A Two-stage Genetic Programming Hyper-heuristic Approach with Feature Selection for Dynamic Flexible Job Shop Scheduling

Fangfang Zhang
School of Engineering
and Computer Science
Victoria University of Wellington
Wellington, New Zealand
fangfang.zhang@ecs.vuw.ac.nz

Yi Mei
School of Engineering
and Computer Science
Victoria University of Wellington
Wellington, New Zealand
yi.mei@ecs.vuw.ac.nz

Mengjie Zhang
School of Engineering
and Computer Science
Victoria University of Wellington
Wellington, New Zealand
mengjie.zhang@ecs.vuw.ac.nz

ABSTRACT

Dynamic flexible job shop scheduling (DFJSS) is an important and a challenging combinatorial optimisation problem. Genetic programming hyper-heuristic (GPHH) has been widely used for automatically evolving the routing and sequencing rules for DFJSS. The terminal set is the key to the success of GPHH. There are a wide range of features in DFJSS that reflect different characteristics of the job shop state. However, the importance of a feature can vary from one scenario to another, and some features may be redundant or irrelevant under the considered scenario. Feature selection is a promising strategy to remove the unimportant features and reduce the search space of GPHH. However, no work has considered feature selection in GPHH for DFJSS so far. In addition, it is necessary to do feature selection for the two terminal sets simultaneously. In this paper, we propose a new two-stage GPHH approach with feature selection for evolving routing and sequencing rules for DFJSS. The experimental studies show that the best solutions achieved by the proposed approach are better than that of the baseline method in most scenarios. Furthermore, the rules evolved by the proposed approach involve a smaller number of unique features, which are easier to interpret.

CCS CONCEPTS

• **Computing methodologies → Planning under uncertainty**;

KEYWORDS

Feature Selection, Dynamic Flexible Job Shop Scheduling, Genetic Programming Hyper-heuristics

ACM Reference Format:
Fangfang Zhang, Yi Mei, and Mengjie Zhang. 2019. A Two-stage Genetic Programming Hyper-heuristic Approach with Feature Selection for Dynamic Flexible Job Shop Scheduling. In *Genetic and Evolutionary Computation Conference (GECCO '19), July 13–17, 2019, Prague, Czech Republic.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3321707.3321790

GECCO '19, July 13–17, 2019, Prague, Czech Republic
© 2019 Association for Computing Machinery.
ACM ISBN 978-1-4503-6111-8/19/07…$15.00
https://doi.org/10.1145/3321707.3321790

1 INTRODUCTION

Job shop scheduling (JSS) [4] is an important combinatorial optimisation problem, which aims to optimise resource allocation. In the job shop floor, we expect to complete the processing of jobs (i.e. each job consists of a sequence of operations) in the shortest time, thus to improve its productivity. In flexible job shop scheduling, the machine resource is more flexible than ordinary JSS in that an operation can be processed on more than one machine. Thus, the machines need to be properly assigned to the operations (*routing*) and all the operations should be processed in an effective order (*sequencing*). In addition, in the real-world, the scheduling process is conducted under dynamic events such as job arrivals over time [29] and machine breakdown [1, 21], which is known as dynamic flexible job shop scheduling (DFJSS).

Over the years, lots of approaches have been proposed for solving the JSS problems. There are three major types of scheduling methods. In the early days, *exact methods* such as dynamic programming [6] and integer linear programming [25] have been investigated. On one hand, exact methods aim at finding optimal solutions. On the other hand, these techniques are too time-consuming when the problems are getting large. In addition, it is hard for exact methods to handle dynamic problems where a lot of real time decisions are needed to be made quickly. Subsequently, *heuristic methods* which are able to find a good but not necessarily optimal solution have been applied. These methods are much quicker and can be used to solve large scale problem in a reasonable time. However, it cannot be used efficiently to handle dynamic problems. *Dispatching rules*, as priority functions, might be the most popularly used heuristic for JSS. It is very simple to implement and handle dynamic problem efficiently. It is noted that in DFJSS, a dispatching rule consists of a routing rule (for machine assignment) and a sequencing rule (for operation sequencing).

However, manually designing dispatching rules is very time-consuming and heavily relies on domain knowledge which is not always available. Genetic programming (GP) [12], as a hyper-heuristic method, has been successfully applied to automatically evolve dispatching rules for JSS [9, 19, 23]. The evolved rules by GP are the combination of terminals and functions. It is noted that the selection of the terminals and functions is critical for GP to succeed. In a given problem, the feature set should be defined carefully so that irrelevant information is not considered. The inclusion of irrelevant information can easily convert a problem that is amenable to efficient algorithmic solutions into one that is intractable. On the other hand, it is important that the feature set is large enough to include all information that is relevant to solve the problem. In DFJSS, there are many characteristics (e.g. machine-related, job-related and system-related) that may be useful. However, it is unknown whether each of them will be actually useful, and some of them may be redundant and irrelevant in the considered scenarios. For example, the weights of jobs are not important in minimising mean-flowtime (i.e. the average time to process a job), but it plays an important role in minimising mean-weighted-flowtime (i.e. the average time to process a job taking its weight into account). Furthermore, two feature sets (one for evolving routing rules and the other for evolving sequencing rules) should be optimised simultaneously. This makes it more challenging for designing proper terminal sets in DFJSS.

The problem can be addressed by using feature selection to choose only informative features based on different scenarios. Feature selection is an important part of data processing which can be used to enhance the quality of the feature space, thus to simplify the learned model, speed up the learning process and improve the performance. It has been successfully used to solve classification [8, 24, 26], clustering [13] and regression [7, 31] problems. Although GP itself can automatically perform feature selection for JSS, its ability is limited. For example, even in the best rule evolved by GP, there are still some irrelevant features. More advanced techniques are needed in this area.

To the best of our knowledge, little is yet known about using feature selection in JSS. Mei et al. proposed a feature ranking approach in [17] and feature selection approach with niching and surrogate techniques in [15] to evolve dispatching rules. The results showed that using only the features selected by the proposed approach can lead to significantly better evolved rules without sacrificing its efficacy. However, the approach was only investigated in the JSS problems and only one feature set was involved. There are some challenges and gaps for feature selection in solving the DFJSS problems as follows.

(1) The simulation is different by nature and surrogate cannot be directly used. That means the feature selection can be very time-consuming or inaccurate.
(2) We need to consider both routing and sequencing rules and do feature selection for two terminal sets rather than one terminal set.

