A New Representation in Genetic Programming for Evolving Dispatching Rules for Dynamic Flexible Job Shop Scheduling

Fangfang Zhang, Yi Mei, and Mengjie Zhang

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

Abstract. Dynamic flexible job shop scheduling (DFJSS) is a very important problem with a wide range of real-world applications such as cloud computing and manufacturing. In DFJSS, it is critical to make two kinds of real-time decisions (i.e. the routing decision that assigns machine to each job and the sequencing decision that prioritises the jobs in a machine's queue) effectively in the dynamic environment with unpredicted events such as new job arrivals and machine breakdowns. Dispatching rule is an ideal technique for this purpose. In DFJSS, one has to design a routing rule and a sequencing rule for making the two kinds of decisions. Manually designing these rules is time consuming and requires human expertise which is not always available. Genetic programming (GP) has been applied to automatically evolve more effective rules than the manually designed ones. In GP for DFJSS, different features in the terminal set have different contributions to the decision making. However, the current GP approaches cannot perfectly find proper combinations between the features in accordance with their contributions. In this paper, we propose a new representation for GP that better considers the different contributions of different features and combines them in a sophisticated way, thus to evolve more effective rules. The results show that the proposed GP approach can achieve significantly better performance than the baseline GP in a range of job shop scenarios.

Keywords: Representation · Dispatching rules · Dynamic flexible job shop scheduling · Genetic programming.

1 Introduction

Job shop scheduling (JSS), as an important optimisation problem, has received a great deal of attention from both academics and industry researchers. It captures practical and challenging issues in real world scheduling tasks such as managing grid/cloud computing [1] and designing manufacturing processes [2]. JSS aims to make a schedule to process a number of jobs with a set of machines. Each job consists of a sequence of operations which need to be processed one by one. Classical JSS assumes that one operation can be processed on only one specific machine. Thus, the task is to schedule the operations in the queue of the machines.

Flexible job shop scheduling (FJSS) is different from the classic JSS in that each operation can be processed by multiple candidate machines. Therefore, FJSS includes two sub-tasks, which are machine assignment and operation sequencing. Machine assignment is to select an appropriate machine for each operation from its candidate machines. Operation sequencing is to determine the order of processing the allocated jobs in each machine to obtain feasible and satisfactory solutions. FJSS is NP-hard [3].

In practice, the JSS problems are typically in *dynamic* environment. For instance, the jobs arrive over time and their attributes are not completely known until they arrive in the shop floor. There are also other types of dynamic events in JSS problem such as order cancellations [4] and machine breakdowns [5]. In this paper, we focus on dynamic new job arrivals because it is the most frequent and common factor in the shop floor. The challenge of dynamic flexible job shop scheduling (DFJSS) is how to capture both the machine assignment (*routing*) decision and operation sequencing (*sequencing*) decision simultaneously along with the new jobs arriving over time.

Exact approaches to search for optimal solutions, such as dynamic programming [6] and branch-and-bound [7], are too time-consuming and inapplicable for solving large scale JSS problems. Therefore, heuristic search methods such as tabu search [8] and genetic algorithm [9] have been commonly adopted to find "near-optimal" solutions in a reasonable time. However, heuristic search methods are not suitable for solving DFJSS problems because of their lack of ability to react to the dynamic events in real time. Dispatching rules are promising in this case because of their low time complexity, the ease of implementation and the ability to cope with dynamic situations in the job shop. Since the term *dispatching rule* has been used in different contexts, it is worth highlighting that the concept of a dispatching rule in DFJSS consists of a routing rule and a sequencing rule (i.e. two kinds of rules). The dispatching rules are normally designed manually, which is very time-consuming and requires human expertise which is not always available. In addition, many manually designed dispatching rules are relatively simple and normally restricted to some specific assumptions [10] and have difficulties in handling complex practical scenarios [11, 12]. Genetic programming (GP) has been proven to be a dominating method to automatically design dispatching rules for JSS [11, 13, 12, 14]. In order to evolve both the routing rule and sequencing rule for DFJSS, GP was hybridised with the cooperative co-evolution framework to co-evolve the two kinds of rules [15, 16].

In GP for DFJSS, different features in the terminal set have different contributions to the decision making. For example, the feature named WIQ (work in queue, i.e. the total processing time of operations in the queue of a machine) is known to be a dominating feature for making the routing decision, as intuitively a machine with lighter workload should be preferred [17]. On the other hand, machine ready time is another important feature for routing decisions, i.e., it tends to be better to assign the operation to a machine that can become idle in the earliest time. However, the contribution of machine ready time should be smaller than WIQ. Intuitively, a machine with a lighter workload but a later ready time should still be better than a machine with a heavier workload but an earlier ready time. However, the existing GP approaches fail to properly combine the features in accordance with their contributions. As a result, the evolved routing rules usually focus too much on WIQ, but overlook the other non-dominating features. The schedules obtained may become ineffective over the longer term when facing the real-world shop environments.

