





# Genetic Programming with Adaptive Search Based on the Frequency of Features for Dynamic Flexible Job Shop Scheduling

Fangfang Zhang<sup>1</sup> , Yi Mei<sup>1</sup> , Su Nguyen<sup>2</sup> , and Mengjie Zhang<sup>1</sup> 

<sup>1</sup> 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

<sup>2</sup> Centre for Data Analytics and Cognition, La Trobe University, Victoria 3086,  
Melbourne, Australia  
P.Nguyen4@latrobe.edu.au

**Abstract.** Dynamic flexible job shop scheduling (DFJSS) is a very valuable practical application problem that can be applied in many fields such as cloud computing and manufacturing. In DFJSS, machine assignment and operation sequencing decisions need to be made simultaneously in dynamic environments with unpredicted events such as new job arrivals. Scheduling heuristic is an ideal candidate for solving the DFJSS problem due to its efficiency and simplicity. Genetic programming (GP) has been successfully applied to evolve scheduling heuristics for job shop scheduling automatically. However, GP has a huge search space, and the traditional search algorithms do not utilise effectively the information obtained from the evolutionary process. This paper proposes a new method to make better use of the information during the evolutionary process of GP to further enhance the ability of GP. To be specific, this paper proposes two adaptive search strategies based on the frequency of features in promising individuals to guide GP to evolve effective rules. This paper examines the proposed algorithm on six different DFJSS scenarios. The results show that the proposed GP with adaptive search can converge faster and achieve significantly better performance than the GP without adaptive search in most scenarios while no worse in all other scenarios without increasing the computational cost.

**Keywords:** Adaptive search · Scheduling heuristic · Dynamic flexible job shop scheduling · Genetic programming.

## 1 Introduction

Job shop scheduling (JSS) [1] is an important combinatorial optimisation problem, which has essential roles in all walks of life such as manufacturing [2,3] and cloud computing [4]. The task in JSS is to process a number of jobs by a set of machines. Each job has a sequence of operations. The goal is to optimise the machine resources to achieve the objectives such as minimising the max-flowtime.

Flexible JSS (FJSS) [5] is a variant of JSS which is better to reflect requirements in real-world applications. In FJSS, an operation can be processed on a set of machines. It indicates that two decisions need to be made simultaneously. One is *machine assignment* (i.e. assign an operation to a particular machine), and the other is *operation sequencing* (i.e. choose an operation as the next operation to be processed by an idle machine). In addition, many practical scheduling problems are dynamically changing over time, for example, due to new job arrivals [6,7]. Dynamic FJSS (DFJSS) is to consider FJSS under dynamic environments.

Scheduling heuristics such as dispatching rules [8] are widely used to handle such kinds of dynamic problems. A scheduling heuristic is a heuristic that works like a priority function to evaluate the priorities of operations and machines. To be specific, in DFJSS, a machine that has the highest priority value based on the *routing rule* (i.e. routing scheduling heuristic) will be assigned a job. An operation with the highest priority value based on the *sequencing rule* (i.e. sequencing scheduling heuristic) will be chosen as the next operation to be processed. There are some rules such as SPT (i.e. shortest processing time) and WIQ (i.e. the workload in the queue of a machine) which have been identified as effective rules for JSS. However, they are manually designed by experts, which is time-consuming and not always available. In practical, it is hard to manually design effective rules due to the complexity of the job shop environments.

Genetic programming (GP) [9], as a hyper-heuristic (GPHH) method, has been successfully applied to automatically evolve scheduling heuristic for JSS [10,11]. As a population-based algorithm, GP tries to improve the scheduling heuristics (i.e. individuals) generation by generation. In traditional GP, features are randomly chosen to build subtrees for mutation and generate individuals. However, the importance of features can be different. Such a way that chooses all the features randomly cannot fully play its role because the importance of the features is ignored. The challenge is that the search space of GP is huge (i.e. the individual can be a big tree), and the traditional search might not be enough. This paper proposes the adaptive search to guide GP to the more promising region by utilising the information during the evolutionary process. The proposed algorithm aims to guide the behaviour of GP over time adaptively.

The key to the success of GP is that it can automatically detect important features and optimise the structure of individuals guided by the fitness function. From an evolutionary perspective, the individuals themselves, especially good individuals, contain useful information which can be further utilised to improve evolutionary efficiency. An advantage is that information generated during the evolutionary process can be easily used without putting more extra effort to get the information. In this paper, the frequency of features based on the individuals that have good performance will be further used to guide GP to find more effective rules for DFJSS adaptively. To this end, two adaptive search strategies which can be realised by mutation and re-initialisation will be proposed.

The overall goal of this paper is to develop effective adaptive search strategies with the frequency of features to guide GP to find effective scheduling heuristics for DFJSS efficiently. The proposed algorithms are expected to speed up

the convergence of GP and find effective rules without additional computing requirement. In particular, this paper has the following research objectives:

- Develop adaptive search strategies with the frequency of features in promising individuals to guide GP towards the more promising areas.
- Verify the effectiveness and efficiency of the proposed GP algorithm with the adaptive strategy by comparing its performance and convergence speed with the baseline GP.
- Analyse how the adaptive search affects the evolutionary process of GP.