Moreover, there is no work about using feature selection to solve the DFJSS problem. It is important to *consider feature selection as a part of the evolutionary process* to avoid introducing feature bias into the model.

1.1 Goals

This paper aims to develop a two-stage GP approach where the terminals chosen using feature selection are used to improve performance of GP during the evolutionary optimisation. The two-stage framework is expected to help GP find more effective rules. In particular, we have the following research objectives.

(1) Propose a novel two-stage framework for GP with feature selection to evolve routing and sequencing rules simultaneously.
(2) Verify the effectiveness of the proposed algorithm by comparing with genetic programming with cooperative coevolution (named as CCGP) which was proposed for DFJSS in [28].
(3) Analyse the proposed approach in terms of the number and the quality of the selected features.

2 BACKGROUND

In this section, the problem description, the mechanism of dispatching rules, the main processes of GP and the challenge of feature selection for solving the DFJSS problems are illustrated.

2.1 Dynamic Flexible Job Shop Scheduling

Given a set of machines $M = \{M_1, M_2, ..., M_m\}$ and jobs $J = \{J_1, J_2, ..., J_n\}$, flexible job shop scheduling [14] aims to determine which machine to process a particular job and which job will be chosen to process next by a particular machine. To be specific, each job J_j has a sequence of l_j ($l_j <= m$) operations $O_j = (O_{j1}, O_{j2}, ..., O_{jl_j})$. Each operation O_{ij} can only be processed by one of its own optional machines $\pi(O_{ij})$ and its processing time $\delta(O_{ij})$ depends on the machine that processes it. For the dynamic job shop scheduling problem, jobs arrive in the job shop over time and their information can only be known when they arrive. In DFJSS, the machine assignment and operation sequencing tasks are considered at the same time taking the dynamic events into account.

2.2 Dispatching Rules for Dynamic Flexible Job Shop Scheduling

In order to follow the order constraint, only *ready operations* are allowed to be allocated to machines. Naturally, operations become ready when one of two different events occur. The first one is the first operation of a job. The second one is the subsequent operation whose proceeding operation is just finished.

A dispatching rule consists of a routing rule and a sequencing rule in DFJSS. Once a ready operation becomes ready (*routing decision point*), it will be allocated to the machine with the highest priority which is calculated by the routing rule. Once a machine becomes idle (*sequencing decision point*), the sequencing rule will be triggered to select the operation with the highest priority as the next task. In this way, new decisions are only made at decision points based on the current information, which is more efficient and can handle dynamic events well. On one hand, each operation will be allocated to a machine by a routing rule first and then selected by a sequencing rule later. On the other hand, at each decision point, the corresponding routing rule and sequencing rule will be triggered. It means the routing and sequencing processes are conducted in an interactive way.

2.3 Genetic Programming Hyper-heuristics

A hyper-heuristic [3, 5] is a heuristic search method that seeks to select or generate heuristics to efficiently solve hard computational search problems. The unique feature is that hyper-heuristics search in a search space of heuristics. Hyper-heuristic is often incorporated with machine learning technique to achieve its goal. In JSS, a hyper-heuristic method aims at improving the generalisation of evolved rules to enhance the performance. GP, as a hyper-heuristic method, has been successfully applied to evolve dispatching rules for solving

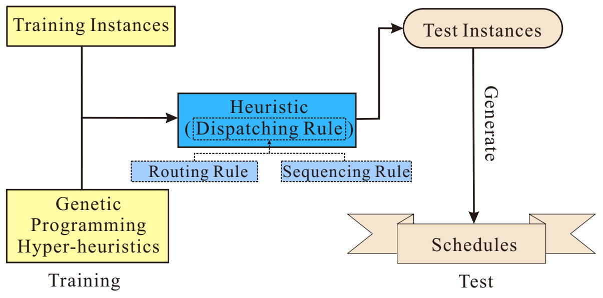

Figure 1: The overall process of genetic programming hyper-heuristics for dynamic flexible job shop scheduling.

JSS problems [2, 10, 11, 18, 19, 30]. There are some advantages of using GP as a hyper-heuristic method. The first one is its flexible representation (i.e. tree structure). It is not necessary to define the structure of priority functions in advance. The second one is that the tree-based programs provide us the opportunities to understand the behaviour of the evolved rules, which is important for real-world applications.

Figure 1 shows the overall processes of GP hyper-heuristics for solving the JSS problems. There are two phases in the whole process, i.e. the training process and the test process. The output of the training process is a heuristic (i.e. a dispatching rule) rather than a solution. Then, in the test process, the dispatching rule serves as the input. Based on the evolved dispatching rules, a schedule is generated and the performance (i.e. objective value) of the dispatching rule can be measured.

2.4 Feature Selection

Feature selection is an effective process for selecting a subset of relevant features by removing irrelevant, redundant and misleading features [27]. It has been successfully used to solve classification [24, 26], clustering [13] and regression [7, 31] problems. In a job shop floor, there are many characteristics that can be used for GP, however, it is not known which are more important in a specific case. On one hand, it is better to include all relevant features of the problem to help GP to evolve rules. On the other hand, the search space grows exponentially as more terminals are included. Feature selection is a good technique for handling this problem faced in DFJSS.

GP has the capability of finding hidden relationships between a subset of features. It can simultaneously identify relevant features and evolve the best tree using the relevant features in an adaptive evolutionary process. However, from our preliminary work, even in this way, GP still contains some irrelevant features in the best individual. Its ability can be further improved by feature selection technique.

However, there are some challenges to apply feature selection in DFJSS. Firstly, the bounds of the objectives are not clear and it is hard to measure the importance of a feature. Secondly, the task in DFJSS (i.e. prioritising operations or machines) is different from the traditional machine learning task (e.g. classification, regression and clustering), which makes traditional feature selection methods not directly applicable.

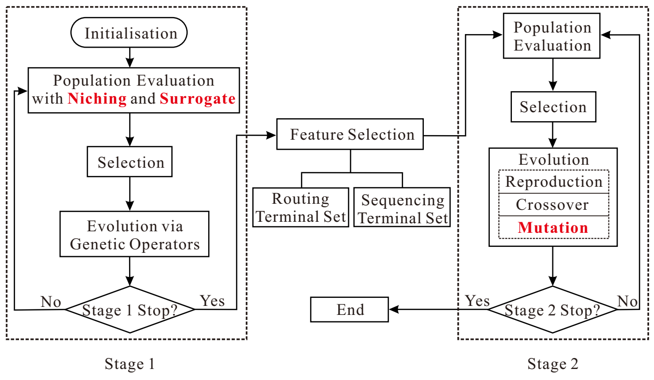

Figure 2: The flowchart of two-stage genetic programming with feature selection.

