Evolving Dispatching Rules for Multi-objective Dynamic Flexible Job Shop Scheduling via Genetic Programming Hyper-heuristics

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 one of the well-known combinational optimisation problems, which aims to handle machine assignment (routing) and operation sequencing (sequencing) simultaneously in dynamic environment. Genetic programming, as a hyper-heuristic method, has been successfully applied to evolve the routing and sequencing rules for DFJSS, and achieved promising results. In the actual production process, it is necessary to get a balance between several objectives instead of simply focusing only one objective. No existing study considered solving multi-objective DFJSS using genetic programming. In order to capture multi-objective nature of job shop scheduling and provide different trade-offs between conflicting objectives, in this paper, two well-known multi-objective optimisation frameworks, i.e. non-dominated sorting genetic algorithm II (NSGA-II) and strength Pareto evolutionary algorithm 2 (SPEA2), are incorporated into the genetic programming hyper-heuristic method to solve the multi-objective DFJSS problem. Experimental results show that the strategy of NSGA-II incorporated into genetic programming hyper-heuristic performs better than SPEA2-based GPHH, as well as the weighted sum approaches, in the perspective of both training performance and generalisation.**

I. Introduction

Job shop scheduling (JSS) is well-known as a typical combinatorial optimisation problem [1], which aims to increase productivity by optimising the processing sequence of tasks. In JSS, a number of jobs which consist of a sequence of operations need to be processed by a set of machines and each operation can only be processed on a predefined machine. However, in the real world, normally an operation can be processed by more than one machine. This leads to the need for the routing process (i.e. assign operation to a machine), which is the so-called flexible job shop scheduling (FJSS). FJSS is a hard problem because machine assignment (*routing*) and operation sequence (*sequencing*) should be addressed simultaneously.

In real-world manufacturing system, the working environment is dynamically changing by unpredictable real-time events, such as job arrival over time, machine breakdown and order cancellation. FJSS that takes dynamic events into consideration is named as dynamic flexible job shop scheduling (DFJSS). In this paper, for DFJSS, only job arrival event which is the most frequent and common factor in the shop floor, is taken into consideration. Dynamic scheduling is the key to solving the DFJSS problem.

Exact approaches such as integer programming [2] and mathematical programming [3], which aim to search for optimal solutions, have been used to handle the JSS problem. However, they are too time-consuming, especially when the problems are getting large. *Heuristic search methods* such as simulated annealing and genetic algorithm [4] have been investigated to search "near-optimal" solutions for solving the JSS problem with a reasonable time. However, they are faced with the problem of rescheduling and it is hard for them to handle dynamic events where lots of real time decisions are needed to be made quickly. It is worth mentioning that *dispatching rule* might be the most popular used heuristic to solve the DJSS problem. Dispatching rule, as priority function, is applied to calculate the priority values of operations at the decision point and chose operation with highest priority value as the next operation to be processed. However, manually designing dispatching rules is hard and time-consuming, and it is impractical for human to design all kinds of dispatching rules for use in different environments.

Hyper-heuristic [5] is an automated methodology for selecting or generating heuristics to solve hard computational search problems. Genetic programming (GP), as a hyper-heuristic method, was investigated to evolve dispatching rules in [6] for the first time. A general GP-based hyper-heuristic (GPHH) framework was presented in [7], [8]. The unique feature of GPHH is that its search space is heuristic instead of solution. The purpose for using GPHH is to improve the generalisation of dispatching rules by generating new heuristic using existing heuristics. It has successfully solved many problems in different fields such as timetabling [9] and job shop scheduling [10], [11]. For DFJSS, *routing rule* (for machine assignment) and *sequencing rule* (for operation sequencing) are needed to work together to get a schedule. To the best of our knowledge, GP with multi-tree representation is the state-of-the-art approach to evolve these two rules simultaneously [12].