## 2 Background

### 2.1 Dynamic Flexible Job Shop Scheduling

In FJSS problem,  $n$  jobs  $J = \{J_1, J_2, \dots, J_n\}$  need to be processed by  $m$  machines  $M = \{M_1, M_2, \dots, M_m\}$ . Each job  $J_j$  has an arrival time  $at(J_i)$  and a sequence of operations  $O_j = (O_{j1}, O_{j2}, \dots, O_{ji})$ . Each operation  $O_{ji}$  can only be processed by one of its optional machines  $\pi(O_{ji})$  and its processing time  $\delta(O_{ji})$  depends on the machine that processes it. It indicates that there are two decisions which are routing decision and sequencing decision in FJSS. In DFJSS, not only two decisions need to be made simultaneously, but also the dynamic events are necessary to be taken into account when making schedules. This paper focuses on one dynamic event (i.e. continuously arriving new jobs). That is, the information of a job is unknown until its arrival time.

### 2.2 Genetic Programming Hyper-heuristic for DFJSS

A hyper-heuristic [12] is a heuristic search method that seeks to select or generate heuristics to efficiently solve hard problems. The unique characteristic is that hyper-heuristic works on heuristic search space instead of solution search space.

GP, as a hyper-heuristic method [13], has been successfully applied to more informative scheduling heuristics for combinational optimisation problems such as packing [14,15], timetabling [16,17] and JSS [18,19,20,21]. Scheduling heuristics, including routing and sequencing rules, are needed in DFJSS in our research. To follow the sequence constraint of operations of a job, this paper only starts to allocate an operation when it becomes a *ready operation*. There are two sources of ready operations. One is the first operation of a job. The second is the operation that its proceeding operation is just finished. Once an operation becomes a ready operation (*routing decision point*), it will be allocated to the machine by the routing rule. When a machine becomes idle, and its queue is not empty (*sequencing decision point*), the sequencing rule will be triggered to choose the next operation to be processed.

GP has shown its superiority in DFJSS [18,19]. However, most of the existing works follow the traditional way of the evolutionary process of GP, which may not enough due to its large search space. To this end, this paper introduces the adaptive search to help GP evolve more effective scheduling heuristics (i.e. routing rule and sequencing rule) for DFJSS.

### 3 The Proposed GP with Adaptive Search

In this paper, the adaptive search aims to guide the behaviour of the GP algorithm over time by utilising the information generated during the evolutionary process of GP. It is expected to speed up the convergence of GP and evolve effective rules. It is not trivial to answer “when”, “how”, and “where” to apply the adaptive search. These three research questions are investigated as follows.

*Question 1: When to use the adaptive search?*

This paper uses the adaptive strategy at every generation.

*Question 2: How to use the adaptive search?*

In this paper, the frequency of features is the number of occurrences of features. The number of occurrences of features based on the entire top ten individuals (i.e. roughly 1% of the population size) will be utilised to guide the behaviour of GP to improve its convergence speed and find more promising rules for DFJSS, since top ten individuals have much better fitnesses than others. Based on the number of occurrences of features, the probability of each feature is calculated. The larger the number of the occurrence, the higher the probability that the feature is given. When generating new individuals and subtrees for mutation, the features will be selected based on their probability. The higher the probability, the easier the feature is to be selected for building new trees. The pseudo-code of calculating the probabilities of features are shown in Algorithm 1.

---

**Algorithm 1:** Pseudo-code of calculating the probabilities of features

---

**Input** : Top ten individuals  
**Output:** The probabilities of features *probabilities*

- 1: *probabilities*  $\leftarrow$  null
- 2: **for**  $i = 1$  to  $|featureSize|$  **do**
- 3:     | *occurrence<sub>i</sub>*: count the number of occurrence of a feature in the *top ten*  
       | *individuals*
- 4: **end**
- 5: *sumOccurrences*: sum up the occurrences of all features
- 6: **for**  $i = 1$  to  $|featureSize|$  **do**
- 7:     |  $probability_i = \frac{occurrence_i}{sumOccurrences}$
- 8: **end**
- 9: *probabilities*  $\leftarrow$  *probability<sub>i</sub>*
- 10: **return** *probabilities*

---

*Question 3: Where to use the adaptive search?*

During the evolutionary process of GP, there are two occasions using selected features. One is when building subtree for mutation. The other is when generating new individuals. To guide the behaviour of GP as much as possible, it is straightforward to simply apply the adaptive search by mutation. Another adaptive search strategy related to re-initialisation is also proposed in this paper.

**Mutation.** As a genetic operator, mutation aims to maintain the diversity of the population by replacing one subtree with a randomly generated subtree. The new individual produced by mutation can be very bad (i.e. too random). If the

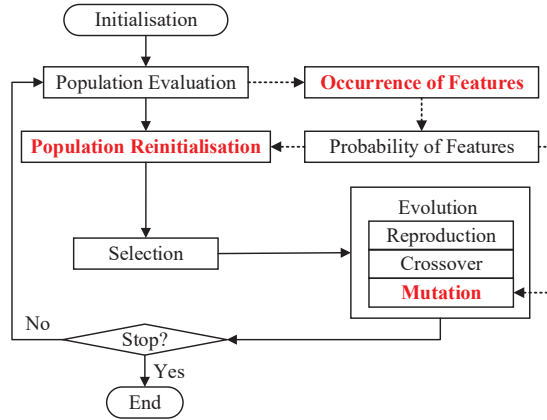


Fig. 1: The flowchart of the proposed GP with the adaptive search.

mutation direction can be guided to some extent, it may enhance the effectiveness of mutation. The subtree that builds with the informed features has such a role because it has a high chance to bring more useful building blocks.

**Re-initialisation.** The quality of individuals in the population can be different. Some individuals have good performance and have a higher chance to be selected as parents to generate offspring. However, there still has a number of individuals that will not contribute too much to the next generation due to their lower performance. These kinds of individuals are useless to some extent. This paper proposes to use a re-initialisation strategy to generate some useful individuals to the population to replace them at each generation. The reinitialised new individuals generated based on the frequency of features are structurally different and have reasonably good fitness.