3 THE NEW APPROACH

In this section, the proposed two-stage framework is illustrated first. Then, the mechanism of feature selection is described. The whole proposed approach is the incorporation of feature selection and the two-stage framework.

3.1 Two-stage Framework

Existing works [15, 17] about feature selection for dynamic JSS were mainly presented in an offline way. It means that feature selection is always applied as a pre-processing step to get a promising subset of terminals first and then the selected terminals are used in another independent GP run to solve the problems. In terms of GP with feature selection, there are some drawbacks of the offline way. Firstly, this will waste some generated good structures of individuals which have been generated in the process of feature selection. Secondly, the evaluation in GP is very time-consuming and GP with feature selection will make this phenomenon even worse. These facts foster the motivation to propose a more effective mechanism for GP with feature selection. In this paper, in order to take advantage of effective structures already generated in the process of feature selection and save evaluations, a two-stage genetic programming approach with feature selection is proposed. The flowchart of the proposed approach, which is named as FSGP, is shown in Figure 2.

The main steps, which are evaluation, selection and evolution, are the same as the classical GP algorithm. The difference is that there is a check point (i.e. generation 50) for separating the whole GP process into two stages. *In the first stage* (i.e. before generation 50), GP proceeds with a niching evaluator and a surrogate model. *The output of stage 1 is a ready population for feature selection.* At generation 50, feature selection mechanism will be invoked to select two sets of informative terminals for evolving routing and sequencing rules, respectively. That means the terminals set both for evolving routing and sequencing rules are reset. *In the second stage* (i.e. after generation 50), the population will be evaluated without niching and surrogate techniques. The mutation operator samples from the selected terminals rather than the entire terminal set while generating the new sub-tree. The main process of the second stage is the same as the classical GP.

In conclusion, the two-stage GP approach contains two consecutive phases. The first stage is mainly for feature selection with niching and surrogate techniques. In the second stage, the obtained information in the first stage is well utilised by inheriting its evolutionary process. In addition, selected features are used to guide the search space during the subsequent evolutionary search.

3.2 Niching Based and Surrogate-assisted Feature Selection

An effective feature selection with niching and surrogate for dynamic JSS was proposed in [15]. In this paper, we apply the idea but with a number of adjustments to dynamic *flexible* JSS.

3.2.1 Niching. For feature selection, the first question is how to measure the importance of a feature. It has two factors. *One is the observed individuals* which are used as the baseline to analyse the features. *The other is the way of measuring the importance of a feature.* Both of them are critical for selecting features.

Clearing [22] (i.e. used to reduce the number of poor individuals in crowded areas within the search space), as a niching technique, is applied to get a *diverse* set of *good* individuals as samples for selecting informative features. Following the suggestions in [15], it is conducted by adding a clearing process with a radius $\sigma = 0$ and a capacity $k = 1$ after traditional fitness evaluation. This aims at investigating more features instead of falling into local optimum. It is worth mentioning that two sets of individuals (i.e. one set from a sub-population) are selected. Then, the feature selection procedure is conducted on the two sets, separately.

The contributions of features for individuals are then calculated to decide which feature will be chosen. The contributions are defined proportional to the importance of features for the fitness values. For example, 20 individuals are selected as observed individuals. There are three main steps for defining the contribution of a feature. Firstly, the measured feature is replaced by one in all 20 individuals. The fitnesses of all new individuals (i.e. the measured feature is replaced by one) are obtained. Secondly, the fitness differences of 20 individual after and before replacing the measured feature to one are calculated as shown in Eq. (1). Note that this paper is seeking to minimise the objective value. If $fitness_d > 0$, it means the fitness becomes worse without the measured feature. That is to say, this measured feature is very important. The larger the $fitness_d$ value, the more important the feature is. Thirdly, the fitnesses of 20 individuals are normalised as shown in Eq. (2) and act as the voting weights as shown in Eq. (3). If $fitness_d > 0.001$, the weight will be assigned to the measured feature as voting score. Finally, a feature is selected if the total weight voting for it is larger than the total weight not voting for it.

$$fitness_d = fit(r|t = 1) - fit(r) \tag{1}$$

$$n(r) = \frac{1}{1 + fit(r)} \tag{2}$$

$$w(r) = max\left\{\frac{n(r) - n_{min}}{n_{max} - n_{min}}, 0\right\} \tag{3}$$

3.2.2 Surrogate. The proposed feature selection is computational expensive. In order to handle this, the half shop surrogate model [20] is applied to reduce the computation time for feature selection. The surrogate model aims to improve the efficiency by simplifying the problem model. In this paper, in the surrogate model, the number of machines is five (i.e. in test process, the number of machines is ten) and the number of jobs is 500. In this case, the computational complexity of the feature selection is greatly reduced. Specifically, a generation of GP during feature selection takes about 10% of the time of a generation of the standard GP.

Table 1: The terminal set.

Notation	Description
NIQ	The number of operations in the queue
WIQ	Current work in the queue
MWT	Waiting time of a machine
PT	Processing time of an operation on a specified machine
NPT	Median processing time for the next operation
OWT	The waiting time of an operation
WKR	Median amount of work remaining for a job
NOR	The number of operations remaining for a job
W	Weight of a job
TIS	Time in system

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
Crossover/Mutation/Reproduction rate	80% / 15% / 5%
Parent selection	Tournament selection with size 7
Number of generations in FSGP	100
Number of generations in CCGP	100
Number of generations in the first stage	50
Terminal/non-terminal selection rate	10% / 90%

4 EXPERIMENTAL STUDIES

In this section, the parameter setting for GP and the used simulation model will be described first. Then, the test performance of proposed approach is verified followed by the analysis of selected features.

4.1 Parameter Setting of Genetic Programming

In our experiment, the terminal set of GP is shown in Table 1 [16]. The features indicate the characteristics related to jobs, machines and system. The function set is {+, −, ∗, /, *max*, *min*}, following the setting in [16]. The arithmetic operators take two arguments. The "/" operator is protected division, returning one if divided by zero. The *max* and *min* functions take two arguments and return the maximum and minimum of their arguments, respectively. The other parameter settings of GP are shown in Table 2.

