}, O_{j2}, ..., O_{jl_j}\}$ that need to be processed one by one, where l_j is the number of operations of job J_j.

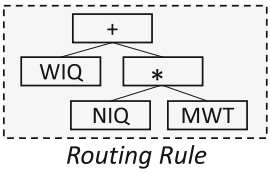

Fig. 1. An example of the routing rule learned by GP.

Each operation O_{ji} can be processed on more than one machine $M(O_{ji}) \subseteq \pi(O_{ji})$ [20]. Thus, the machine that processes an operation determines its processing time $\delta(O_{ji}, M(O_{ji}))$. This paper focuses on one of the most common dynamic events in real life, i.e., new jobs arrive dynamically [21,22]. The information about a new job is unknown until it arrives on the shop floor. Below are the main constraints of the DFJSS problems.

- A machine can only process one operation at a time.
- Each operation can be handled by only one of its candidate machines.
- An operation cannot be handled until its precedents have been processed.
- Once started, the processing of an operation cannot be stopped.

We consider three commonly used objectives in this paper. The calculations of them are shown as follows.

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

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

2.2 GP for DFJSS

GP starts with a randomly initialised population that contains a number of individuals (i.e., *initialisation*). GP programs consist of terminals and functions, which are naturally priority functions for prioritising machines and operations in DFJSS. Figure 1 shows an example of the routing rule which is a priority function of WIQ + NIQ * MWT where WIQ and NIQ are the workload and the number of operations in the queue of a machine, MWT is the needed time of a machine to finish its current processing operation. The quality of individuals is evaluated by measuring the decision marking performance of applying individuals on DFJSS simulations (i.e., *evaluation*). New offspring are generated by genetic operators, i.e., crossover, mutation, and reproduction, with selected parents (i.e., *evolution*). New offspring will be put into and evaluated in the next generation. GP improves the quality of individuals generation by generation until the stopping criterion is met. The best learned scheduling heuristic at the last generation is reported as the output of the GP algorithm [23,24].

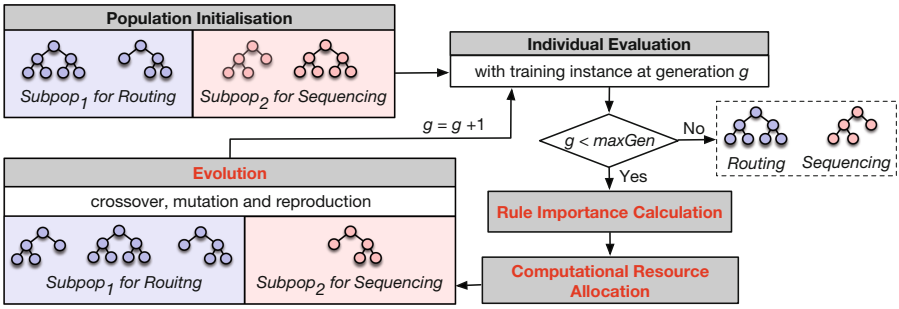

Fig. 2. The flowchart of the proposed algorithm.

3 Importance-Aware Scheduling Heuristic Learning

3.1 An Overview of the Proposed Algorithm

Figure 2 shows the flowchart of the proposed algorithm, and the newly developed components are highlighted in red. We use cooperative coevolution strategy to learn the routing rule and the sequencing rule simultaneously [25,26]. The population consists of two subpopulations, and the first (second) subpopulation $Subpop_1$ ($Subpop_2$) is used to learn the routing (sequencing) rule. The evolutionary processes of the two subpopulations are independent except for the individual evaluation. Since a routing rule and a sequencing rule have to work together to make decisions in DFJSS, for individual evaluation, the individuals in $Subpop_1$ ($Subpop_2$) at the current generation are evaluated with the best individual in $Subpop_2$ ($Subpop_1$) at the previous generation. The best scheduling heuristic obtained from the whole population is reported as the best at the current generation, i.e., can either be from $Subpop_1$ or from $Subpop_2$. Since there is no previous generation for the first generation, for individual evaluation in $Subpop_1$ ($Subpop_2$), we randomly select one individual in $Subpop_2$ ($Subpop_1$).