In this paper, we aim to develop a new representation for routing rule to help GP evolve more effective rules for DFJSS. In particular, we consider the following research objectives:

– Develop a novel component that can appropriately take the information of workload of machines into consideration.
– Propose a novel representation that tends to take into account the effect of the dominating feature WIQ and other non-dominating features properly.
– Verify the effectiveness of proposed GP approach with new representation by comparing its performance with the baseline GP.
– Analyse the rules evolved by the proposed GP approach.

2 Background

2.1 Dynamic Flexible Job Shop Scheduling

Given a set of machines $M = \{M_1, M_2, ..., M_m\}$ and jobs $J = \{J_1, J_2, ..., J_n\}$, FJSS 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. Then FJSS is to find an effective schedule subject to the following constraints:

1) The $(j+1)th$ operation of J_i (denotes by $O_{i(j+1)}$) can only be processed after its preceding operation O_{ij} has been processed.

2) Each operation O_{ij} can be processed on one of the corresponding set of machines $\pi(O_{ij}) \subseteq M$ with $\delta(O_{ij})$.

3) Each machine can process at most one operation at a time.

4) The scheduling is non-preemptive, i.e. the processing of an operation cannot be stopped or paused until it is completed.

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.

2.2 Dispatching Rules in Dynamic Flexible Job Shop Scheduling

In DFJSS, a *routing decision situation* will be generated when a new job arrivals or an operation is finished and its subsequent operation becomes a ready operation. A *sequencing decision situation* will be derived when a machine becomes idle and its queue is not empty. Two kinds of dispatching rules are needed in DFJSS, which are routing rule and sequencing rule, respectively. The quality of a schedule depends highly on how well the routing rule and the sequencing rule work together. The routing rule will be triggered to decide which machine to allocate the operation when a routing decision situation is derived. The sequencing rule will be triggered to determine which operation in its queue will be chosen to process next when a sequencing decision situation is derived. Once the trigger conditions are met, the corresponding decisions will be made immediately.

The machine or operation with the highest priority assigned by routing or sequencing rule is identified respectively. Once one operation is finished, its information related to objectives will be recorded to its corresponding job. After all the operations are processed, the recorded information related to all jobs is obtained. Finally, the fitness can be calculated based on the information according to different objectives.

2.3 Related Work

In recent years, GP has been widely used to automatically design dispatching rules for solving JSS problems [18, 19]. Tree-based GP is commonly used in many studies [20, 21]. In 2007, Tay and Ho [2] proposed a GP approach to evolve priority rules for FJSS with multiple objectives by combining them into a single function. Hildebrandt et al. [11] then used GP to evolve dispatching rules in different simulations for the single objective of meanflow time. The results show that the evolved rules perform very well in different scenarios. However, the early studies only aim to evolve sequencing rule by fixing the routing rule. Cooperation co-evolution was applied to GP to evolve the rules at the same time in [15]. The results show that the evolved rules are more effective.

However, in the standard GP, all the features are considered equally in the terminal set. The WIQ feature is a dominating factor [17, 22] that is much more important than other features for routing decisions, therefore, the routing rules tend to select the machine with minimal WIQ. In this case, WIQ tends to be dominant and overweights the other non-dominating features. However, there are many other features in the job shop and they might be less important than WIQ, but still contribute to the routing decisions. Using them improperly could lead to suboptimal performance.

3 The Proposed GP Approach

In order to identify effective combinations of the features with different importance, this paper considers to separate WIQ from other features, thus to expect GP can focus more on non-dominating features during the evolutionary process.

A new representation for routing rule is designed to learn dispatching rules which can take more information of non-dominating features. It is noted that the representation design only applies on routing rule. In this section, the proposed representation is presented first, followed by the designed components.

3.1 Representation

The routing rule is defined as the product of two parts, which are named as component 1 and component 2 (i.e. the details will be given in section 3.2). Component 1 is *predefined* and component 2 is *evolved* by GP approach automatically. The motivation of this design is to separate the WIQ with non-dominating features. It is worth mentioning that a machine with smallest priority value (i.e. highest priority) will be chosen in this paper. Multiplication is used here to combine these two components together as it is a more appropriate combination operator than addition and subtraction. Specifically, the value of component 2 might be much larger than the value of component 1. An example of the routing rule is shown in Fig. 1. The terminals and functions in Fig. 1 will be described in section 4.2.

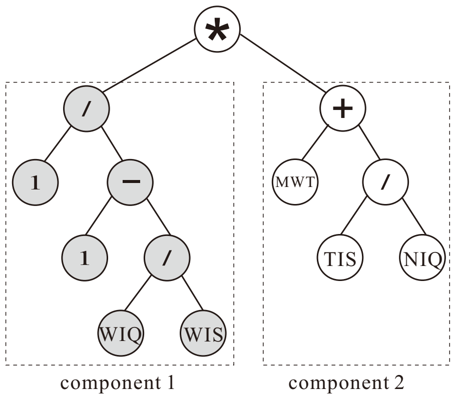

Fig. 1: An example of routing rule with new representation.