4.2 Simulation Configuration

Simulation has been broadly used to investigate complex problems. The simulated environment is used as an experimental model to study factors affecting DFJSS. *Problem instance* for an instantiation of the problem scenario with a particular pseudo-random number generator seed is actually a simulation model.

In this paper, there are 5000 jobs need to be processed by ten machines. For DFJSS simulation, new jobs will come over time

Table 3: The best and mean(standard deviation) of the objective value of FSGP and CCGP over 50 independent runs for six dynamic scenarios.

Scenario	FSGP		CCGP	
	Best	Mean(sd)	Best	Mean(sd)
<Tmax,0.85>	1158.38	**1217.15(49.07)**	1156.91	1222.84(62.91))
<Tmax,0.95>	**1819.23**	1906.03(84.44)	1829.64	1893.68(66.40)
<Tmean,0.85>	**383.16**	386.69(3.84)	383.34	385.91(2.83)
<Tmean,0.95>	**545.72**	**552.33(6.42)(-)**	546.40	554.73(6.91)
<WTmean,0.85>	**824.98**	831.77(9.86)	825.57	829.45(6.03)
<WTmean,0.95>	**1094.67**	1108.84(15.82)	1094.97	1106.86(11.54)

according to a Poisson process with rate λ. Each job with different weights has several operations (i.e. follow a uniform discrete distribution between one and ten). The job weight indicates the importance of a job and 20%, 60% and 20% jobs will get weight 1, 2 and 4, respectively. The processing time of each operation is assigned 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.

It is noted that *utilisation* is the proportion of time that a machine is busy. The expression is shown in Eq. (4). In Eq. (4), μ is the average processing time of the machines. P_M is the probability of a job visiting a machine. For example, P_M is 2/10 if each job has two operations.

$$p = \lambda * \mu * P_M \tag{4}$$

In addition, in order to make sure the accuracy of collected data, warm up jobs (i.e. 1000) are used to get a steady state of the job shop floor and we collect data from the next 5000 jobs. The simulation stops when the 6000th jobs is finished.

4.3 Test Performance

We use the term *scenario* to describe the combination of properties defining the shop. In order to verify the *effectiveness* and *robustness* of proposed approach, it will be tested in six different scenarios. The scenario consists of three different objective (i.e. max-flowtime, mean-flowtime and mean-weighted flowtime) and two utilisation levels (i.e. 0.85 and 0.95). For the convenience of description, Tmax, Tmean and WTmean are used to indicate max-flowtime, mean-flowtime and mean-weighted flowtime, respectively. Wilcoxon signed rank test with a significance level of 0.05 is used to verify the performance of proposed approach. In the following results, "-/+" indicate the corresponding result is significantly better or worse than the counterpart.

Table 3 shows the best and mean (standard deviation) of the test objective value of FSGP and CCGP in the six different scenarios. In most of the scenarios, the best solutions achieved by FSGP are better than that of CCGP. In <Tmax,0.85>, the mean and standard deviation obtained by FSGP are slightly smaller than that of CCGP. In addition, in <Tmean,0.95>, the performance of FSGP is significantly better than its counterpart. In the remaining scenarios, FSGP and CCGP perform at a similar level. In order to show the generalisation of FSGP, Figure 3 shows the training fitness versus average normalised objective value on the test instances. The scatter plot

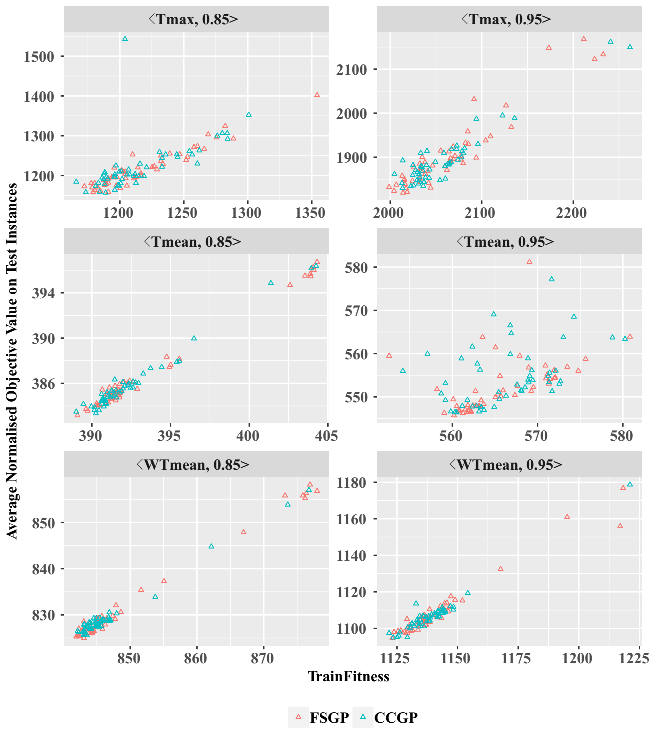

Figure 3: The training fitness versus test objective scatter plot based on the 50 final results of FSGP and CCGP.

contains the 50 final results of FSGP and CCGP. Overall, the generalisation of FSGP is promising in all the examined scenarios, as the test performance is very consistent with the training performance.

4.4 Unique Feature Analysis

The number of unique features indicates how many different features are used in building the rules. Intuitively, the smaller the number, the easier to interpret the evolved rules. We expect to achieve rules with a smaller number of unique features.

Table 4: The mean and standard deviation of the number of unique features in sequencing and routing rules of FSGP and CCGP over 50 independent runs for six dynamic scenarios.