Before evolution, we first measure the importance of the routing rule and the sequencing rule. Then, we use the rule importance information to allocate computational resources, i.e., individuals, for learning different rules. More computational resources will be allocated to the important rule, which is expected to improve the overall scheduling effectiveness in DFJSS. The number of individuals in $Subpop_1$ and $Subpop_2$ is adaptive. As the example shown in Fig. 2, at the beginning, there are two individuals in each subpopulation for learning each rule. After the computational resources allocation, three individuals are used to learn the routing rule and one individual is utilised for the sequencing rule. The details of the developed new components are shown in the following subsections.

3.2 Measure the Importance of the Routing and Sequencing Rules

This paper measures the importance of rules based on their contributions to the fitness improvement which is calculated according to consecutive generations.

Algorithm 1: Reward Calculation of the Routing and Sequencing Rule

```
1: rewardRouting = 0, rewardSequencing = 0, counter = 3
2: while counter ⩽ g do
3:     if fitness_1 < fitness'_1 and fitness_2 < fitness'_2, or fitness_1 > fitness'_1
       and fitness_2 > fitness'_2 then
4:         if △1 < △2 then
5:             rewardRouting = rewardRouting + 1
6:         end
7:         if △1 > △2 then
8:             rewardSequencing = rewardSequencing + 1
9:         end
10:        if △1 = △2 then
11:            rewardRouting = rewardRouting + 0
12:            rewardSequencing = rewardSequencing + 0
13:        end
14:    else
15:        rewardRouting = rewardRouting + 0
16:        rewardSequencing = rewardSequencing + 0
17:    end
18:    counter = counter + 1
19: end
20: return rewardRouting, rewardSequencing
```

We assume the best fitness of $Subpop_1$ and $Subpop_2$ are $fitness_1$ and $fitness_2$ at the current generation, and $fitness'_1$ and $fitness'_2$ at the previous generation. We calculate the fitness improvement of $Subpop_1$ for learning routing rules and $Subpop_2$ for learning sequencing rules as $\triangle 1 = (fitness_1 - fitness'_1)/fitness'_1$ and $\triangle 2 = (fitness_2 - fitness'_2)/fitness'_2$, respectively. In the general minimisation problems, we can compare $\triangle 1$ and $\triangle 2$ directly (i.e., $\triangle 1 \leqslant 0$ and $\triangle 2 \leqslant 0$), and treat the one with smaller $\triangle$ (i.e., larger $|\triangle|$) as the important one. However, it is not always the case in this paper due to the used instance rotation strategy, i.e., different generations use different training instances, which has been successfully used to train scheduling heuristics with GP [27,28]. This indicates that the fitness scales are different across generations due to the difference of training instances, and we are not sure whether the fitness will increase or decrease across consecutive generations. Thus, this paper defines that the routing rule will be more important than the sequencing rule when $\triangle 1 < \triangle 2$ under the conditions of either $fitness_1 > fitness'_1$ and $fitness_2 > fitness'_2$ or $fitness_1 < fitness'_1$ and $fitness_2 < fitness'_2$ (lines 3–13). The rewards for the routing rule *rewardRouting* and the sequencing rule *rewardSequencing* are calculated as shown in Algorithm 1, where g is the current generation number. It is noted that we do not measure the rule importance at the generations in either of the following two cases, i.e., If $fitness_1 > fitness'_1$ and $fitness_2 < fitness'_2$, or If $fitness_1 < fitness'_1$ and $fitness_2 > fitness'_2$ (lines 14–17), due to the unknown fitness change information. How to measure rule importance in these two cases will be studied in our future work.

3.3 Adaptive Computational Resource Allocation Strategy

We start to measure the rule importance from generation three (i.e., $g \geqslant 3$, population at generation 1 is randomly initialised, and we do not consider it for avoiding randomness). At a generation, we use the reward obtained by rules so far to decide the number of individuals for learning each rule. The ratios for deciding the number of individuals for the routing rule is shown as below:

$$ratioRouting = \frac{rewardRouting}{rewardRouting + rewardSequencing} \tag{1}$$

Thus, the number of offspring generated per generation for learning the routing rule and sequencing rule is $popsize * ratioRouting$ and $popsize * (1 - ratioRouting)$, respectively. The number of individuals for learning the routing and sequencing rule is adaptive over generations, which are highly related to the rule importance.