It is not trivial to decide which individuals to remove from the population. Removing too many individuals takes the risk to lose the quality of the population. Removing very few individuals may not work. This paper uses simulation information to decide which individuals will be removed. If an individual leads to a very long queue of a machine during the simulation, it will be replaced by randomly generated individuals based on the frequency of features. This is because the evolved best rules will not assign a machine lots of operations from our preliminary investigation. In this way, the current population is more likely to have more promising individuals, and thus more capable of generating better offspring for the next generations.

Fig. 1 depicts the general outline of the proposed GP with the adaptive search. GP starts to initialise the population randomly, and then evaluate the individuals in the population. The individuals in the new generation (i.e. offspring) are generated in the evolution stage along with the selection. For the proposed algorithm, there are three main differences compared with standard GP. The first one is that the frequency of features are counted based on the en-

tire top ten individuals after evaluating the individuals. The top ten individuals are better than others obviously, which are good for measuring the frequency of features from our preliminary work. This information is converted into a probability for each feature. The second one is that the re-initialisation strategy is applied to import new potential good individuals into the population by generating new individuals based on the frequency of features. The last one is that the frequency of feature information is utilised to guide the mutation direction. The adaptive strategies are applied over time.

## 4 Experiment Design

### 4.1 Simulation Model

Simulation is a common method to investigate complex real-world problems [22]. This paper assumes there are 5000 jobs that need to be processed by ten machines. The importance of jobs might be different, which are indicated by weights. The weights of 20%, 60%, and 20% of jobs are set as one, two and four, respectively. The number of operations of each job varies by a uniform discrete distribution between one and ten. The processing time of each operation is set by uniform discrete distribution with the range [1, 99]. The number of candidate machines for an operation follows a uniform discrete distribution between one and ten.

In each problem instance, jobs arrive stochastically according to a Poisson process with rate  $\lambda$ . To improve the generalisation ability of the evolved rules for DFJSS problems, the seeds used to generate the jobs are rotated at each generation. In addition, to make sure the accuracy of the collected data, a warm-up period of 1000 jobs is used. If any machine in the system has more than 100 operations, the simulation will be stopped, and the current evaluating individual is replaced by a new individual based on re-initialisation strategy.

### 4.2 Parameter Settings

In our experiment, the terminal and function set are shown in Table 1, following the setting in [23]. The “/” operator is protected division, returning one if divided by zero. The other parameter settings of GP are shown in Table 2.

### 4.3 Comparison Design

Four algorithms are taken into the comparison in this paper. The cooperative coevolution genetic programming (CCGP) [6] which can be used to evolve routing rule and sequencing rule simultaneously, is selected as the baseline algorithm. Our proposed algorithm, which incorporates with adaptive strategy by mutation, is named as MUGP (i.e. generate subtree based on the frequency of features for mutation). The algorithm that incorporates re-initialisation strategy (i.e. reinitialise some useful individuals based on the machine information during

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                      | The waiting time of an operation             |
| Job-related       | WKR                      | The 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                               |
| <b>functions</b>  | $+, -, *, /, \max, \min$ | as usual meaning                             |

Table 2: The parameter setting of GP.

| Parameter                            | Value                            |
|--------------------------------------|----------------------------------|
| Number of subpopulations             | 2                                |
| Subpopulation size                   | 512                              |
| Method for initialising population   | ramped-half-and-half             |
| Initial minimum/maximum depth        | 2 / 6                            |
| Maximal depth of programs            | 8                                |
| The number of elites                 | 10                               |
| Crossover/Mutation/Reproduction rate | 80% / 15% / 5%                   |
| Parent selection                     | Tournament selection with size 7 |
| Number of generations                | 51                               |
| Terminal/non-terminal selection rate | 10% / 90%                        |

the simulation) is named as IMGP. The proposed algorithm, which incorporated by both mutation and re-initialisation, is named as IM<sup>2</sup>GP. MUGP, IMGP and IM<sup>2</sup>GP are compared with CCGP, respectively.

To verify their effectiveness, the proposed algorithms are tested on *six different scenarios*. The scenarios consist of three objectives (e.g. max flowtime, mean flowtime, and mean weighted flowtime) and two utilisation levels (e.g. 0.85 and 0.95). For the sake of convenience, Fmax, Fmean, and WFmean are used to indicate max flowtime, mean flowtime, and mean weighted flowtime, respectively. The objective functions are shown as follows.

- Minimising Fmax =  $\max\{C_1, C_i, \dots, C_n\}$
- Minimising Fmean =  $\frac{\sum_{i=1}^n \{C_i - r_i\}}{n}$
- Minimising WFmean =  $\frac{\sum_{i=1}^n w_i * \{C_i - r_i\}}{n}$

where  $C_i$  is the completion time of job  $J_i$ ,  $r_i$  is the release time of  $J_i$ , and  $w_i$  is the weight of  $J_i$ .