3.2 Components Design

The goal of component 1 is to extract the information of the current state solely related to the workload of machines. To this end, component 1 is designed not only to consider the information of WIQ, but also to consider the workload distribution of machines in the system. The workload ratio (i.e. workload of one machine over the entire workload in the system) that can help measure the distribution of allocated jobs of machines. This is to prevent a solution from assigning too much work on a single machine. Thus, the design of component 1 needs to meet two requirements. Firstly, it can be used to indicate the information of workload of individual machines properly. Secondly, it should express the difference of the workload of different machines in the system appropriately.

Under the above consideration, the expression of the proposed component 1 is shown in Eq. (1), where WIQ stands for the total processing time of operations in the queue of a machine, and WIS (work in system. i.e. the total processing time of operations in the queue of all machines) denotes the sum of the workload of all the machines in the job shop. The $component_2$ is the value (V_{GP}) obtained by GP method.

$$component_1 = 1/(1 - WIQ/WIS) \tag{1}$$

$$component_2 = V_{GP} \tag{2}$$

$$Priority = component_1 * component_2 \tag{3}$$

As shown in Eq. (1), component 1 has the following two characteristics. Firstly, it can ensure that all the values obtained by Eq. (1) are larger than 1. It means that component 1 will enlarge the value obtained by the GP approach. In other words, component 1 plays a role of a penalty coefficient. Secondly, the penalty of each machine is different. The heavier the workload of a machine, the severer the punishment is given to that machine. This way, a machine with a lighter workload tends to have a smaller priority value, and is more likely to be selected.

An example of the combination mechanism is shown as follows. We assume there are two candidate machines for one operation. The value obtained by original GP approach are V_{GP1} and V_{GP2}. The workloads of these two machines are denoted by WIQ_1 and WIQ_2. At one decision point, the workload in the entire shop floor (WIS, i.e. work in system) is the same for all the machines. According to the Eq. (1), (2) and (3), the different situations of the proposed representation are shown in Table 1, where $C()$ stands for the relationship between the corresponding elements. In Table 1, " $=, >, <$ " are used to show the relationship between V_{GP1} and V_{GP2}, WIQ_1 and WIQ_2, $component_1$ and $component_2$, $Priority_1$ and $Priority_2$.

Table 1 shows how the difference between machines workload (WIQ) can influence the final priority value ($Priority$) under different situations categorised by V_{GP}. All the cases of $V_{GP1} = V_{GP2}$ and most cases of $V_{GP1} > V_{GP2}$ and $V_{GP1} < V_{GP2}$ have the same trend that the WIQ is positively correlated with $component_1$ and $Priority$. This means a machine with lighter workload will get a smaller component 1 and final priority value, and vice versa. This ensures that the priority values obtained by GP is well considered. There are some special cases as follows. When $V_{GP1} > V_{GP2}$ ($V_{GP1} < V_{GP2}$), if $WIQ_1 < WIQ_2$ ($WIQ_1 > WIQ_2$), the final priority value ($Priority$) is not sure and depends on the accurate result according to Eq. (1), (2) and (3). Thus, on one hand, the machine with heavier workload but small V_{GP} still has chance to have a small priority value, and thus be selected. On the other hand, the machine with a poor V_{GP} can also be selected if its workload is very light.

Overall, the proposed priority function with the above two components is expected to consider the effect of both the workload of machines (in component 1) and the other features (in component 2), and combine them together properly by multiplication.

Table 1: The situations of the final priority values with the proposed components.

$C(V_{GP})$	$C(WIQ)$	$C(component_1)$	$C(Priority)$
	$<$	$<$	$<$
$=$	$=$	$=$	$=$
	$>$	$>$	$>$
	$<$	$<$	?
$>$	$=$	$=$	$>$
	$>$	$>$	$>$
	$<$	$<$	$<$
$<$	$=$	$=$	$<$
	$>$	$>$	?

4 Experiment Design

This work applies the framework (i.e. cooperative evolution genetic programming, CCGP) in [15], which is the current state-of-the-art algorithm of DFJSS to evolve routing and sequencing rules simultaneously. The proposed algorithm, which is named as rCCGP, is compared with CCGP [15] to verify its effectiveness on different job shop scenarios using three commonly used objectives, namely: (1) max-flowtime, (2) mean-flowtime, and (3) mean-weighted-flowtime.

To verify the performance of the evolved rules, we will use the test beds based on dynamic flexible simulation model [23, 24]. In order to test the effectiveness and robustness of proposed algorithm, six simulation scenarios based on the three objectives and two utilisation levels ($3 * 2$) are investigated.