4 Experiment Design

Simulation Model: This paper considers to process 6000 jobs including 1000 warm-up jobs with ten machines. The importance of jobs varies which are represented by weight, i.e., 20%, 60%, 20% jobs are with weights 1, 2, and 4, respectively [29]. Each job has a certain number of operations which follows a uniform discrete distribution between one and ten. Each operation can be processed by more than one machine, where the number of options follows a uniform discrete distribution between one and ten. The processing time of each operation follows a uniform discrete distribution with the range [1, 99]. Utilisation level (P) is a factor to simulate different DFJSS scenarios, and a higher utilisation level indicates a busier DFJSS. The utilisation is calculated as $P = \mu * P_M/\lambda$, where μ is the average processing time of machines, P_M is the probability of a job visiting a machine, λ is the rate of the Poisson process for simulating job arrival.

Design of Comparisons: GP, which has an equal number of individuals for learning the routing and sequencing rules, is selected as a baseline for comparison. The algorithm that gives the *important* rule more individuals is named IGP. To measure the performance of IGP, IGP will be compared with GP. To further verify the effectiveness of IGP, we compare with a reverse algorithm named UNIGP that gives *unimportant* rule more individuals by swapping the number of individuals for routing and sequencing rules obtained by the proposed individual allocation strategy. The scenarios with utilisation levels of 0.75, 0.85 and 0.95 are used to measure the performance of algorithms. The scenarios are represented as <objective, utilisation level> such as <Fmean, 0.75>.

Parameter Settings: All the algorithms have 1000 individuals with two subpopulations. Each subpopulation is 500 individuals. IGP and UNIGP have an adaptive number of individuals across generations. Each individual of the algorithm consists of terminals, i.e., shown in Table 1 [21], and functions, i.e., +,

Table 1. The terminal and function sets.

	Terminals	Description
Machine-related	NIQ	The number of operations in the queue
	WIQ	Current work in the queue
	MWT	Waiting time of a machine
Operation-related	PT	Processing time of an operation
	NPT	Median processing time for next operation
	OWT	Waiting time of an operation
Job-related	WKR	Median amount of work remaining of a job
	NOR	The number of operations remaining of a job
	W	Weight of a job
	TIS	Time in system

Table 2. The mean (standard deviation) of objective values on test instances of GP, IGP, and UNIGP according to 30 independent runs in nine scenarios.

Scenarios	GP	IGP	UNIGP
<Fmean, 0.75>	336.23(1.26)	335.63(1.07)(↑)	335.94(1.19)(≈)(≈)
<Fmean, 0.85>	384.69(1.63)	383.79(1.50)(↑)	386.97(4.06)(↓)(↓)
<Fmean, 0.95>	550.94(5.79)	549.69(2.95)(≈)	558.08(9.64)(↓)(↓)
<Tmean, 0.75>	13.28(0.40)	13.09(0.29)(↑)	13.76(0.77)(↓)(↓)
<Tmean, 0.85>	40.27(1.85)	39.56(0.82)(≈)	42.15(2.92)(↓)(↓)
<Tmean, 0.95>	175.49(2.85)	174.25(2.43)(↑)	182.88(6.94)(↓)(↓)
<WTmean, 0.75>	27.04(1.05)	26.66(1.02)(↑)	27.71(2.22)(≈)(↓)
<WTmean, 0.85>	75.82(3.83)	74.46(1.90)(↑)	76.57(4.37)(≈)(↓)
<WTmean, 0.95>	294.58(9.65)	290.45(6.10)(↑)	303.93(15.40)(↓)(↓)
Average rank	2	**1.51**	2.49

* An algorithm is compared with its left algorithm(s) one by one if has.

$-$, $*$, protected /, max, min. The initialised GP programs are generated by the ramp-half-and-half method with a minimal (maximal) depth of 2 (6). The depths of all programs are no more than 8. Tournament selection with size 7 is used to select parents for producing offspring. The new offspring are generated by elites of a value of 10, and crossover, mutation and reproduction with rates 80%, 15%, and 5%, respectively. The maximal number of generations of algorithms is 51.

5 Results and Discussions

We use the results from 30 independent runs to verify the performance of the proposed algorithm. We apply the Friedman test to see whether there is a significant difference among algorithms. If yes, then Wilcoxon test with a significance level of 0.05. is used to compare two algorithms, and "↑", "↓", "≈" indicate an algorithm is better, worse or similar with the compared algorithm.