In the shop floor, there are many objectives to be optimised. In this case, one may prefer to achieve a trade-off between different objectives rather than focusing on only one objective. In previous works, there are mainly two kinds of methods to handle multi-objective problem. An intuitive approach is the weighted sum method which combines all objectives into one single objective by predefined weights. The main drawback of this method is that the weights are not always available. In contrast, the Pareto-based methods aim to achieve a set of optimal solutions for users. In the reported literatures, multi-objective DFJSS has not yet been well considered.

For multi-objective problems, a decision maker may not want to have a huge number of Pareto optimal solutions at high computational cost [13]. They are often interested in obtaining a small number of evenly distributed solutions at low computational cost. Non-dominated sorting genetic algorithm II (NSGA-II) [14], [15] and strength Pareto evolutionary algorithm 2 (SPEA2) [16] are two common candidates for multi-objective methods in this purpose.

In this paper, we will incorporate the strategies of NSGA-II and SPEA2 into GPHH with multi-tree representation. It is worth mentioning that multi-tree representation is used to optimise routing rule and sequencing rule simultaneously by assigning two trees to one individual. This is the first time that the strategies of Pareto-based multi-objective approaches are incorporated into GPHH to handle DFJSS problems by evolving a set of combinations of routing and sequencing rules at the same time.

A. Goals

The overall goal of this paper is to incorporate the strategies of NSGA-II and SPEA2 into GPHH framework with multi-tree representation to achieve a trade-off between different objectives in the DFJSS problem. To achieve this, it has the following research objectives in this paper.

1) Design new approaches that incorporate the strategies of NSGA-II and SPEA2 into GPHH with multi-tree representation for DFJSS, respectively.
2) Compare the performance of the proposed methods.
3) Analyse the consistency of rule behaviour between training process and test process.

II. Background

In this section, the mechanism of how dispatching rule works for DFJSS is described and related works about FJSS, dynamic job shop scheduling (DJSS) and DFJSS are discussed.

A. Dispatching Rules for DFJSS

In traditional job shop scheduling, dispatching rule generally refers to sequencing rule. However, in DFJSS, a dispatching rule consists of a routing rule and a sequencing rule. Machine assignment will be done according to the priorities of machines obtained by routing rule. Operation sequencing will be made based on the priorities of operations obtained by sequencing rule. There are several unique features of using dispatching rule to solve DFJSS.

1) A dispatching rule, as a priority function, is time efficient and easy to implement. This leads itself to react dynamic events quickly without rescheduling.
2) A routing rule will be triggered to choose a machine for an operation when a new job comes or an operation is finished and its subsequent operation becomes a ready operation.
3) A sequencing rule will be triggered to choose an operation to be processed next when a machine becomes idle and its queue is not empty.
4) On one hand, all the operations will be allocated by routing rule first and then handled by sequencing rule for processing. On the other hand, once the conditions of rule triggers are met, the corresponding routing and sequencing action will be executed. That means the routing process and sequencing process are conducted in an interactive way.
5) The priorities of machines and operations are not fixed over time but can change dynamically as scheduling proceeds.

B. Related Work

There are many studies about solving DJSS [11], [17] and FJSS problems with GPHH [18], [19]. However, in FJSS, the routing rule is fixed and only sequencing rule is evolved. Yska et al. [20], [21] proposed a new GPHH algorithm with cooperative coevolution to explore the possibility of evolving both routing and sequencing rules together. The results showed that co-evolving the two rules together can lead to much more promising results than evolving the sequencing rule only. This is the first work that considered to evolve routing rule and sequence rule at the same time by GP. Zhang et al. [12], [22] proposed to evolve these two rules for DFJSS by assigning two trees to an individual. The results shows that GP with multi-tree representation is more effective and efficient than cooperative coevolution strategy. However, all of them work on single objective problem and there are a few literatures that work on FJSS, DJSS and DFJSS based on multiple objectives.