4.1 Simulation Configuration

For dynamic simulation, commonly used configuration is adopted. In the job shop, there are ten machines, which has been proven to be a good showcase for job shop environment. There are 5000 jobs that need to be processed by ten machines. In order to get a steady state, a warm up period of 1000 jobs is used and we collect data from the next 5000 jobs. The new jobs keep coming until the 6000th job is finished. In each problem instance, jobs arrive stochastically according to a Poison process with rate λ and the average processing time for machines has mean μ. The utilisation is the proportion of time (p) that a machine is busy as shown in Eq. (4). Two utilisation levels (i.e. 0.85 and 0.95) are used in this paper.

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

In Eq. (4), P_M is the probability of a job visiting a machine. For example, if each job has two operations and there are ten machines, P_M is 2/10.

Different weights are set to jobs to indicate the urgency or importance of jobs (weight 1 (20%), weight 2 (60%), weight 4 (20%)). Uniform discrete distribution between 1 and 10 is designed for deciding both the number of operations per job and the number of candidate machines per operation. In addition, processing time of each operation will follow uniform discrete distribution between 1 and 99. In this work, the processing time of each operation is the same for all the candidate machines.

4.2 Parameter Settings

In our experiment, the terminal set and function set in [24] are adopted. The details are shown in Table 2. It is worth mentioned that "/" is the protected division that returns 1 if divided by 0.

Table 2: The terminal and function sets.

Terminals	**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
NPT	Median processing time for next operation
OWT	The waiting time of an operation
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
functions	$+$, $-$, $*$, $/$, max, min

The GP parameter settings follow the standard setting that have been used in most existing studies [20, 22, 15]. The population size is 1024 and the maximum depth of programs is 8. The crossover, mutation and reproduction rates are 0.80, 0.15 and 0.05, respectively. The rates of terminal and non-terminal selection are 0.10 and 0.90. Tournament selection is set as parent selection method with a tournament size of 7. The learning process continues until the generation reaches the maximum number of generations, which is set to 51.

5 Results and Discussions

The proposed GP approach with respect to the test performance and distribution of average objective value is investigated. 50 independent runs are executed, which assures that the results represent the average behavior instead of extreme situations. Then, the evolved rules are analysed. The $(-, +)$ marks show whether our proposed approaches converge significantly better or poorer than the basic approach in Wilcoxon rank sum test with a significance level of 0.05. Better results of min and max values are shown in bold. For the convenience of description, <obj, uti> indicates the simulation scenarios, where *obj* and *uti* are the objective and the utilisation level.

5.1 Test Performance of Evolved Rules

The comparison of the performance of the evolved rules obtained by rCCGP and CCGP are shown in Table 3. The statistical tests show that the evolved rules obtained by rCCGP are significantly better than the rules obtained by CCGP in four (scenario <Tmax, 0.95>, <Tmean, 0.85>, <WTmean, 0.85> and <WTmean, 0.95>) out of six scenarios. In scenario <Tmax, 0.85>, the performance is quite similar between rCCGP and CCGP with respective to the mean(sd), min and max value. In scenario <Tmean, 0.95>, although the rules evolved by rCCGP is not significantly better, the min, mean(sd) and max value are all better than their counterparts. In addition, rCCGP can reach better best-case and worse worst-case performance than CCGP in most cases.

Table 3: The mean(standard error), min and max of the objective value of rCCGP and CCGP over 50 independent runs for six dynamic scenarios.

Scenario	mean(sd)		min		max	
	rCCGP	CCGP	rCCGP	CCGP	rCCGP	CCGP
<Tmax,0.85>	1202.96(28.34)	1202.36(30.98)	1158.79	1152.97	1273.17	1270.26
<Tmax,0.95>	1864.83(30.69)**(-)**	1883.66(36.67)	**1813.77**	1829.47	**1942.71**	2034.34
<Tmean,0.85>	384.36(2.28)**(-)**	385.81(2.58)	**382.31**	382.72	396.83	395.81
<Tmean,0.95>	550.32(4.80)	552.14(6.46)	**543.18**	545.64	**569.43**	577.49
<WTmean,0.85>	828.30(6.02)**(-)**	829.38(3.47)	**823.08**	824.68	856.29	839.67
<WTmean,0.95>	1107.63(12.47)**(-)**	1110.72(10.77)	**1095.88**	1097.74	1169.34	1143.34

5.2 Distribution of Average Objective Value

Fig. 2 shows the violin plot of the average objective value obtained by rCCGP and CCGP. When further looking into the violin plot in Fig. 2, we can see that in most scenarios, the value obtained by rCCGP are distributed at lower positions compared with the value achieved by CCGP expect for scenario <Tmax, 0.85>. Although in <Tmean, 0.85>, <WTmean, 0.85> and <WTmean, 0.95>, there are some outliers which are higher than the maximum outlier in CCGP, the number of outliers is still smaller than their counterparts. Except for the outliers, the value obtained by rCCGP are more concentrated than that of CCGP, even in scenario <Tmax, 0.85>. This suggests that the performance of GP with the proposed new representation is more stable and effective.