Table 3: The mean (standard deviation) of the objective value of CCGP, MUGP, IMGP, and IM<sup>2</sup>GP over 50 independent runs for six dynamic flexible scenarios.

| Scenario      | CCGP           | MUGP            | IMGP             | IM <sup>2</sup> GP |
|---------------|----------------|-----------------|------------------|--------------------|
| <Fmax,0.85>   | 1212.05(34.68) | 1219.73(29.41)  | 1208.64(30.78)   | 1219.61(40.52)     |
| <Fmax,0.95>   | 1941.98(29.93) | 1946.65(47.62)  | 1934.25(25.90)   | 1942.72(31.19)     |
| <Fmean,0.85>  | 385.95(3.22)   | 384.79(1.39)    | 384.89(2.42)(-)  | 384.57(1.46)(-)    |
| <Fmean,0.95>  | 551.18(5.78)   | 549.66(2.90)    | 551.20(4.70)     | 549.51(4.21)(-)    |
| <WFmean,0.85> | 831.41(6.08)   | 829.43(5.31)(-) | 831.20(6.71)     | 829.26(4.14)(-)    |
| <WFmean,0.95> | 1111.01(12.02) | 1107.59(9.57)   | 1105.44(6.71)(-) | 1105.70(6.95)(-)   |

Note that the evolved best rule at each generation is tested on 50 different test instances, and the mean objective value is reported as the objective value of this best rule. This aims to guarantee the accuracy of measuring the performance.

## 5 Results and Discussions

Fifty independent runs are conducted for the comparison. Wilcoxon rank-sum test with a significance level of 0.05 is used to verify the performance of proposed algorithms. In the following results, “-” and “+” indicate the corresponding result is significantly better or worse than its counterpart. If there is no mark there, that means they have similar performance.

### 5.1 Performance of Evolved Rules

Table 3 shows the mean and standard deviation of the objective value of CCGP, MUGP, IMGP, and IM<sup>2</sup>GP over 50 independent runs for six dynamic flexible scenarios. The performance of MUGP is significantly better than CCGP in one scenario (e.g. <WFmean,0.85>). It indicates that the proposed adaptive strategy with the mutation has the potential to take advantage of the information of the frequency of features. However, it does not work in most scenarios. One possible reason is that the mutation rate is too low (i.e. 0.15) to fully utilise the information. The performance of IMGP is significantly better than that of CCGP in only two scenarios (e.g. <Fmean,0.85> and <WFmean,0.95>). One possible reason is that the reinitialised individuals do not have a big impact on the population. The performance of IM<sup>2</sup>GP is significantly better than CCGP in four scenarios (e.g. <Fmean,0.85>, <Fmean,0.95>, <WFmean,0.85> and <WFmean,0.95>). It indicates that the proposed adaptive strategy with mutation and re-initialisation strategies are more promising. For minimising the max-flowtime, in scenario <Fmax,0.85> and <Fmax,0.95>, there is no significant difference among the three algorithms.

Fig. 2 shows the convergence curves of the average objective value on the test instances of CCGP, MUGP, IMGP, and IM<sup>2</sup>GP over 50 independent runs. Except for max-flowtime related scenarios, IM<sup>2</sup>GP can converge faster and achieve



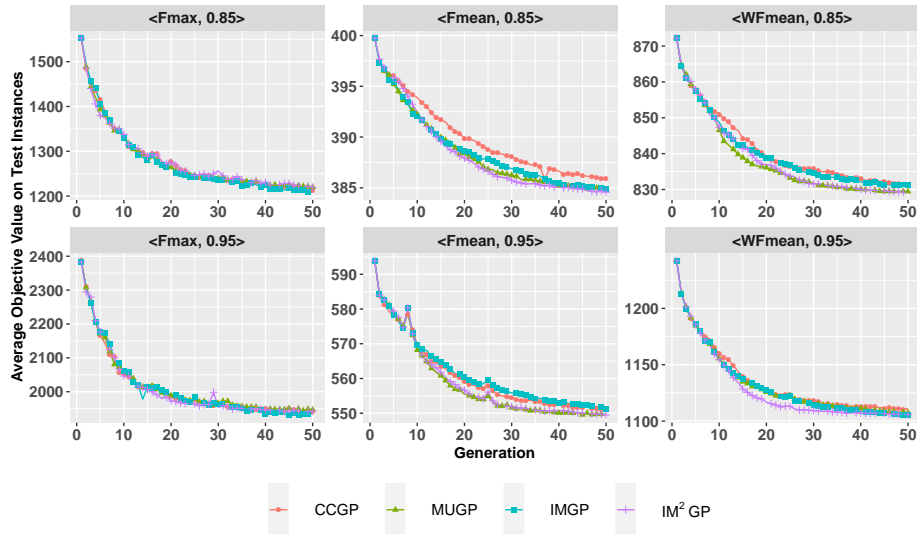


Fig. 2: The convergence curves of CCGP, MUGP, IMGP, and  $IM^2GP$  over 50 independent runs in six scenarios.

better performance than that of CCGP. For minimising max-flowtime, the proposed three algorithms have no obvious advantages. It might be because max-flowtime is more sensitive to the worst case, which is more complex and hard to optimise.

## 5.2 Unique Feature Analyses

The number of unique features in the rules is one of the indicators of the complexity of evolved rules. The number of unique feature means the least number of elements that is needed to construct the rules. A rule with a smaller number of features is easier to be interpreted [24].

Fig. 3 and Fig. 4 show the violin plot of the number of unique features of routing and sequencing rules obtained by CCGP, MUGP, IMGP, and  $IM^2GP$  over 30 independent runs in different scenarios. Violin plots are similar to box plots, except that they also show the probability density of the data at different values, usually smoothed by a kernel density estimator. From an overall view, both for the routing rule and sequencing rule, the rules obtained by MUGP and  $IM^2GP$  involve a smaller set of features. For the routing rule, there is no statistical difference between MUGP and  $IM^2GP$  in most scenarios except the scenario  $\langle Fmean, 0.85 \rangle$ . For the sequencing rule, there is no statistical difference between MUGP and  $IM^2GP$  in all the scenarios. It indicates that the adaptive search strategy only with mutation still can have a significant influence on the unique number of features, although the mutation rate is low. However, both for routing and sequencing rules, the unique number of features of IMGP is similar with that of CCGP in all scenarios.