1) Multi-objective FJSS: An approach which makes use of particle swarm optimisation to assign operations to machines and simulated annealing algorithm to schedule operations on each machine was proposed to solve FJSS problems with multi-objective in [23]. However, weighted sum method was used to handle multi-objective problem and the two processes were conducted in two separate phases. Hybridisation of multi-objective particle swarm optimisation and local search was proposed to solve FJSS problems based on priority by dividing the particle into two parts in [24]. The first and second parts are designed for routing and sequencing respectively. However, this method cannot handle dynamic event with rescheduling. A teaching-and-learning based hybrid genetic-particle swarm optimisation was proposed to solve FJSS in [25]. Actually, the routing decision was made by a predefined rule (i.e. Least Processing Time). Tay et al. used GP to evolve dispatching rules for solving FJSS problems. However, the routing rule was fixed [18].

2) Multi-objective DJSS: A hybrid genetic algorithm and tabu search was proposed to solve multi-objective DJSS problem by rescheduling when unexpected disruptions occur

[26]. Nguyen et al. introduced to use cooperative coevolution genetic programming to automatically design scheduling policies for dynamic multi-objective job shop scheduling with the strategies of non-dominated sorting genetic algorithm II, strength Pareto evolutionary algorithm 2 and harmonic distance-based multi-objective evolution [11]. Later, Nguyen et al. used local search heuristics to enhance the quality of evolved dispatching rules [27]. However, all of them related only to dynamic job shop scheduling problem and the cooperative coevolution has potential assumption that the two evolved rules are independent, which is not the case in DFJSS.

3) Multi-objective DFJSS: NSGA-II was applied to solve DFJSS problems with random machine breakdowns in [28]. This paper mainly focused on investigate the robustness of scheduling based on utilising the available information about machine breakdowns. In addition, four commonly used rules were used to decide the sequence of operations. However, the sequencing rule was fixed and reschedule was needed to handle the dynamic event. A multi-objective evolutionary algorithm based approach was proposed to solve DFJSS in [29]. The main problem is that rescheduling is triggered at each dynamic event point which is time-consuming and cannot react quickly, especially in large scale problem.

In summary, there are four limitations in the previous literatures.

a) The works [23] using weighted sum are not always applicable, as the weights may not be available.
b) The works [23] handle routing and sequencing decisions one by one in two phases, which is not practical in real life.
c) The works [18], [23], [25], [28] handle routing and sequencing decisions simultaneously, however, only one rule is optimised while the other is fixed as an intuitive rule.
d) The works [24]–[26], [28], [29] using heuristic search methods face the shortcomings of not being able to solve dynamic problems without rescheduling.

To the best of our knowledge, there is no approach that can successfully solve the problems as described above in the four limitations. To this end, this paper aims to handle these limitations well.

III. Multi-objective GPHH for Dynamic Flexible Job Shop Scheduling

This section describes the proposed new GPHH methods to solve DFJSS problems by evolving routing rule and sequencing rule simultaneously. We will first introduce how routing rule and sequencing rule are represented in GP. Then, the proposed multi-objective GPHH with multi-tree representation methods incorporated with the strategies of NSGA-II, SPEA2 will be illustrated.

A. Representation

Tree based GP is widely used to solve complex problems with the programs using tree-based representation in its population. Fig. 1 shows an example of a tree-based representation

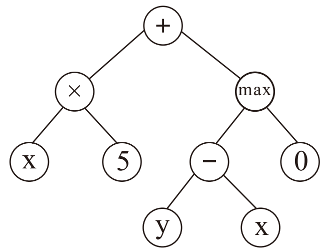


Fig. 1. An example of tree-based GP program.

of the program $5x + max(y - x, 0)$. In this program, the terminals consist of the variables $\{x, y\}$ and two constants $\{5, 0\}$ and the functions compose of $\{\times, +, -, max\}$. The terminals are the leaves of the tree while the functions cannot be located at the leaves of the trees. In GP, the collections of the terminals and functions are called *terminal set* and *function set*, respectively. Obviously, the program is the combination of the components in terminal set and function set.