5.3 Rule Analyses

Routing Rule. The proposed strategy for GP approach only works on routing process directly. The results show that the number of occurrences of the feature WIQ appeared in the final routing rules evolved by rCCGP is much lower than that of evolved by CCGP. For instance, in scenario <WTmean, 0.95>, the number of occurrences for feature WIQ in 50 best routing rules of CCGP (131) is

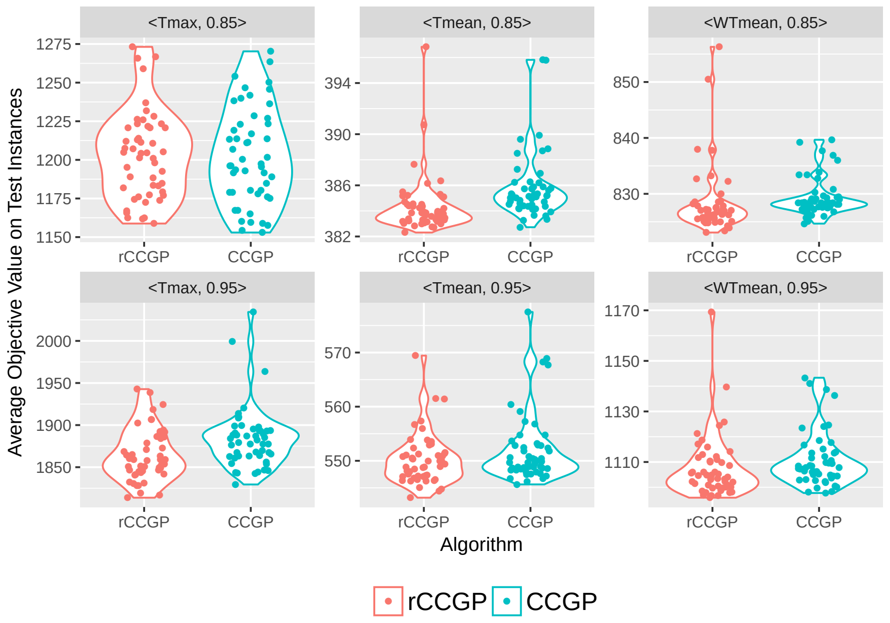

Fig. 2: Violin plot of average objective value obtained by rCCGP and CCGP.

41% lighter than in rCCGP (93). This is consistent with our expectation as in the proposed rCCGP, the workload of machines is considered in component 1, so that the GP-evolved component 2 can be more focused on the other features.

Fig. 3 shows the simplified component 2 of one promising routing evolved by rCCGP in the scenario <Tmax, 0.95>. It obtains a Tmax of 1831. There is no WIQ in the GP evolved rule (component 2). The component 2 consists of four parts and these four parts are added together. So, we can analyses them one by one. The details are shown in Eq. (5) - (8). Note that based on the definition in Table 2, PT and NPT for all the candidate machines of an operation are equal in our experiment. NOR for a job is also the same for routing at the decision point. In addition, TIS and WKR are equal for an ready operation. In summary, by definition given in Table 2, at any routing decision situation, the PT, NPT, NOR, TIS and WKR features can be treated as constants, as they are the same for all the candidate machines. Adding or subtracting a constant can be removed from the priority function, since it will not change the relative preference between machines. In our experiment, the smaller the priority value, the higher the priority.

$$\begin{aligned} part_1 =& 1 + max\{min\{NIQ, MWT\}, PT - MWT\} \\ & - min(W, MWT + NOR) \end{aligned} \tag{5}$$

The first part is shown as Eq. (5). Obviously, 1 and W are small constants, and thus $min(W, MWT + NOR)$ equals W in most cases since W is usually