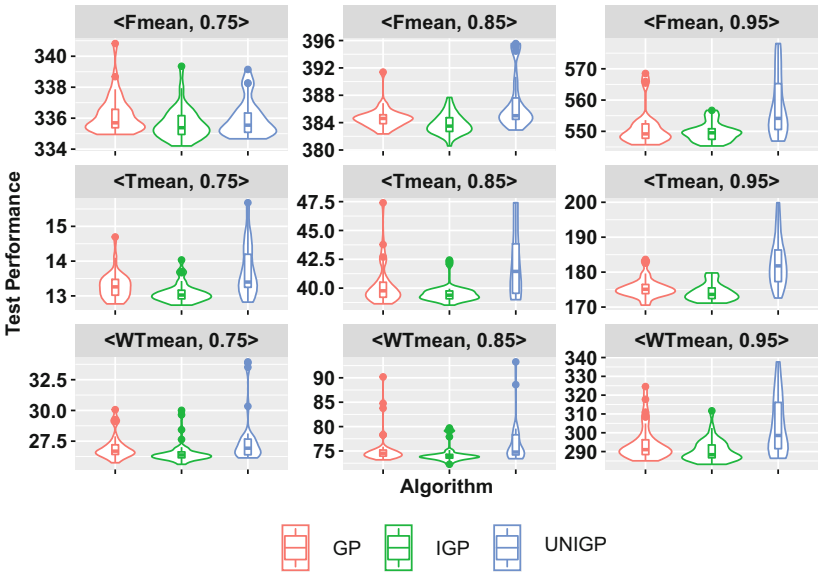

Fig. 3. Violin plots of the obtained objective values on test instances of GP, IGP and UNIGP according to 30 independent runs in nine scenarios.

Quality of Learned Scheduling Heuristics: Table 2 shows the mean and standard deviations of objective values on unseen instances of GP, IGP and UNIGP over 30 independent runs in nine scenarios. The results show that IGP is significantly better than GP in most of the examined scenarios. This verifies the effectiveness of the proposed algorithm with adaptive computational resources allocation strategy. In addition, UNIGP is much worse than baseline GP and IGP which is as expected, since UNIGP applies the opposite idea from IGP. This verifies the proposed algorithm from a reverse point of view. Overall, we can see that IGP is the best algorithm among them with the smallest rank value of 1.51. Figure 3 shows the violin plots of obtained test objective values of GP, IGP, and UNIGP based on 30 independent runs in nine scenarios. We can see that the proposed algorithm IGP shows its superiority and achieves the best performance with a lower objective distribution.

Accumulated Rewards for Routing and Sequencing Rules: Figure 4 shows the curves of average accumulated reward values of IGP-Routing and IGP-Sequencing based on 30 independent runs in nine scenarios. It is clear that the reward values for the routing rule are increasing steadily along with the generations. However, there is only a small increase on the reward values of the sequencing rules in most of the scenarios. The results show that the routing rule plays a more important role than the sequencing rule in DFJSS. The proposed algorithm is expected to give more individuals for learning the routing rules.

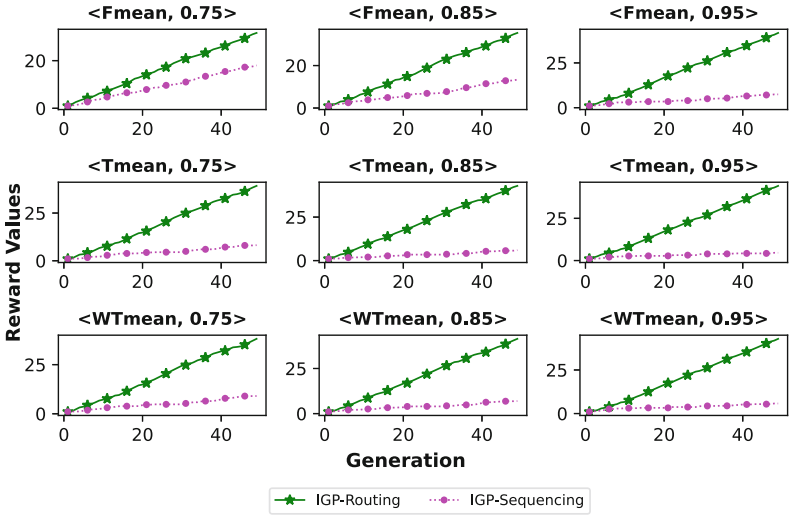

Fig. 4. Curves of average accumulated reward values of IGP-Routing and IGP-Sequencing according to 30 independent runs in nine scenarios.