When using multi-tree representation, one individual can contain more than one tree. This special trait can be utilised to tackle different problems at the same time. For handling the DFJSS problem, in [12], each individual contains two trees. The first tree is used to evolve the routing rule and the second tree is designed to evolve the sequencing rule. In this way, these two rules can be evolved simultaneously with the evolutionary process proceeds. An example of one individual for DFJSS is shown in Fig. 2.

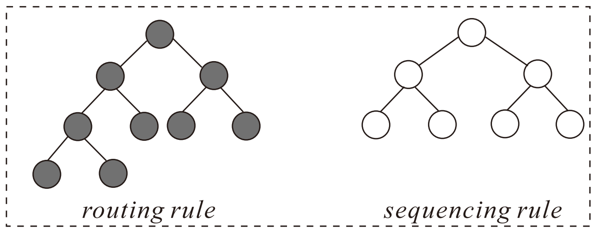


Fig. 2. An example of one individual for DFJSS.

B. Multi-objective GPHH for DFJSS

In this paper, we hybridise NSGA-II and SPEA2 with GPHH with multi-tree representation for solving multi-objective DFJSS. The resultant algorithms are named as NSMTGP-II and SPMTGP2, respectively.

In NSGA-II, every individual in the population has two attributes which are non-dominated rank and crowding. The individual with better rank and smaller crowing distance will be preferred. In SPEA2, the individual with smaller strength value is preferred. The flowchart of NSMTGP-II and SPMTGP2 are shown in Fig. 3 and Fig. 4.

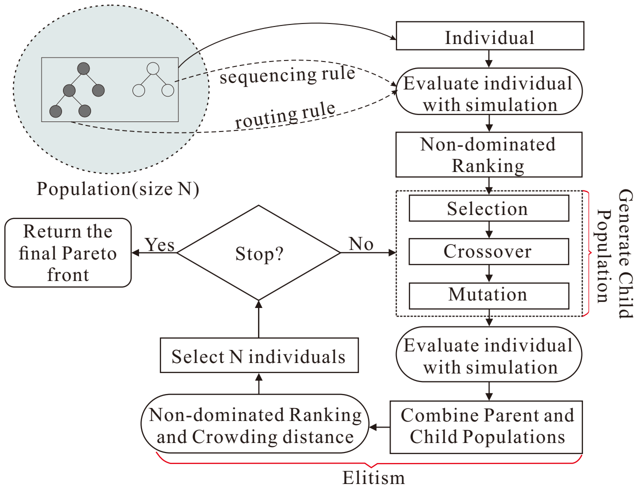

Individual
sequencing rule
routing rule
Evaluate individual with simulation
Non-dominated Ranking
Population(size N)
Selection
Crossover
Mutation
Generate Child Population
Return the final Pareto front
Yes
Stop?
No
Select N individuals
Evaluate individual with simulation
Non-dominated Ranking and Crowding distance
Combine Parent and Child Populations
Elitism

Fig. 3. Overview of NSMTGP-II.