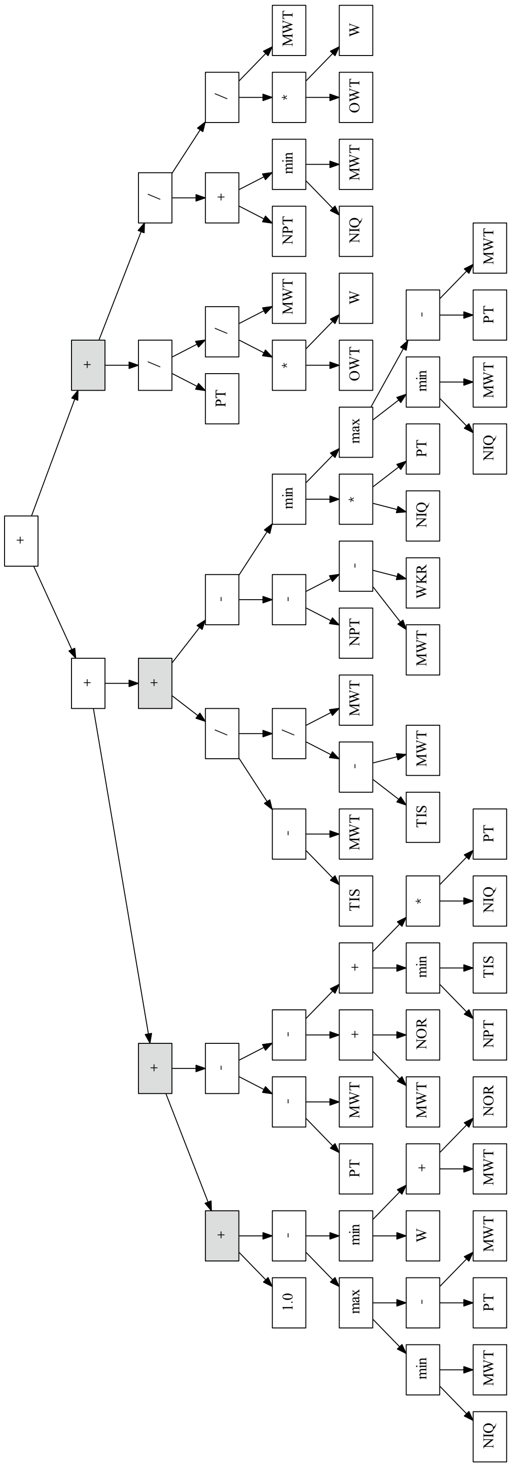

Fig. 3: Component 2 of one of the best performing rules for routing evolved by rCCGP in the scenario <tmax, 0.95>.

smaller than $MWT + NOR$. Therefore, Eq. (5) can be further simplified as $max\{min\{NIQ, MWT\}, PT - MWT\}$. It shows that the routing rule prefers machines with larger MWT, i.e. the earlier available machine (MWT = current time - machine ready time).

$$part_2 = PT * (1 + NIQ) - 2 * MWT - NOR + min\{NPT, TIS\} \tag{6}$$

The second part can be presented as Eq. (6). After ignoring some variables that can be considered as constants, Eq. (6) can be further simplified to $PT * NIQ - 2 * MWT$. It means the routing rule prefers machines with smaller NIQ (i.e. number of operations in the queue) and larger MWT.

$$\begin{aligned} part_3 =& NPT + WKR \\ &- min\{NIQ * PT, max\{min(NIQ, MWT), PT - MWT\}\} \end{aligned} \tag{7}$$

The third part is described as Eq. (7). It can be simplified as -$min\{NIQ * PT, max\{min(NIQ, MWT), PT - MWT\}$ after ignoring the first two constant terms. In addition, no matter what $min\{NIQ*PT, max\{min(NIQ, MWT), PT - MWT\}$ returns, it will be cancelled out by the same component in $part_1$ or $part_2$.

$$part_4 = \frac{MWT * (PT + NPT + min\{NIQ, MWT\})}{OWT * W} \tag{8}$$

The last part can be denoted as Eq. (8). OWT (i.e. the waiting time of an operation) for a ready operation in routing process equals zero in our experiment (i.e. the details are shown in section 2.2), therefore, the $part_4$ will return 1 (i.e. protected division).

According to the analysis mentioned above, this routing rule can be roughly simplified as $max\{min\{NIQ, MWT\}, PT - MWT\} - 2 * MWT$ or $PT * NIQ - 2 * MWT$.

Table 4 shows the number of times (proportion) a feature appears in the routing rule mentioned above and the counterpart in scenario <Tmax, 0.95>. The number of occurrences of features in the routing rule evolve by rCCGP and CCGP are 39 and 25, respectively. The number of designed terminals is 10 and the details can be seen in Table 2. The number of considered features in rCCGP (8) is more than that of evolved by CCGP (6). It suggests that the proposed GP approach can pay more attention to other features to get more information, thus to improve its performance. It is worth mentioning that the number of features considered in rCCGP is nine in fact because we consider WIQ by component 1.

MWT, which occurs 15 times, is the most frequently seen feature (15/39 = 0.38) of the evolved rule by rCCGP. For CCGP, MWT is also the most popularly used one, however, in terms of the proportion, it is less considered than that of in rCCGP (0.28 < 0.38). rCCGP and CCGP pay different attention to different features (i.e. the features have different importance in rCCGP and CCGP).

Sequencing Rule. The corresponding sequencing rule of the routing rule compared in last section is observed here. This is mainly to investigate what effect

Table 4: The number of occurrences (proportion) of features in one promising routing rule evolved by rCCGP (component 2) and CCGP in scenario <Tmax,0.95>.

Feature	Count (rCCGP)	Count (CCGP)
MWT	15 (0.38)	7 (0.28)
PT	6 (0.15)	0(0.00)
NIQ	5 (0.13)	6 (0.24)
NPT	3 (0.08)	0 (0.00)
TIS	3 (0.08)	2 (0.08)
W	3 (0.08)	0 (0.00)
NOR	2 (0.05)	2 (0.08)
OWT	2 (0.05)	5 (0.20)
WKR	0 (0.00)	0 (0.00)
WIQ	0 (0.00)	3 (0.12)
total	39	25

routing rule will have on sequencing rule. The sequencing rule evolved by rCCGP and CCGP are shown in Fig. 4 and Fig. 5. The size (i.e. number of nodes) of the sequencing rule is 45 evolved by rCCGP while the sequencing rule evolved by CCGP in the same scenario is 67. Obviously, the sequencing is much smaller. When looking at the sequencing rule evolved by rCCGP, the most popular pat-