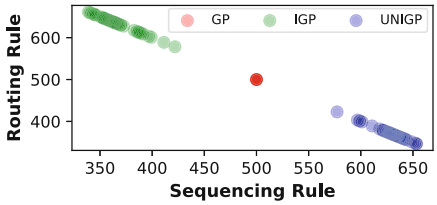

Fig. 5. Scatter plots of the number of individuals for learning the routing rule and the sequencing rule across all generations of GP, IGP, and UNIGP.

The Number of Individuals for Learning Rules: Figure 5 shows the scatter plots of individuals for learning the routing and sequencing rule across all generations of GP, IGP and UNIGP in scenario <Fmean, 0.75>. For GP, a fixed number of individuals are equally set for learning the routing rule (i.e., 500 individuals) and the sequencing rule (i.e., 500 individuals). IGP gives more individuals for learning the routing rule, which UNIGP biases more on the learning on the sequencing rule. The results show that the routing rule is more important than the sequencing rule in DFJSS. Furthermore, the proposed algorithm IGP can adaptively allocate more individuals for learning the routing rule. Similar pattern is also found in other scenarios.

Ratios of Number of Individuals for Learning the Routing Rule: Figure 6 shows the curves of average ratios of the number of individuals for learning the routing rule along with generations of IGP and UNIGP. At the first two generations, the ratios of the number of individuals for learning routing

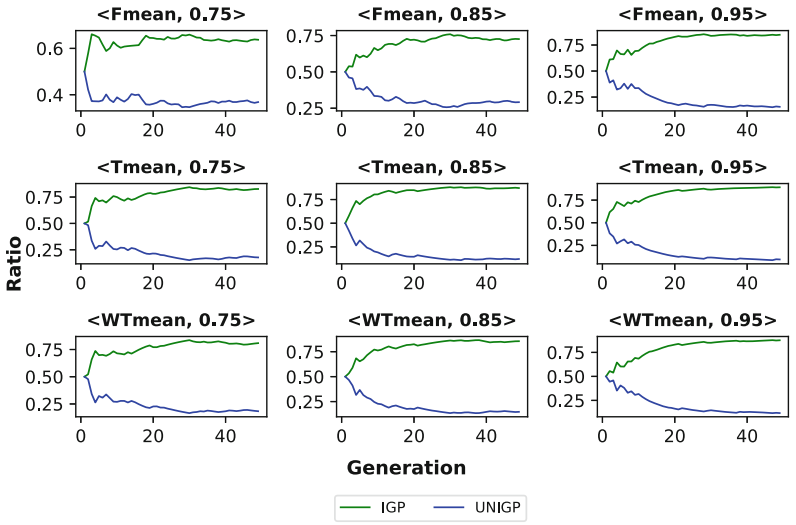

Fig. 6. Curves of the average ratios of the number of individuals for learning the routing rule along with generations of IGP and UNIGP according to 30 independent runs.

and sequencing rules are the same, which are 0.5 in all scenarios (computational resource allocation starts at generation three). From generation three, IGP starts to increase the ratios of the number of individuals for learning routing rules. After generation 20, the ratios arrive at a relatively steady state, which are around 0.85 in most scenarios. UNIGP has shown the opposite trend, where the ratios of the number of individuals for learning the routing rule keep decreasing to about 0.15 at generation 20 and stay at a relatively constant number after that.