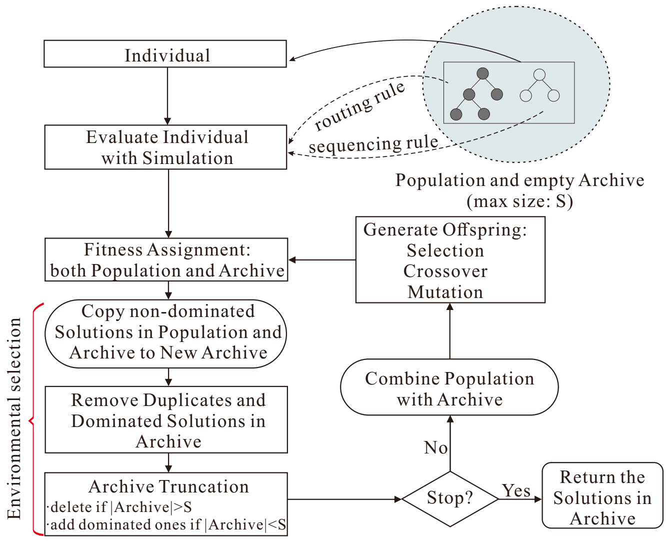

Individual
Evaluate Individual with Simulation
routing rule
sequencing rule
Population and empty Archive (max size: S)
Fitness Assignment: both Population and Archive
Generate Offspring: Selection Crossover Mutation
Copy non-dominated Solutions in Population and Archive to New Archive
Environmental selection
Remove Duplicates and Dominated Solutions in Archive
Combine Population with Archive
No
Archive Truncation
·delete if |Archive|>S
·add dominated ones if |Archive|<S
Stop?
Yes
Return the Solutions in Archive

Fig. 4. Overview of SPMTGP2.

IV. Experiment Design

For the purpose of this study, we assume the machines are independent from each other and the set up times and move times between operations are negligible. The comparison design, simulation model and parameter settings in this paper will be illustrated as follows.

A. Comparison Design

In order to verify the performance and robustness of the proposed approaches, six different scenarios are designed based on three objectives (e.g. max-flowtime, mean-flowtime and mean-weighted-flowtime) and two utilisation levels. The details can be found in Table I. We will investigate them by pairwise comparison. In this paper, each compared algorithm is run 50 times independently. For a more comprehensive comparison, we also compare NSMTGP-II and SPMTGP2 with the weighted-sum based GP (WMTGP). WMTGP simply combines the two objectives into a single one during

TABLE I
Six scenarios

Scenario	Objective 1	Objective 2	Utilisation
1	max-flowtime	mean-flowtime	0.85
2	max-flowtime	mean-flowtime	0.95
3	max-flowtime	mean-weighted-flowtime	0.85
4	max-flowtime	mean-weighted-flowtime	0.95
5	mean-flowtime	mean-weighted-flowtime	0.85
6	mean-flowtime	mean-weighted-flowtime	0.95

the optimisation. In the experiment, we examine six weight vectors, i.e. (0,1), (0.2,0.8), (0.4,0.6), (0.6,0.4), (0.8,0.2) and (1,0), which correspond to different preferences between the objectives.

B. Simulation Model

In this paper, DFJSS simulation is used to verify the performance and commonly used configuration is adopted [12]. The task in the simulation is to process 5000 jobs by ten machines. In order to distinguish the importance of different jobs, weights are assigned to each job. 20%, 60% and 20% jobs will have weight 1, 2 and 4, respectively.

Jobs will arrive the job shop stochastically according to a Possion process with rate λ. The number of operations of a job and the number of candidate machines of an operation follow a uniform discrete distribution between one and ten. Uniform discrete distribution between 1 and 99 is applied to decide the processing time of an operation.

The utilisation level (p) is an important indicator that shows how busy a machine is. It can be expressed as Eq. (1). In Eq. (1), μ is the average processing time of 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.

$$p = \lambda * \mu * P_M \tag{1}$$

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.

TABLE II
The terminal set.

Notation	Description
NIQ	The number of operations in the queue
WIQ	The workload in the queue of a machine
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	Waiting time in system for a job

C. Parameter Settings

In our experiment, the terminals are shown in Table II [22]. These terminals can provide the information (e.g. machine-related, job-related and system-related) of the job shop to help GP evolve dispatching rule. The function set consists of

TABLE III
THE **MEAN AND STANDARD DEVIATION** OF THE **HV** OF NSMTGP-II, SPMTGP2 AND WMTGP OVER 50 INDEPENDENT RUNS IN **TRAINING AND TEST PROCESS** IN SIX SCENARIOS