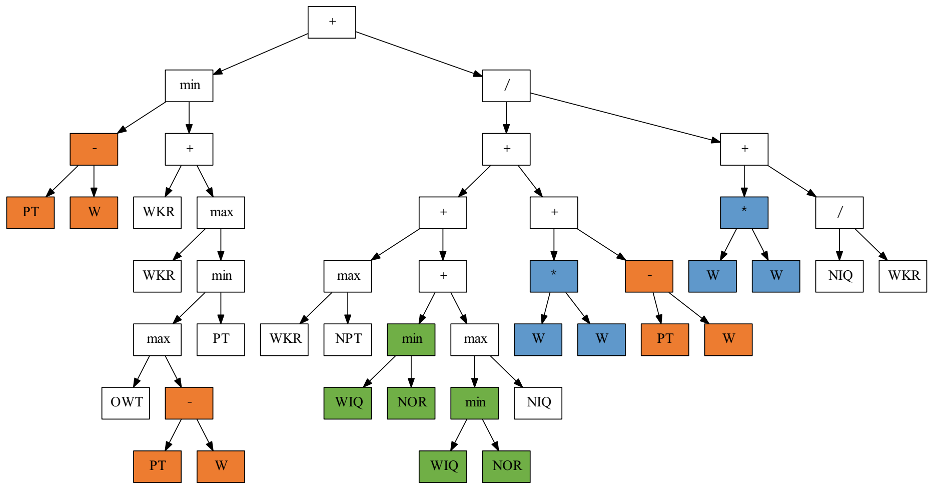

Fig. 4: The corresponding sequencing rule of the routing rule mentioned in last section evolved by rCCGP in the scenario <tmax, 0.95>.

tern is $PT - W$ followed by $W * W$ and $min\{WIQ, NOR\}$. WIQ/W appears most often (i.e. six times) followed by $PT - W$ in the sequencing rule evolved by CCGP. It means if the workload is not well considered in routing process, the machine might be assigned too many tasks. Thus, the sequencing rule should take WIQ into account. Intuitively, a machine with lots of work should take different processing strategy compared with a machine has fewer tasks.

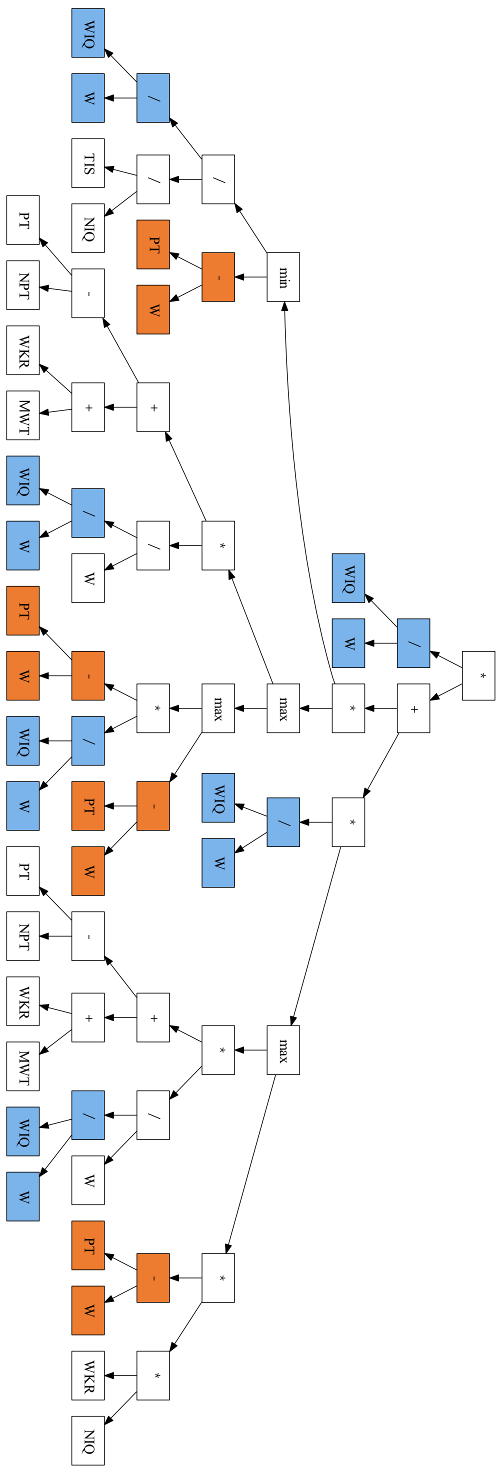

Fig. 5: The corresponding sequencing rule of the routing rule mentioned in last section evolved by CCGP in the scenario <tmax, 0.95>.