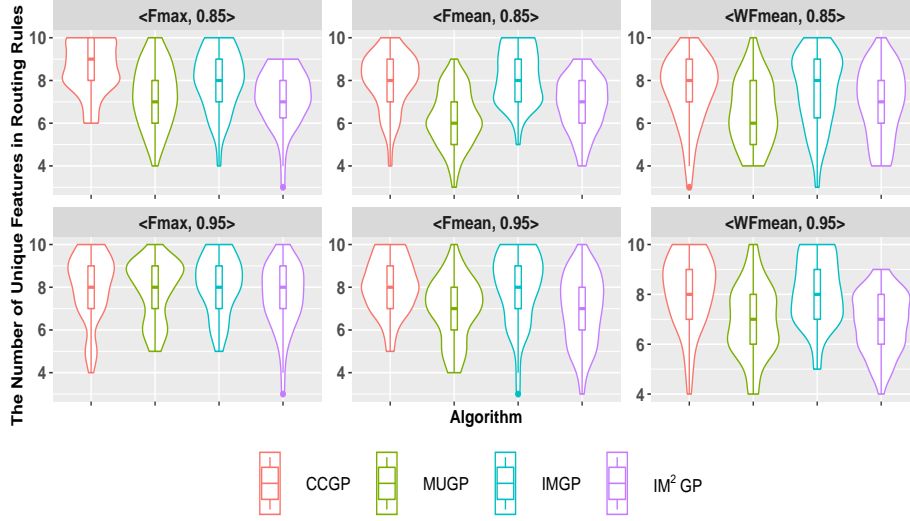


Fig. 3: Violin plot of the number of unique features of *routing rules* obtained by CCGP, MUGP, IMGP, and IM<sup>2</sup>GP over 30 independent runs in six scenarios.

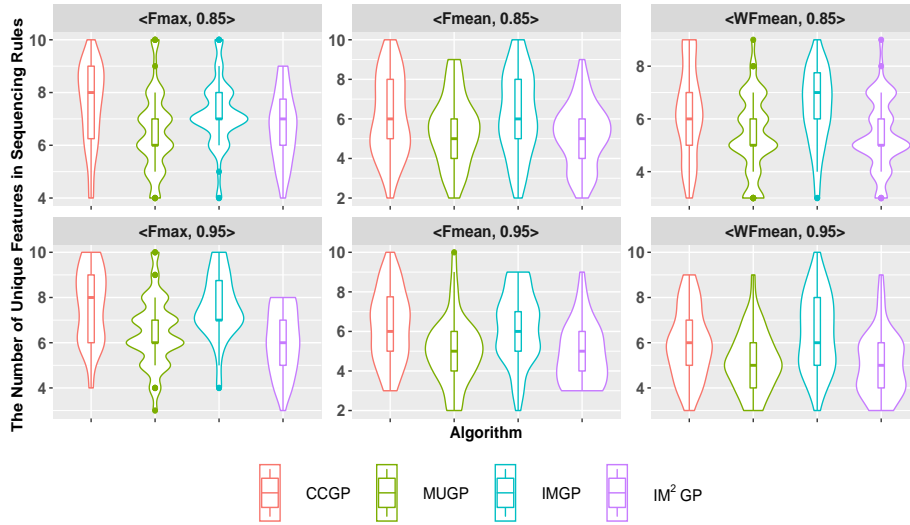


Fig. 4: Violin plot of the number of unique features of *sequencing rules* obtained by CCGP, MUGP, IMGP, and IM<sup>2</sup>GP over 30 independent runs in six scenarios.

### 5.3 The Frequency of Features

Fig. 5 shows the curves of the frequency of features in routing rules during the evolutionary process of IM<sup>2</sup>GP. It shows that the MWT (i.e. machine waiting time) is the most important feature for the routing rules in all scenarios. The importance of MWT is much higher than other features. In the scenarios whose utilisation levels are 0.85, WIQ (i.e. the workload in the queue) plays a secondary role. In the scenarios whose have a higher utilisation level (i.e. 0.95), NIQ (i.e. the number of operations in the queue) plays a significant role. Intuitively, both WIQ and NIQ are important indicators for measuring the workload for machines, they might have the same functions, and one might take over another one. However, we do not know how they work in different scenarios. It is interesting to see that the role of NIQ is significantly higher than that of WIQ in the scenarios that have higher utilisation level. One possible reason is that NIQ is an important factor in busy scenarios.