Scenario	**Training**			**Test**		
	NSMTGP-II	***SPMTGP2***	***WMTGP***	***NSMTGP-II***	***SPMTGP2***	***WMTGP***
1	**0.923(0.041)***	0.889(0.046)(-)	0.787(0.061)	**0.991(0.050)***	0.954(0.056)(-)	0.833(0.032)
2	**0.928(0.019)***	0.908(0.022)(-)	0.505(0.050)	**0.919(0.028)***	0.895(0.036)(-)	0.602(0.049)
3	**0.912(0.042)***	0.877(0.056)(-)	0.723(0.079)	**0.458(0.021)***	0.439(0.031)(-)	0.350(0.049)
4	**0.879(0.020)***	0.857(0.033)(-)	0.494(0.062)	**0.772(0.042)***	0.746(0.040)(-)	0.536(0.045)
5	**0.815(0.184)**	0.660(0.263)	0.904(0.054)(-)*	**0.114(0.029)**	0.109(0.143)(-)	0.120(0.007)
6	**0.761(0.109)***	0.696(0.144)	0.737(0.059)	0.204(0.121)	0.169(0.052)	0.189(0.014)

$^{\text{bold}}$ The significantly better approach between NSMTGP-II and SPMTGP2.
* The significantly better approach between NSMTGP-II and WMTGP.
$^{-}$ The significantly better approach between SPMTGP2 and WMTGP.

TABLE IV
THE MEAN AND STANDARD DEVIATION OF THE **IGD** OF NSMTGP-II, SPMTGP2 AND WMTGP OVER 50 INDEPENDENT RUNS IN **TRAINING AND TEST PROCESS** IN SIX SCENARIOS

Scenario	**Training**			**Test**		
	NSMTGP-II	***SPMTGP2***	***WMTGP***	***NSMTGP-II***	***SPMTGP2***	***WMTGP***
1	**0.037(0.023)***	0.057(0.027)(-)	0.234(0.043)	**0.074(0.030)***	0.088(0.027)(-)	0.153(0.017)
2	**0.032(0.012)***	0.044(0.013)(-)	0.324(0.029)	**0.039(0.018)***	0.055(0.022)(-)	0.351(0.045)
3	**0.037(0.023)***	0.060(0.031)(-)	0.225(0.058)	**0.036(0.014)***	0.053(0.025)(-)	0.214(0.015)
4	**0.041(0.015)***	0.060(0.025)(-)	0.354(0.044)	0.032(0.08)*	0.035(0.009)(-)	0.066(0.013)
5	**0.177(0.146)**	0.305(0.233)	0.130(0.035)(-)	**1.013(0.072)**	1.053(0.185)	0.999(0.015)(-)
6	**0.104(0.083)***	0.151(0.111)	0.144(0.055)	0.850(0.139)	0.902(0.092)	0.853(0.027)

$^{\text{bold}}$ The significantly better approach between NSMTGP-II and SPMTGP2.
* The significantly better approach between NSMTGP-II and WMTGP.
$^{-}$ The significantly better approach between SPMTGP2 and WMTGP.

$\{+, -, *, /, max, min\}$. The "/" operator is protected division, returning 1 if dividing by zero.

Ramped-half-and-half method is used to generate the initial population and the minimum and maximum depth of individuals are two and six. There are 1024 individual in the population and maximum generation is 51. The maximum depth of each individual is eight. Crossover, mutation and reproduction are three genetic operators to generate new population. Their rates are 0.80, 0.15 and 0.05. The rates of terminal and non-terminal selection are 0.10 and 0.90. Tournament selection with a tournament size of seven is applied to select individuals for genetic operators.

V. RESULTS AND ANALYSIS

In this section, the performance of the proposed approaches will be measured. Then, further analysis of the behaviour of the evolved dispatching rule will be conducted.