Scenario	Sequencing Rule		Routing Rule	
	FSGP	CCGP	FSGP	CCGP
<Tmax,0.85>	6.38(1.52)(-)	7.14(1.50)	8.44(1.33)	8.44(1.26)
<Tmax,0.95>	6.66(1.42)(-)	7.42(1.25)	8.34(1.38)	8.48(1.25)
<Tmean,0.85>	5.16(1.48)(-)	6.86(1.70)	8.02(1.29)	8.44(1.26)
<Tmean,0.95>	6.32(1.35)(-)	6.96(1.68)	7.80(1.36)(-)	8.28(1.26)
<WTmean,0.85>	5.36(1.38)(-)	6.28(1.58)	8.12(1.30)	8.14(1.25)
<WTmean,0.95>	5.58(1.33)(-)	6.64(1.47)	8.02(1.08)	8.18(1.35)

Table 4 shows the number of unique features of the sequencing and routing rules that evolved by FSGP and CCGP in the six scenarios. It is obvious that the number of unique features of the sequencing rules is reduced a lot more than that of the routing rules by FSGP. To be specific, the number of unique features of the

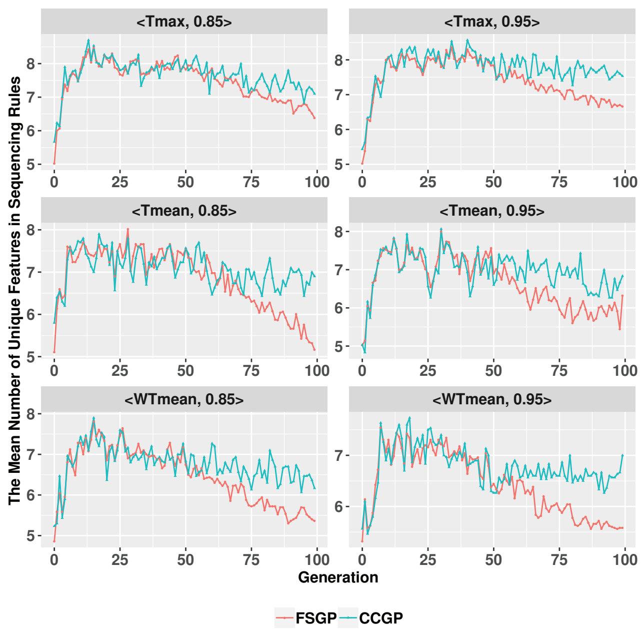

Figure 4: The mean number of unique features in *sequencing rules* evolved by FSGP in six scenarios.

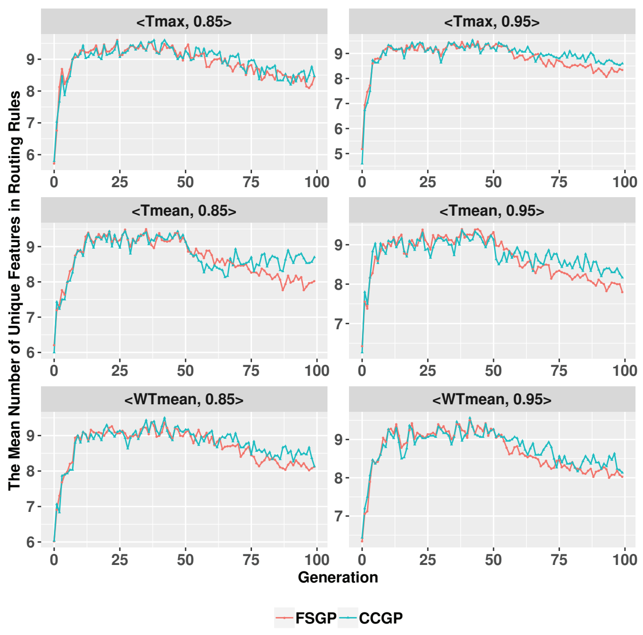

Figure 5: The mean number of unique features in *routing rules* evolved by FSGP in six scenarios.

sequencing rules evolved by FSGP is significantly smaller than that of CCGP. In addition, the number of unique features in the routing rules evolved by FSGP is significantly smaller than its counterpart in scenario <Tmean,0.95>. In the remaining scenarios, they do not have significant difference.

Now we further look at the convergence curve of the number of unique features of sequencing and routing rules in different scenarios as shown in Figures 4 and 5. We can see that the number of unique features of the sequencing and routing rules obtained by FSGP and CCGP in the first stage (i.e. before generation 50) are almost the same in all scenarios. It is noted that the evolutionary processes of FSGP and CCGP are exactly the same but simplified rules, for example, WIQ/WIQ is simplified as 1, are recorded in CCGP. This means more features are counted in FSGP and it is reasonable for the comparison here. In the second stage (i.e. after generation 50), the number of unique features becomes smaller in FSGP. This suggests that the selected features guide the evolutionary process via mutation to get more interpretable rules without sacrificing its performance.

4.5 Feature Analysis

Figure 6 shows the selected features in the sequencing rules evolved by FSGP over 50 independent runs in six scenarios. It shows that PT is the most important feature for optimising all the considered objectives. There are also some other promising feature such as TIS and NIQ in scenario <Tmax,0.85> and <Tmax,0.95>. In scenario <Tmean,0.85> and <Tmean,0.95>, PT and WKR are the most important features as they appear in each run. In addition, in scenario <WTmean,0.85> and <WTmean,0.95>, in addition to PT and WRK, W is one of the most important features. It is consistent with our expectation because weight is an important indicator for minimising weighted-flowtime.

Figure 7 shows the selected features in the routing rules obtained by FSGP over 50 independent runs in different scenarios. Comparing with selected features in sequencing rules, more features are selected, especially in scenario <Tmax,0.85> and <Tmax,0.95>. This may be because the routing decision needs to take more advantage of the information provided by different kinds of features. In general, MWT, OWT and WIQ are the top three features. This is also consistent with our intuition, as a less busy machine which becomes idle sooner should be more preferred by routing. It is interesting that W (i.e. weight) is not as important for routing rules as it is for sequencing rules.

4.6 Rule Analysis

In order to further understand the behaviour of the evolved rules by FSGP and CCGP, in this section, we analyse the sequencing and routing rules evolved by FSGP and CCGP. We take scenario <Tmean,0.95> as an example, and the rules evolved for other scenarios show a similar pattern.

Figure 8 and Figure 9 show one of the best sequencing rules evolved by FSGP and CCGP, respectively. It is very interesting that the main structure of these two rules are very similar as marked in grey colour. They are only different in some subtrees. Figure 10 shows the routing rules collaborated with the sequencing rule shown in Figure 8. The size of the routing rule (i.e. the number of nodes) evolved by CCGP, which is not shown here due to space limitations, is much smaller than its counterpart.