Comparison with Algorithms with Fixed Number of Individuals: Based on the discussion in the previous section, we can see that the found ratios of individuals for the routing rule are around 0.85. In other words, about 850 (i.e., $1000 * 0.85$) individuals are used by IGP for learning the routing rule. It is interesting to know whether fixing the number of individuals for learning rules can get the same performance as IGP or not. To investigate this, we compare IGP with GP500, GP650 and GP850, where 500 (500), 650 (350), and 850 (150) are the number of individuals for learning the routing (sequencing) rule. We choose the most complex scenarios (i.e., <WTmean, 0.75>, <WTmean, 0.85>, and <WTmean, 0.95>) that consider the job importance for this investigation. Figure 7 shows the curves of the average objective values of GP500, GP650, GP850 and IGP on test instances in mean-weighted-tardiness related scenarios over 30 independent runs. The results show that GP850 performs better than GP500 and GP650 in most cases. This indicates that finding a good threshold for the number of individuals for learning rules can improve the performance. In addition, the results also show that IGP shows its superiority compared with all other algorithms in terms of the convergence speeds and final performance.

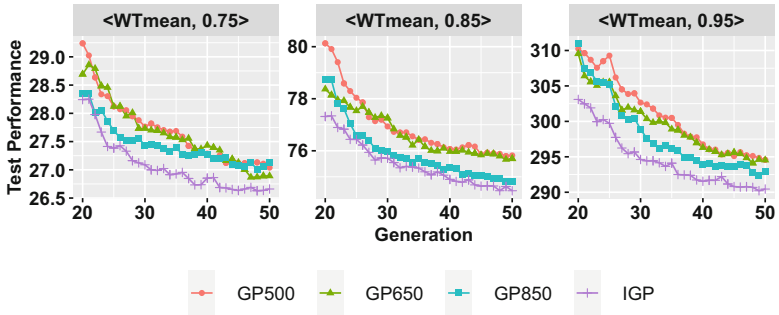

Fig. 7. Curves of average objective values of GP500, GP650, GP850 and IGP on test instances in mean-weighted-tardiness related scenarios based on 30 independent runs.

We can see that simply fixing the number of individuals for learning rules is not effective as an adaptive computational resource allocation strategy. One possible reason is that the importance of the routing rule and sequencing rules' importance may differ on different instances at different generations. Another possible reason is that although the routing rule and the sequencing rule differ in importance to the schedule, it is still necessary to allocate enough resources to learn the less important rule to get a good enough rule before generation 20 as shown in Fig. 6 (i.e., the schedule quality in DFJSS depends on two rules). The superior performance of IGP verifies the effectiveness of the proposed algorithm to detect rule importance and allocate computational resources automatically and adaptively.

Sizes of the Learned Scheduling Heuristics: To verify the effect of the proposed IGP on learned scheduling heuristics, this section investigates rule size. We use the number of nodes for measuring the rule sizes [30]. Since the routing rule and the sequencing rule work together in DFJSS, it is reasonable to look at the sum of their rule sizes. We find that there is no significant difference between the rule sizes (routing rule plus sequencing rule) of GP and IGP, such as with the mean and standard deviation of 98.53(23.93) and 101.87(21.83)($\approx$) in <Fmean, 0.75>. Figure 8 shows the violin plots of the average sizes of routing rules and sequencing rules over population in nine scenarios. Overall, the results show that the routing rules are larger than the sequencing rules for both GP and IGP. This also demonstrates that the routing rule is more important than the sequencing rule. We can also see that there is an increase on the routing rule sizes of IGP compared with GP, especially in the three scenarios with Tmean (i.e., <Tmean, 0.75>, <Tmean, 0.85> and <Tmean, 0.95>). This trend is clearer in the scenarios with higher utilisation levels. In contrast with the increase of the routing rule size, the sequencing rule size becomes smaller in most of the scenarios. This indicates that using more (less) resources on learning the rule can increase (decrease) the corresponding rule size.