A. Test Performance of Evolved Dispatching Rules

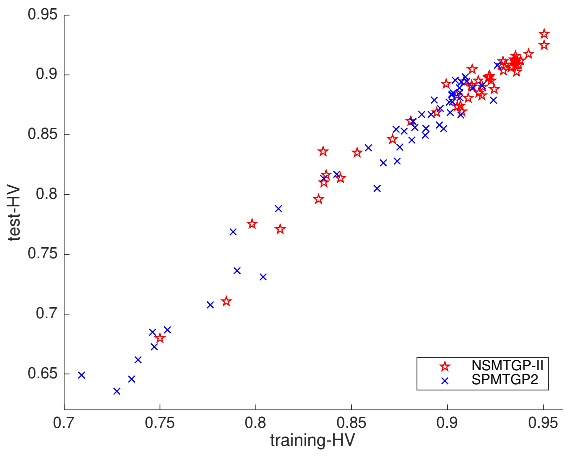
For multi-objective problems, two widely used performance indicators are hypervolume (HV) [16] and inverted generational distance (IGD) [27]. HV is used to measure the Pareto front by taking the number of optimal solutions of Pareto front and its uniformity. IGD is applied to indicate the optimal degree (i.e. how close is the optimal solution to the true Pareto front) and coverage of the optimal solutions. Since the true Pareto front is unknown in this research, we will use a reference Pareto front instead. The reference Pareto front is generated by getting all the non-dominated dispatching rule found by three approaches (e.g. NSMTGPII, SPMTGP2 and WMTGP with all weight vectors) in 50 independent runs. It is worth mentioning that the non-dominated dispatching rules of WMTGP are generated by combining the results of WMTGP with all weight vectors together and the computational cost of WMTGP is six times as NSMTGP-II and SPMTGP2.

For the value of HV, the larger the better. For the value of IGD, the smaller the better. The Wilcoxon rank sum test with a significance level of 0.05 will be used to verify the performance of the proposed approaches.

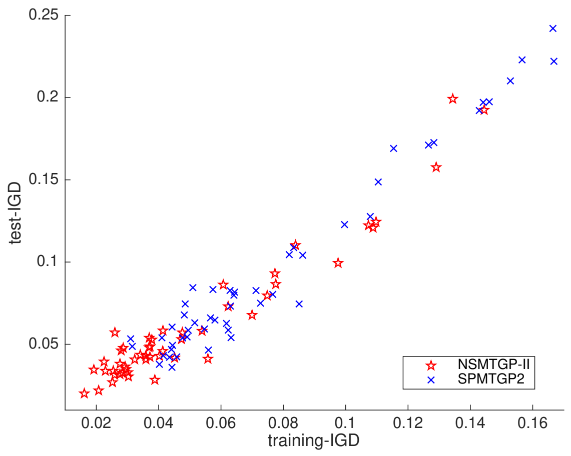
Table III shows mean and standard deviation of HV of the proposed three methods in traing and test process. In test process, from scenario 1 to scenario 4, NSMTGP-II performs significantly better than both SPMTGP2 and WMTGP in test process. In scenario 5, NSMTGP-II is significantly better than SPMTGP2 and SPMTGP2 is better than WMTGP. However, in scenario 6, there is no significantly difference among NSMTGP-II, SPMTGP2 and WMTGP.

Table IV shows mean and standard deviation of IGD of the proposed three methods in training and test process. In test process, we can see that NSMTGP-II and SPMTGP2 are significantly better than WMTGP in four scenarios (scenario 1, 2, 3 and 4). In scenario 5, NSMTGP-II is significantly better than SPMTGP2 and SPMTGP2 is significantly worse than WMTGP. In addition, NSMTGP-II is not significantly better than WMTGP. There is no significantly difference among the proposed approaches in scenario 6.

Table III and Table IV also show that both for HV and IGD, the training process shows much better results than that in test process. NSMTGP-II is significantly better than SPMTGP2 in



(a) The training HV versus test HV.