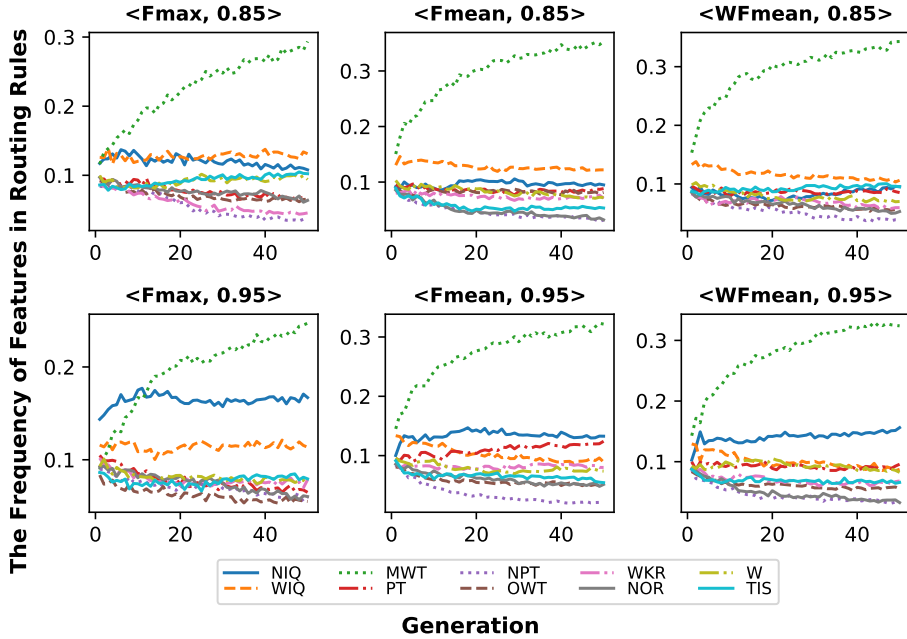


Fig. 5: The curves of the frequency of features in *routing rules* during the evolutionary process of IM<sup>2</sup>GP.

Fig. 6 shows the curves of the frequency of features in sequencing rules during the evolutionary process of IM<sup>2</sup>GP. Different from routing rules, there are three features (e.g. WKR, TIS, and PT) play a vital role in minimising max-flowtime. PT and WKR also are two important features in minimising mean-flowtime and

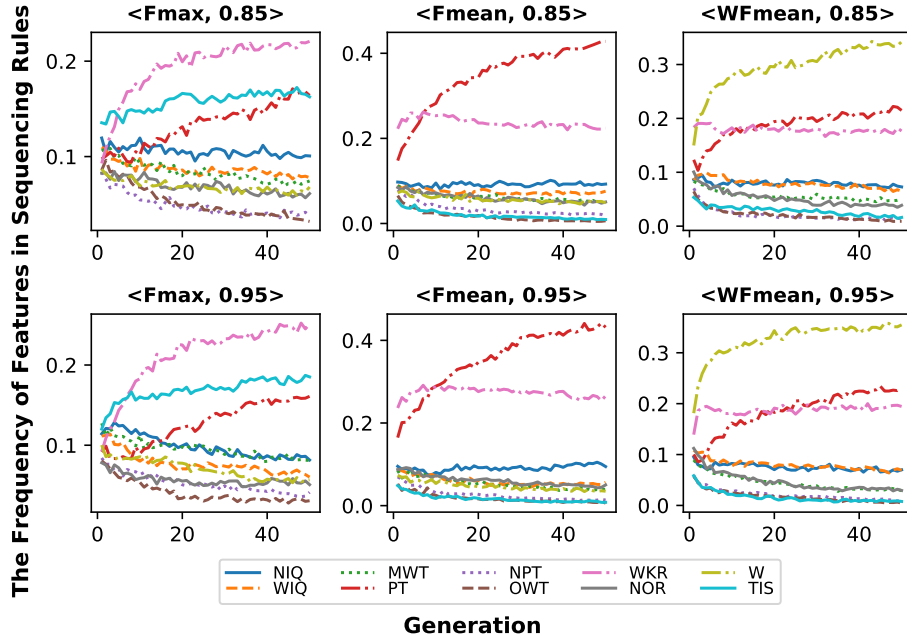


Fig. 6: The curves of the frequency of features in *sequencing rules* during the evolutionary process of IM<sup>2</sup>GP.

weighted mean-flowtime. Except for them, W plays a dominant role in weighted mean-flowtime, which is consistent with our intuition. Besides, W plays its role mainly in sequencing rules instead of routing rules.

It is interesting to see the trend of the feature frequency without adaptive strategies. Fig. 7 shows the curves of the frequency of features of the routing rules that evolved by CCGP. Comparing Fig. 5 and Fig. 7, both IM<sup>2</sup>GP and CCGP can detect important features and use them to build individuals, this is the advantage of GP itself. The difference is that IM<sup>2</sup>GP can further enhance this ability. Fig. 7 shows that the frequency of feature MWT is much higher than that of CCGP (i.e. the most obvious one) during the evolutionary process (i.e. generation 50). For IM<sup>2</sup>GP, in the scenarios with utilisation level 0.85, the frequency of WIQ is higher than other features, which is not that clear for CCGP. Besides, in the scenarios with utilisation level 0.95, the importance of NIQ is easier to be distinguished than that of in CCGP.

#### 5.4 Reinitialised Individuals

Fig. 8 shows the curves of the number of reinitialised routing rules. In all the scenarios, in the beginning, there are a lot of reinitialised individuals in the population. As the number of generations increases, the number of reinitialised