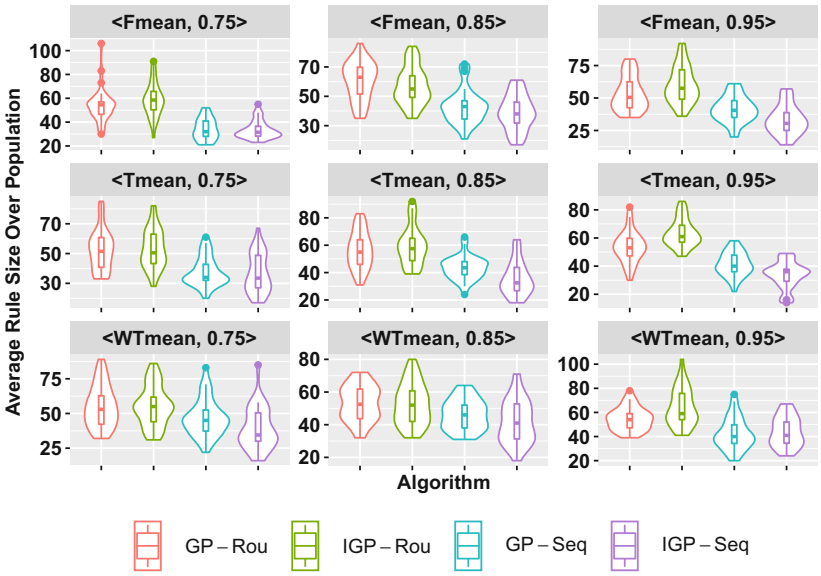

Fig. 8. Violin plots of the sizes of average routing rules and sequencing rules over population of GP and IGP in nine scenarios.

6 Conclusions and Future Work

The goal of this paper is to develop an effective importance-aware scheduling heuristics learning GP approach to automatically learn the routing and sequencing rules for DFJSS. The goal has been achieved by proposing a novel rule importance measure, and an adaptive strategy to allocate computational resources, i.e., GP individuals, for learning the routing rule and the sequencing rule.

The results show that the importance of the routing rule and the sequencing rule differs, and the routing rule is more important than the sequencing rule in the examined DFJSS scenarios. The proposed rule importance strategy based on the improvement of fitness across generations can detect the rule importance in DFJSS properly. Furthermore, the developed adaptive computational resources allocation strategy based on the rule importance measure has successfully optimised the learning process for the routing rule and the sequencing rule. The effectiveness of the proposed IGP has also been verified by the analyses in terms of the exact number and the ratios of allocated individuals for rules, the accumulated reward for rules, and the comparison with the algorithms with a fixed number of individuals for learning rules. Further analyses show that there is no significant difference between the rule size of the pairs of routing rule and sequencing rule, however, the routing (sequencing) rule obtained by the proposed algorithm is larger (smaller) than compared algorithms. In addition, we observe that the routing rule is normally larger than the sequencing rule learned by GP algorithms, which can also be an indicator of the importance of the routing rule.

Some interesting directions can be further investigated in the near future. The rule importance in different DFJSS scenarios may differ. For example, the sequencing rule might be more important than the routing rule if there are a small number of machines. More comprehensive analyses are needed. Moreover, this paper confirms that the importance of the routing rule and the sequencing rule can differ. A more advanced strategy to improve the overall decision marking in DFJSS by recognising such differences is worth investigating.

References

1. Nie, L., Gao, L., Li, P., Li, X.: A GEP-based reactive scheduling policies constructing approach for dynamic flexible job shop scheduling problem with job release dates. J. Intell. Manuf. **24**(4), 763–774 (2013)
2. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: Guided subtree selection for genetic operators in genetic programming for dynamic flexible job shop scheduling. In: Hu, T., Lourenço, N., Medvet, E., Divina, F. (eds.) EuroGP 2020. LNCS, vol. 12101, pp. 262–278. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-44094-7_17
3. Zhang, F., Nguyen, S., Mei, Y., Zhang, M.: Genetic Programming for Production Scheduling. MLFMA, Springer, Singapore (2021). https://doi.org/10.1007/978-981-16-4859-5
4. Nguyen, S., Zhang, M., Johnston, M., Chen Tan, K.: Hybrid evolutionary computation methods for quay crane scheduling problems. Comput. Oper. Res. **40**(8), 2083–2093 (2013)
5. Hart, E., Ross, P., Corne, D.: Evolutionary scheduling: a review. Genet. Program Evolvable Mach. **6**(2), 191–220 (2005)
6. Jaklinović, K., Durasević, M., Jakobović, D.: Designing dispatching rules with genetic programming for the unrelated machines environment with constraints. Exp. Syst. Appl. **172**, 114548 (2021)
7. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: Correlation coefficient-based recombinative guidance for genetic programming hyperheuristics in dynamic flexible job shop scheduling. IEEE Trans. Evol. Comput. **25**(3), 552–566 (2021). https://doi.org/10.1109/TEVC.2021.3056143
8. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Stat. Comput. **4**(2), 87–112 (1994)
9. Burke, E.K., et al.: Hyper-heuristics: a survey of the state of the art. J. Oper. Res. Soc. **64**(12), 1695–1724 (2013)
10. Braune, R., Benda, F., Doerner, K.F., Hartl, R.F.: A genetic programming learning approach to generate dispatching rules for flexible shop scheduling problems. Int. J. Prod. Econ. **243**, 108342 (2022)
11. Pillay, N., Qu, R.: Hyper-Heuristics: Theory and Applications. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96514-7
12. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling. IEEE Trans. Cybern. **52**(8), 8142–8156 (2022). https://doi.org/10.1109/TCYB.2021.3050141