6 Conclusions and Future Work

The goal of this paper was to help GP evolve more effective dispatching rules for DFJSS. The goal has been successfully achieved by developing a new representation. The new representation is based on the domain knowledge that the workload of a machine WIQ is the dominating feature for making the routing decision. To fully explore the best way of using the other non-dominating features, the new representation was designed as a combination of two parts, one solely related to the workload of machines, and the other focused on the other non-dominating features. This way, GP can focus on exploring more appropriate ways of using the other features than the dominating feature.

The results show that the proposed GP approach with new representation can achieve significantly better performance in most of the involved scenarios. To be specific, the distributions of average objective values obtained by rCCGP in five out of six scenarios are better than that of in CCGP. This means the proposed new representation works well in almost all the examined instances. It confirms the effectiveness of the proposed component for workload information and the combination strategy. It is also known that the routing rule obtained by rCCGP can focus more on the non-dominating features with the proposed new representation. In addition, the evolved corresponding sequencing rule is also affected to consider different information and tended to be smaller than its counterpart. Overall, the results demonstrate that the proposed way of using domain knowledge successfully helps GP evolve more effective routing and sequencing rules for DFJSS.

In the future, more strategies will be investigated to make full use of the information provided by features. In addition, useful techniques will be adopted to inspect the evolved rules.

References

1. Nguyen, S.B.S., Zhang, M.: A hybrid discrete particle swarm optimisation method for grid computation scheduling. In: Evolutionary Computation (CEC), 2014 IEEE Congress on. pp. 483–490. IEEE (2014)
2. 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)
3. Brucker, P., Schlie, R.: Job-shop scheduling with multi-purpose machines. Computing 45(4), 369–375 (1990)
4. Oktaviandri, M., Hassan, A., Shaharoun, A.M.: Decision support tool for job shop scheduling with job cancellation. International Conference on Engineering of Tarumanagara (ICET) (2013)
5. Sabuncuoglu, I., Bayız, M.: Analysis of reactive scheduling problems in a job shop environment. European Journal of Operational Research 126(3), 567–586 (2000)
6. Bertsekas, D.P., Bertsekas, D.P., Bertsekas, D.P., Bertsekas, D.P.: Dynamic programming and optimal control, vol. 1. Athena Scientific Belmont, MA (2005)
7. Lawler, E.L., Wood, D.E.: Branch-and-bound methods: A survey. Operations Research 14(4), 699–719 (1966)

8. Nowicki, E., Smutnicki, C.: A fast taboo search algorithm for the job shop problem. Management Science 42(6), 797–813 (1996)
9. Pezzella, F., Morganti, G., Ciaschetti, G.: A genetic algorithm for the flexible job-shop scheduling problem. Computers & OR 35(10), 3202–3212 (2008)
10. Gomes, M.C., Barbosa-Póvoa, A.P., Novais, A.Q.: Reactive scheduling in a make-to-order flexible job shop with re-entrant process and assembly: a mathematical programming approach. International Journal of Production Research 51(17), 5120–5141 (2013)
11. 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)
12. Pickardt, C.W., Hildebrandt, T., Branke, J., Heger, J., Scholz-Reiter, B.: Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. International Journal of Production Economics 145(1), 67–77 (2013)
13. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: 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), 621–639 (2013)
14. Zhang, F., Mei, Y., Zhang, M.: Genetic programming with multi-tree representation for dynamic flexible job shop scheduling. In: Australasian Joint Conference on Artificial Intelligence. pp. 472–484. Springer (2018)
15. 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)
16. Zhang, F., Mei, Y., Zhang, M.: Surrogate-assisted genetic programming for dynamic flexible job shop scheduling. In: Australasian Joint Conference on Artificial Intelligence. pp. 766–772. Springer (2018)
17. Haupt, R.: A survey of priority rule-based scheduling. Operations-Research-Spektrum 11(1), 3–16 (1989)
18. 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)
19. Nguyen, S., Mei, Y., Zhang, M.: Genetic programming for production scheduling: a survey with a unified framework. Complex & Intelligent Systems 3(1), 41–66 (2017)
20. Mei, Y., Nguyen, S., Xue, B., Zhang, M.: An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. IEEE Transactions on Emerging Topics in Computational Intelligence 1(5), 339–353 (2017)
21. Nguyen, S., Mei, Y., Xue, B., Zhang, M.: A hybrid genetic programming algorithm for automated design of dispatching rules. Evolutionary Computation pp. 1–31 (2018)
22. Mei, Y., Zhang, M., Nyugen, S.: Feature selection in evolving job shop dispatching rules with genetic programming. In: Proceedings of the Genetic and Evolutionary Computation Conference 2016. pp. 365–372. ACM (2016)
23. Hildebrandt, T., Branke, J.: On using surrogates with genetic programming. Evolutionary Computation 23(3), 343–367 (2015)
24. Mei, Y., Nguyen, S., Zhang, M.: Evolving time-invariant dispatching rules in job shop scheduling with genetic programming. In: European Conference on Genetic Programming. pp. 147–163. Springer (2017)