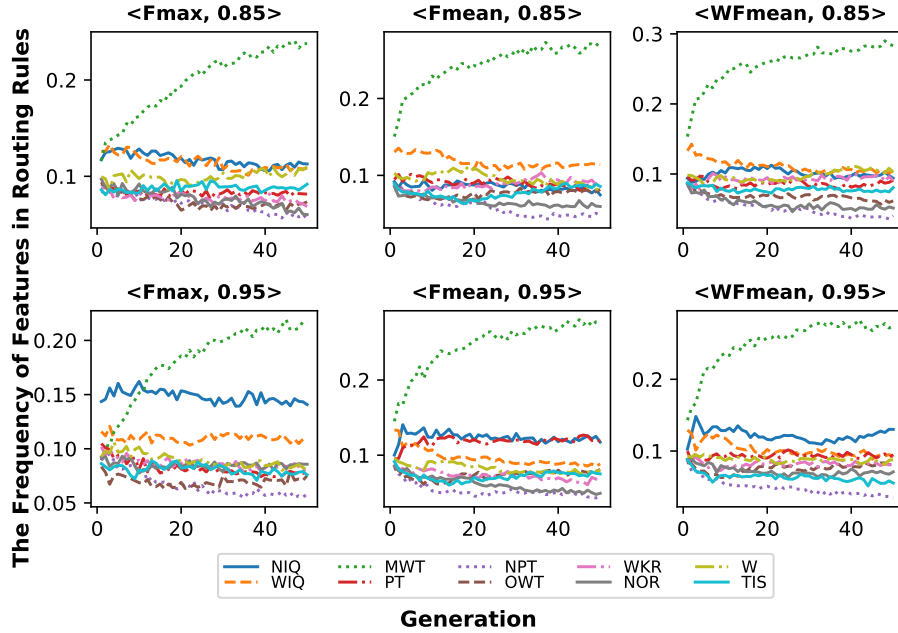


Fig. 7: The curves of the frequency of features in *routing rules* during the evolutionary process of CCGP.

routing rules is getting smaller and smaller. After the fifteenth generation, there is no significant change in the number of reinitialised routing rules.

Fig. 9 shows the curves of the number of reinitialised sequencing rules. Different from routing rules, the sequencing rules are seldom detected as bad rules. This is in line with our expectations. When evaluating sequencing rules, the best routing rule is used as the collaborator, the probability of a machine that is assigned lots of operations is small. Only when a sequencing rule is quite bad, it might be detected as a bad rule. But for the routing rule, even the best sequencing rule is chosen as the collaborator, there are different routing rules which can lead to a high probability of a machine that is assigned lots of operations.

### 5.5 Training time

Table 4 shows the mean and standard deviation of training time of the four algorithms over 50 independent runs in six scenarios. There is no significant difference between the four algorithms. It means the proposed adaptive search strategies do not need extra computational cost. This verifies the advantages of utilising the information generated during the evolutionary process of GP.

In general, IM<sup>2</sup>GP can speed up the convergence and achieve effective rules in most scenarios without extra computational cost, which confirms its effectiveness and efficiency.

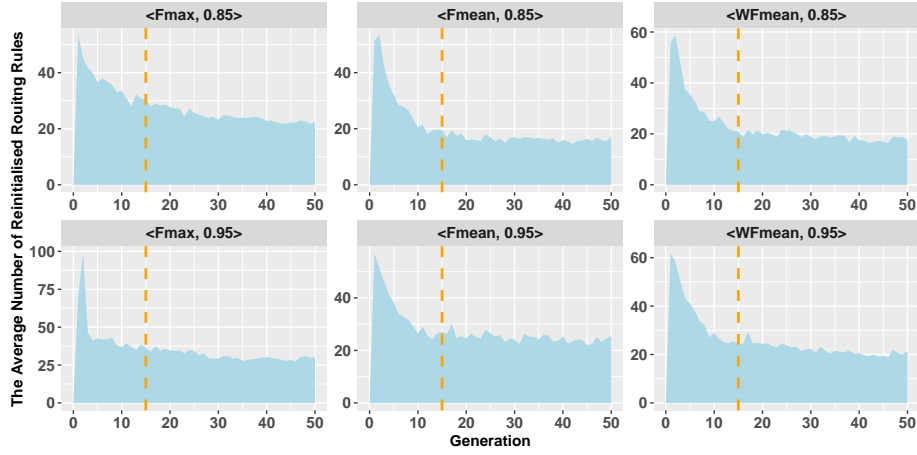


Fig. 8: The curves of the number of *reinitialised routing rules* of IM<sup>2</sup>GP over 50 independent runs in six different scenarios.

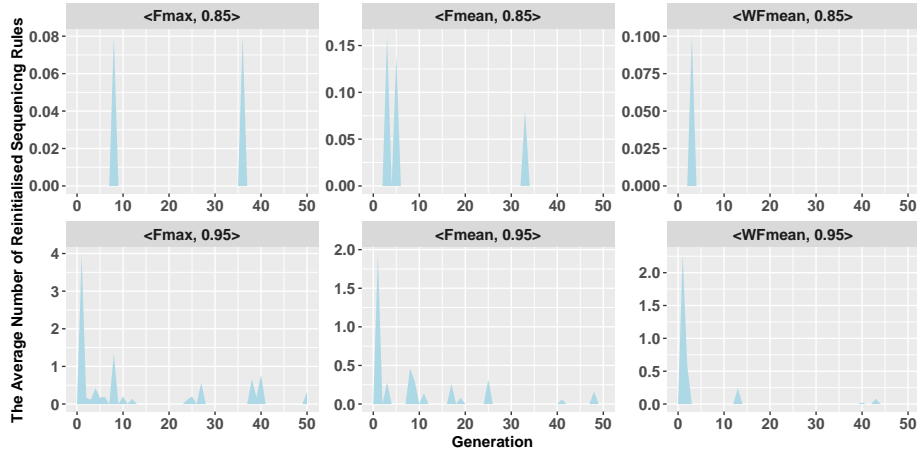


Fig. 9: The curves of the number of *reinitialised sequencing rules* of IM<sup>2</sup>GP over 50 independent runs in six different scenarios.