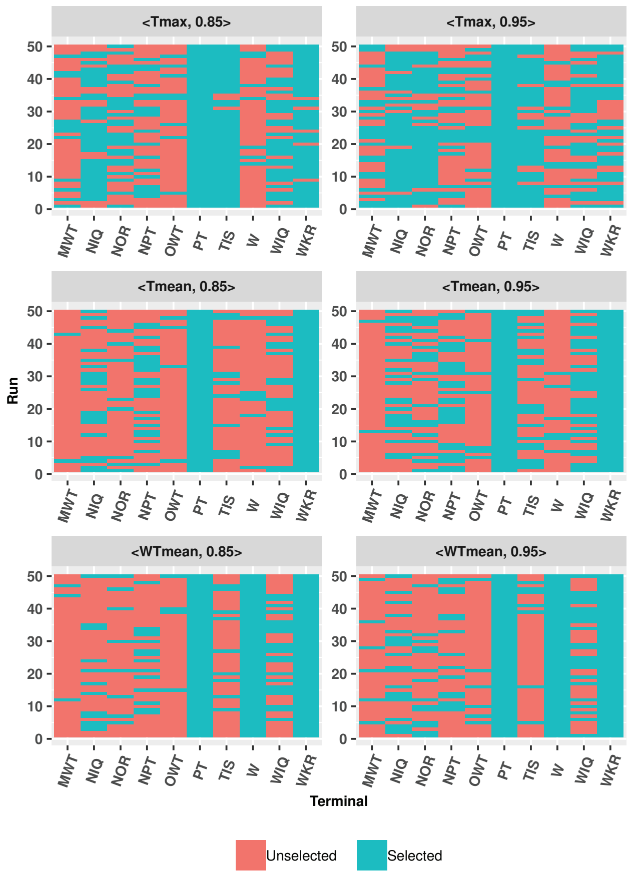

Figure 6: Selected features in *sequencing rules* of FSGP in different scenarios.

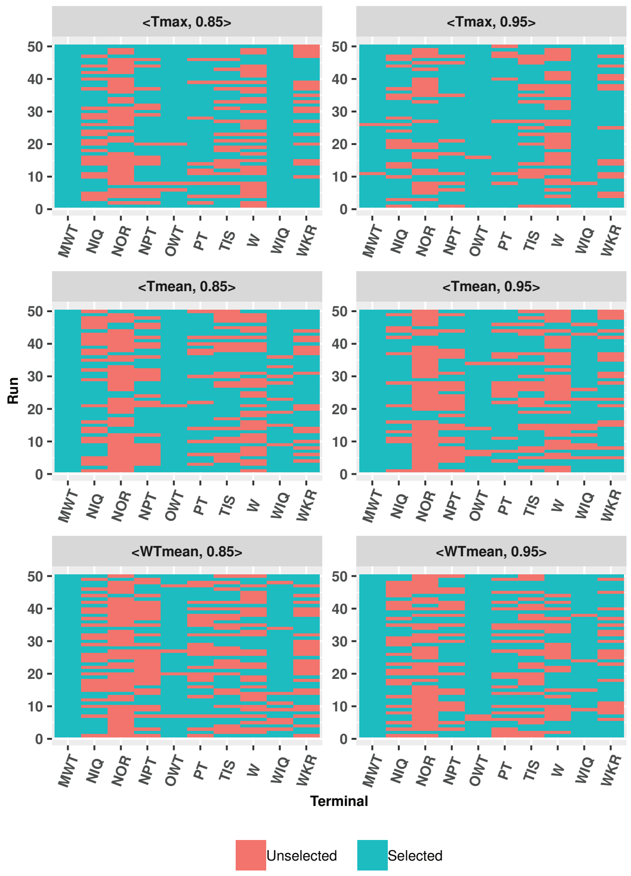

Figure 7: Selected features in *routing rules* of FSGP in different scenarios.

In our experiment, the smaller the priority value, the higher the priority. The routing rule evolved by FSGP can be regarded as three parts as marked in grey colour in Figure 10. The root is min function. The routing rule can be described as follows.

rule = $min\{part_1, part_2, part_3\}$

where,

$$\begin{aligned} part_1 =& W + WKR * PT \\ &+ \frac{PT + WKR}{(WKR - TIS) * WIQ} - min\{WKR, PT^2\} \\ &+ PT^2 - PT - TIS - 2 * NIQ \end{aligned} \tag{5}$$

$$part_2 = PT + max\{WIQ, PT\} + min\{WKR, PT^3\} \tag{6}$$

$$part_3 = PT^2 - PT - max\{WIQ, PT\}\} \tag{7}$$

In these three parts, only the part with smallest value plays an important role. In the routing rule, all the features are positive numbers. In the first part, TIS (i.e. time in system) is normally larger than WKR (i.e. median amount of work remaining for a job), so $(PT+WKR)/[(WKR-TIS)*WIQ]$ is often a negative number. In addition, there are many subtraction operators, thus the value tends to be a negative number. The second part is definitely a positive value and cannot become the smallest one. In the third part, the square of PT is much larger than PT and WIQ, it tends to be a positive value. In conclusion, $part_1$ plays a significant role in the routing rule. It can be further described as follows.

$$\begin{aligned} part_1 =& W + WKR * PT \\ &- \frac{PT + WKR}{(TIS - WKR) * WIQ} - min\{WKR, PT^2\} \\ &+ PT^2 - PT - TIS - 2 * NIQ \end{aligned} \tag{8}$$

It shows that the routing rule normally prefers machines with lower WIQ (i.e. work in queue) if an operation has arrived in the job shop for a long time. In addition, from the Eq. (5), one can see that if an operation has not stayed in the job shop system for a long time (i.e. $TIS < WKR$), the routing rule prefers to assign it to a machine with higher workload, to leave space to other more urgent jobs. This is consistent with our intuition.

5 CONCLUSIONS

In this paper, we designed a novel two-stage genetic programming hyper-heuristic with feature selection for solving the dynamic flexible job shop scheduling problem. The proposed method is based