13. Zhang, F., Mei, Y., Nguyen, S., Zhang, M., Tan, K.C.: Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. IEEE Trans. Evol. Comput. **25**(4), 651–665 (2021)
14. Zhang, F., Mei, Y., Nguyen, S., Tan, K.C., Zhang, M.: Multitask genetic programming-based generative hyper-heuristics: a case study in dynamic scheduling. IEEE Trans. Cybern. (2021). https://doi.org/10.1109/TCYB.2021.3065340
15. Shen, X., Guo, Y., Li, A.: Cooperative coevolution with an improved resource allocation for large-scale multi-objective software project scheduling. Appl. Soft Comput. **88**, 106059 (2020)
16. Ren, Z., Liang, Y., Zhang, A., Yang, Y., Feng, Z., Wang, L.: Boosting cooperative coevolution for large scale optimization with a fine-grained computation resource allocation strategy. IEEE Trans. Cybern. **49**(12), 4180–4193 (2018)
17. Yang, M., et al.: Efficient resource allocation in cooperative co-evolution for large-scale global optimization. IEEE Trans. Evol. Comput. **21**(4), 493–505 (2017). https://doi.org/10.1109/TEVC.2016.2627581
18. Jia, Y.-H., Mei, Y., Zhang, M.: Contribution-based cooperative co-evolution for nonseparable large-scale problems with overlapping subcomponents. IEEE Trans. Cybern. **52**(6), 4246–4259 (2020). https://doi.org/10.1109/TCYB.2020.3025577
19. Zhang, X.-Y., Gong, Y.-J., Lin, Y., Zhang, J., Kwong, S., Zhang, J.: Dynamic cooperative coevolution for large scale optimization. IEEE Trans. Evol. Comput. **23**(6), 935–948 (2019)
20. Brucker, P., Schlie, R.: Job-shop scheduling with multi-purpose machines. Computing **45**(4), 369–375 (1990)
21. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: A preliminary approach to evolutionary multitasking for dynamic flexible job shop scheduling via genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 107–108. ACM (2020)
22. Durasevic, M., Jakobovic, D.: A survey of dispatching rules for the dynamic unrelated machines environment. Exp. Syst. Appl. **113**, 555–569 (2018)
23. Hart, E., Sim, K.: A hyper-heuristic ensemble method for static job-shop scheduling. Evol. Comput. **24**(4), 609–635 (2016)
24. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: Genetic programming with adaptive search based on the frequency of features for dynamic flexible job shop scheduling. In: Paquete, L., Zarges, C. (eds.) EvoCOP 2020. LNCS, vol. 12102, pp. 214–230. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-43680-3_14
25. Yska, D., Mei, Y., Zhang, M.: Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling. In: Castelli, M., Sekanina, L., Zhang, M., Cagnoni, S., García-Sánchez, P. (eds.) EuroGP 2018. LNCS, vol. 10781, pp. 306–321. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-77553-1_19
26. Zhang, F., Mei, Y., Zhang, M.: 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. ACM (2019)
27. Hildebrandt, T., Heger, J., Reiter, B.S.: Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. In: Proceedings of the Conference on Genetic and Evolutionary Computation, pp. 257–264. ACM (2010)

28. Zhang, F., Mei, Y., Nguyen, S., Tan, K.C., Zhang, M.: Instance rotation based surrogate in genetic programming with brood recombination for dynamic job shop scheduling. IEEE Trans. Evol. Comput. (2022). https://doi.org/10.1109/TEVC.2022.3180693
29. Hildebrandt, T., Branke, J.: On using surrogates with genetic programming. Evol. Comput. **23**(3), 343–367 (2015)
30. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. IEEE Trans. Cybern. **51**(4), 1797–1811 (2021)