Table 4: The mean (standard deviation) of **training time** (in minutes) obtained by over 50 independent runs for six different scenarios.

| Scenario      | CCGP   | MUGP   | IMGP   | IM <sup>2</sup> GP |
|---------------|--------|--------|--------|--------------------|
| <Fmax,0.85>   | 73(9)  | 75(12) | 75(10) | 75(11)             |
| <Fmax,0.95>   | 87(15) | 83(12) | 88(11) | 91(23)             |
| <Fmean,0.85>  | 71(10) | 71(12) | 72(11) | 73(12)             |
| <Fmean,0.95>  | 80(13) | 81(12) | 80(11) | 81(11)             |
| <WFmean,0.85> | 73(13) | 73(10) | 75(12) | 78(11)             |
| <WFmean,0.95> | 82(13) | 80(11) | 85(10) | 87(11)             |

## 6 Conclusions and Future Work

The goal of this paper was to develop adaptive search strategies to guide the behaviour of GP for both improving its convergence speed and evolving more effective scheduling heuristics for DFJSS. The goal was achieved by proposing the adaptive mutation and re-initialisation strategies that can utilise the information of the frequency of features information during the evolutionary process.

The results show that with adaptive search, the proposed IM<sup>2</sup>GP can speed up the convergence and achieve better performance in most scenarios while no worse in all other scenarios without increasing the computational cost. The evolved rules by IM<sup>2</sup>GP have better test performance of a given complex job shop scenario, especially in minimising mean-flowtime and weighted mean-flowtime. In terms of the number of unique features, the evolved rules by the proposed algorithms with adaptive strategies contain fewer features. This can potentially improve the interpretability of the evolved rules because the relationships between features tend to be less complicated. Besides, the proposed algorithms that incorporate the frequency of features information do not need extra computational cost. This shows the benefits of making use of the information during the evolutionary process.

Some interesting directions can be further investigated in the near future. This work already shows the effectiveness to take advantage of the information generated during the evolutionary process. We would like to find more promising ways to detect useful information further to improve its performance.

## References

1. Manne, A.S.: On the job-shop scheduling problem. *Operations Research* 8(2), 219–223 (1960)
2. Geiger, C.D., Uzsoy, R., Aytuğ, H.: Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. *Journal of Scheduling* 9(1), 7–34 (2006)
3. Tay, J.C., Ho, N.B.: Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering* 54(3), 453–473 (2008)
4. Nguyen, S.B.S., Zhang, M.: A hybrid discrete particle swarm optimisation method for grid computation scheduling. In: 2014 IEEE Congress on Evolutionary Computation (CEC). pp. 483–490. IEEE (2014)
5. Brucker, P., Schlie, R.: Job-shop scheduling with multi-purpose machines. *Computing* 45(4), 369–375 (1990)
6. Yska, D., Mei, Y., Zhang, M.: Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling. In: European Conference on Genetic Programming. pp. 306–321. Springer (2018)
7. Zhang, F., Mei, Y., Zhang, M.: Genetic programming with multi-tree representation for dynamic flexible job shop scheduling. In: Proceedings of the Australasian Joint Conference on Artificial Intelligence (AI). pp. 472–484. Springer (2018)
8. Durasevic, M., Jakobovic, D.: A survey of dispatching rules for the dynamic unrelated machines environment. *Expert Systems with Applications* 113, 555–569 (2018)

9. Koza, J.R., Poli, R.: Genetic programming. In: *Search Methodologies*, pp. 127–164. Springer (2005)
10. Miyashita, K.: Job-shop scheduling with genetic programming. In: *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*. pp. 505–512. Morgan Kaufmann Publishers Inc. (2000)
11. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Genetic programming for evolving due-date assignment models in job shop environments. *Evolutionary Computation* 22(1), 105–138 (2014)
12. Branke, J., Nguyen, S., Pickardt, C.W., Zhang, M.: Automated design of production scheduling heuristics: A review. *IEEE Transactions on Evolutionary Computation* 20(1), 110–124 (2016)
13. Burke, E.K., Hyde, M.R., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.R.: Exploring hyper-heuristic methodologies with genetic programming. In: *Computational Intelligence*, pp. 177–201. Springer (2009)
14. Burke, E.K., Hyde, M.R., Kendall, G., Woodward, J.R.: A genetic programming hyper-heuristic approach for evolving 2-d strip packing heuristics. *IEEE Trans. Evolutionary Computation* 14(6), 942–958 (2010)
15. Hyde, M.R.: A genetic programming hyper-heuristic approach to automated packing. Ph.D. thesis, University of Nottingham, UK (2010)
16. Bader-El-Den, M.B., Poli, R., Fatima, S.: Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework. *Memetic Computing* 1(3), 205–219 (2009)
17. Pillay, N., Banzhaf, W.: A genetic programming approach to the generation of hyper-heuristics for the uncapacitated examination timetabling problem. In: *Proceedings of the Portuguese Conference on Artificial Intelligence (EPIA)*. pp. 223–234 (2007)
18. Zhang, F., Mei, Y., Zhang, M.: 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 (EvoCOP)*. pp. 33–49. Springer (2019)
19. 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 (GECCO)*. pp. 347–355. IEEE (2019)
20. Durasevic, M., Jakobovic, D.: Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment. *Genetic Programming and Evolvable Machines* 19(1-2), 9–51 (2018)
21. Hildebrandt, T., Heger, J., Scholz-Reiter, B.: Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. In: *Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*. pp. 257–264. ACM (2010)
22. Davis, J.P., Eisenhardt, K.M., Bingham, C.B.: Developing theory through simulation methods. *Academy of Management Review* 32(2), 480–499 (2007)
23. Mei, Y., Zhang, M., Nguyen, S.: Feature selection in evolving job shop dispatching rules with genetic programming. In: *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference (GECCO)*. pp. 365–372 (2016)
24. Gilpin, L.H., Bau, D., Yuan, B.Z., Bajwa, A., Specter, M., Kagal, L.: Explaining explanations: An overview of interpretability of machine learning. In: *5th IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. pp. 80–89 (2018)