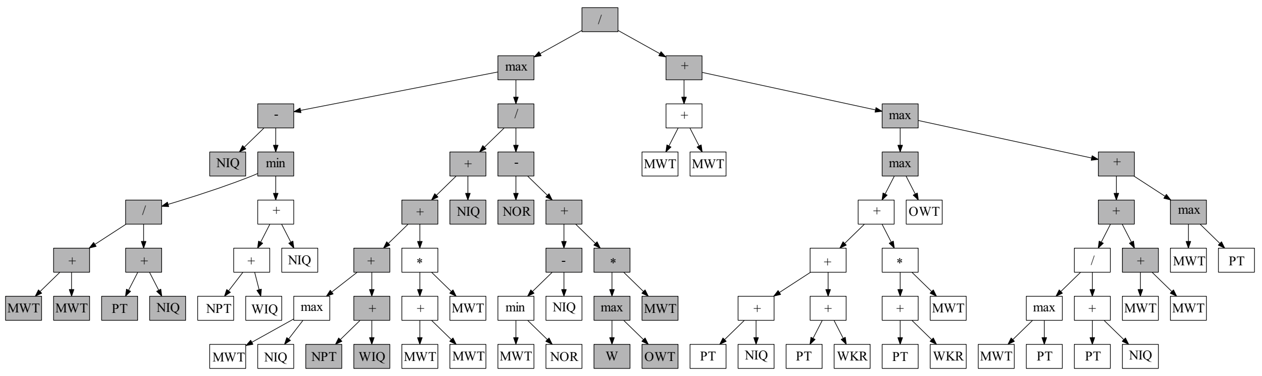
/
max +
- / + max
NIQ min + - MWT MWT max +
/ + NIQ + NIQ NOR + + OWT + max
+ + + NIQ + * - * + * / + MWT PT
MWT MWT PT NIQ NPT WIQ max + + MWT min NIQ max MWT + + + MWT max + MWT MWT
MWT NIQ NPT WIQ MWT MWT MWT NOR W OWT PT NIQ PT WKR PT WKR MWT PT PT NIQ

Figure 8: One of the best sequencing rules evolved by FSGP in scenario <Tmean,0.95>.

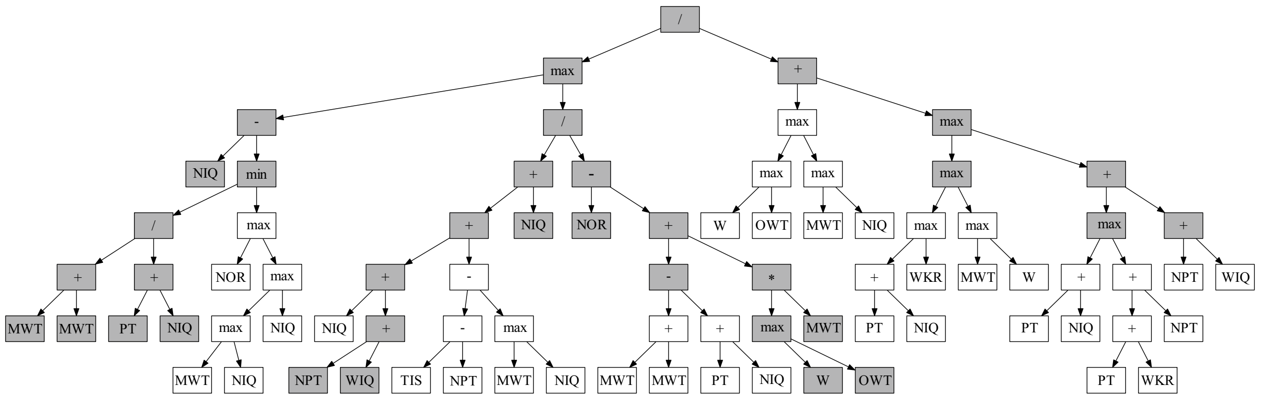
/
max +
- / max max
NIQ min + - max max max +
/ max + NIQ NOR + W OWT MWT NIQ max max max +
+ + NOR max + - - * + WKR MWT W + + NPT WIQ
MWT MWT PT NIQ max NIQ NIQ + - max + + max MWT PT NIQ PT NIQ + NPT
MWT NIQ NPT WIQ TIS NPT MWT NIQ MWT MWT PT NIQ W OWT PT WKR

Figure 9: One of the best sequencing rules evolved by CCGP in scenario <Tmean,0.95>.

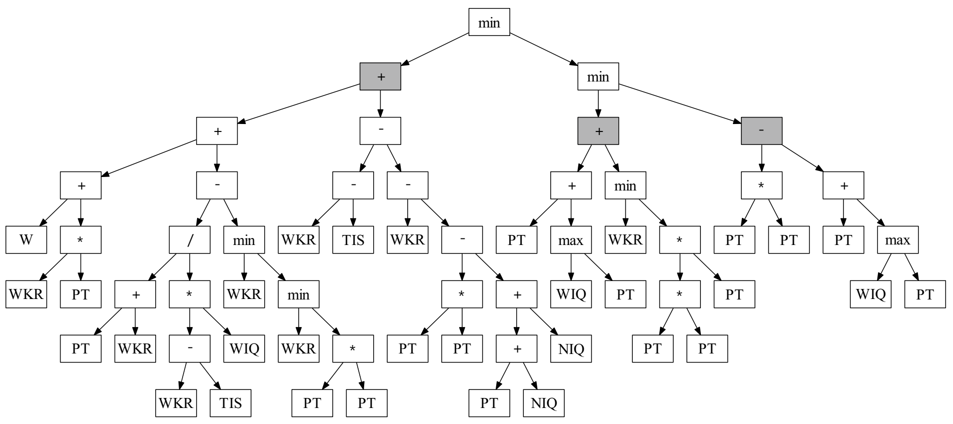
min
+ min
+ - + -
+ - - - + min * +
W * / min WKR TIS WKR - PT max WKR * PT PT PT max
WKR PT + * WKR min * + WIQ PT * PT WIQ PT
PT WKR - WIQ WKR * PT PT + NIQ PT PT
WKR TIS PT PT PT NIQ

Figure 10: One of the best routing rules evolved by FSGP in scenario <Tmean,0.95>.

on the idea that promising features can benefit more for the evolutionary process and can be used to guide the process effectively. In addition, the proposed two-stage learning process was designed to make full use of the information of the evolutionary process. The experimental results show that the proposed approach can achieve better solutions in most scenarios than CCGP in terms of the best solution they can get. Moreover, the proposed method can identify important features accurately and its evolved rules consist of a smaller number of unique features. This is due to the effective guidance of selected features in evolutionary process of the genetic programming hyper-heuristic approach.

In the future, more potential useful features will be designed for solving the DFJSS problem. In addition, more effective measurements of the importance of the features will be investigated.

REFERENCES

[1] Ehsan Ahmadi, Mostafa Zandieh, Mojtaba Farrokh, and Seyed Mohammad Emami. 2016. A multi objective optimization approach for flexible job shop scheduling problem under random machine breakdown by evolutionary algorithms. *Computers & OR* 73 (2016), 56–66.

[2] Jürgen Branke, Torsten Hildebrandt, and Bernd Scholz-Reiter. 2015. Hyper-heuristic Evolution of dispatching rules: a comparison of rule representations. *Evolutionary Computation* 23, 2 (2015), 249–277.

[3] Juergen Branke, Su Nguyen, Christoph W Pickardt, and Mengjie Zhang. 2016. Automated design of production scheduling heuristics: A review. *IEEE Transactions on Evolutionary Computation* 20, 1 (2016), 110–124.

[4] Carlos A. Brizuela and Nobuo Sannomiya. 1999. A diversity study in genetic algorithms for job shop scheduling problems. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 1999), 13-17 July 1999, Orlando, Florida, USA*. 75–82.

[5] Edmund K Burke, Mathew R Hyde, Graham Kendall, Gabriela Ochoa, Ender Ozcan, and John R Woodward. 2009. Exploring hyper-heuristic methodologies with genetic programming. In *Computational Intelligence*. Springer, 177–201.

[6] Haoxun Chen, Chengbin Chu, and J-M Proth. 1998. An improvement of the Lagrangean relaxation approach for job shop scheduling: a dynamic programming method. *IEEE Transactions on Robotics and Automation* 14, 5 (1998), 786–795.

[7] Qi Chen, Mengjie Zhang, and Bing Xue. 2017. Feature selection to improve generalization of genetic programming for high-dimensional symbolic regression. *IEEE Transactions on Evolutionary Computation* 21, 5 (2017), 792–806.

[8] Manoranjan Dash and Huan Liu. 1997. Feature selection for classification. *Intelligent data analysis* 1, 3 (1997), 131–156.

[9] Torsten Hildebrandt, Jens Heger, and Bernd Scholz-Reiter. 2010. 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*. ACM, 257–264.

[10] Rachel Hunt, Mark Johnston, and Mengjie Zhang. 2014. Evolving "less-myopic" scheduling rules for dynamic job shop scheduling with genetic programming. In *Genetic and Evolutionary Computation Conference, GECCO '14, Vancouver, BC, Canada, July 12-16, 2014*. 927–934.

[11] Rachel Hunt, Mark Johnston, and Mengjie Zhang. 2014. Evolving machine-specific dispatching rules for a two-machine job shop using genetic programming. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014*. 618–625.

[12] John R Koza. 1994. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing* 4, 2 (1994), 87–112.

[13] Andrew Lensen, Bing Xue, and Mengjie Zhang. 2016. Particle swarm optimisation representations for simultaneous clustering and feature selection. In *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*. IEEE, 1–8.

[14] Bart L Maccarthy and Jiyin Liu. 1993. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *The International Journal of Production Research* 31, 1 (1993), 59–79.

[15] Yi Mei, Su Nguyen, Bing Xue, and Mengjie Zhang. 2017. An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. *IEEE Transactions on Emerging Topics in Computational Intelligence* 1, 5 (2017), 339–353.

[16] Yi Mei, Su Nguyen, and Mengjie Zhang. 2017. Evolving time-invariant dispatching rules in job shop scheduling with genetic programming. In *European Conference on Genetic Programming*. Springer, 147–163.

[17] Yi Mei, Mengjie Zhang, and Su Nyugen. 2016. Feature selection in evolving job shop dispatching rules with genetic programming. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*. ACM, 365–372.

[18] 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.

[19] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. 2013. A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. *IEEE Transactions on Evolutionary Computation* 17, 5 (2013), 621–639.

[20] Su Nguyen, Mengjie Zhang, and Kay Chen Tan. 2017. Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules. *IEEE Transactions on Cybernetics* 47, 9 (2017), 2951–2965.

[21] John Park, Yi Mei, Su Nguyen, Gang Chen, and Mengjie Zhang. 2018. Investigating a machine breakdown genetic programming approach for dynamic job shop scheduling. In *Genetic Programming - 21st European Conference, EuroGP 2018, Parma, Italy, April 4-6, 2018, Proceedings*. 253–270.

[22] Alain Pétrowski. 1996. A Clearing Procedure as a Niching Method for Genetic Algorithms. In *Proceedings of 1996 IEEE International Conference on Evolutionary Computation, Nayoya University, Japan, May 20-22, 1996*. 798–803.

[23] Christoph W Pickardt, Torsten Hildebrandt, Jürgen Branke, Jens Heger, and Bernd Scholz-Reiter. 2013. Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. *International Journal of Production Economics* 145, 1 (2013), 67–77.

[24] Alper Kursat Uysal. 2016. An improved global feature selection scheme for text classification. *Expert systems with Applications* 43 (2016), 82–92.

[25] Harvey M Wagner. 1959. An integer linear-programming model for machine scheduling. *Naval Research Logistics Quarterly* 6, 2 (1959), 131–140.

[26] Bing Xue, Mengjie Zhang, and Will N Browne. 2013. Particle swarm optimization for feature selection in classification: A multi-objective approach. *IEEE Transactions on Cybernetics* 43, 6 (2013), 1656–1671.

[27] Bing Xue, Mengjie Zhang, Will N Browne, and Xin Yao. 2016. A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on Evolutionary Computation* 20, 4 (2016), 606–626.

[28] Daniel Yska, Yi Mei, and Mengjie Zhang. 2018. Genetic Programming Hyper-Heuristic with Cooperative Coevolution for Dynamic Flexible Job Shop Scheduling. In *European Conference on Genetic Programming*. Springer, 306–321.

[29] Fangfang Zhang, Yi Mei, and Mengjie Zhang. 2018. Genetic programming with multi-tree representation for dynamic flexible job shop scheduling. In *AI 2018: Advances in Artificial Intelligence - 31st Australasian Joint Conference, Wellington, New Zealand, December 11-14, 2018, Proceedings*. 472–484.

[30] Fangfang Zhang, Yi Mei, and Mengjie Zhang. 2018. Surrogate-assisted genetic programming for dynamic flexible job shop scheduling. In *AI 2018: Advances in Artificial Intelligence - 31st Australasian Joint Conference, Wellington, New Zealand, December 11-14, 2018, Proceedings*. 766–772.

[31] Xiaofeng Zhu, Heung-Il Suk, Li Wang, Seong-Whan Lee, Dinggang Shen, Alzheimer's Disease Neuroimaging Initiative, et al. 2017. A novel relational regularization feature selection method for joint regression and classification in AD diagnosis. *Medical image analysis* 38 (2017), 205–214.