(b) The training IGD versus test IGD.

Fig. 5. The training HV and IGD versus test HV and IGD based on the 50 final results of NSMTGP-II and SPMTGP2 in scenario 3.

all scenarios in terms of both HV and IDG.

In order to show the generalisation of NSMTGP-II and SPMTGP2, Fig. 5 shows the training HV (IGD) versus test HV (IGD) scatter plot based on 50 final results of NSMTGP-II and SPMTGP-2 in scenario 3. From the figure, it is clear that both the training and test HV and IGD of NSMTGP-II are much better that of SPMTGP2. The generalisation of both algorithms are similar in terms of the correlation between training and test HV and IGD.

In general, from the perspective of HV and IGD, NSMTGP-II is more promising among the three approaches in most scenarios. This indicates its *effectiveness* and *robustness*.

B. Consistency of Rule Behaviour

It is interesting to investigate the behaviour consistency of evolved rules from training process to test process. This is very important issue in DFJSS when using GPHH because the output of GPHH is a heuristic (dispatching rule) rather than a solution. In practice, one can select only one heuristic from the training performance, and expect it to show consistent test performance in terms of the trade-off between the objectives.

Fig. 6 shows the fitnesses of the Pareto front evolved by NSMTGP-II versus their corresponding test objectives in one of the independent runs in the first scenario. We can see that most of test points have a good consistency with their training points. It indicates that the user can always expect consistent preference between the objectives on the unseen test data. The rules evolved by SPMTGP2 have the same problem.

In order to measure the behaviour consistency, we propose to use the *ratio difference* of training points and test points as a measure. For example, if the training fitness of a rule is (230,100) and its test objective is (150,50), the ratio difference is 0.7 (|230/100-150/50|). The ratio here is designed to show the different degrees (behaviours) of concentration of the evolved rules on different objectives. Thus, the ratio difference can be used to measure the behaviour difference between different rules. The minimum value of ratio difference is zero, which indicates the rules have the same behaviour. Given a set of rules, the mean value of the ratio differences is recorded. The smaller the value, the better.

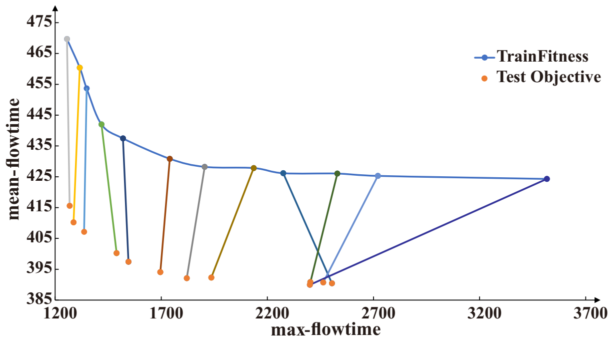
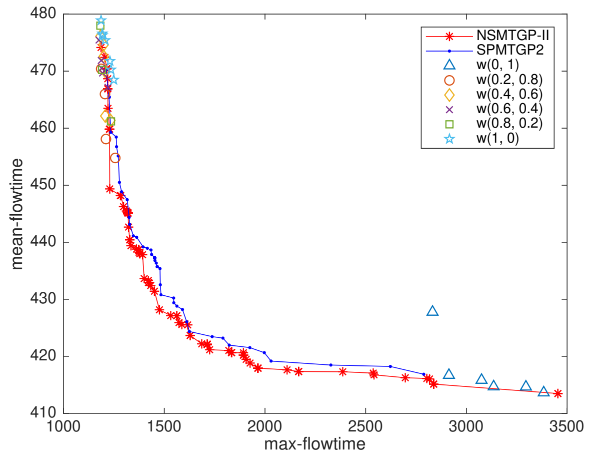


Fig. 6. An example of training fitness versus test objective of NSMTGP-II in one of the independent runs of scenario 1.

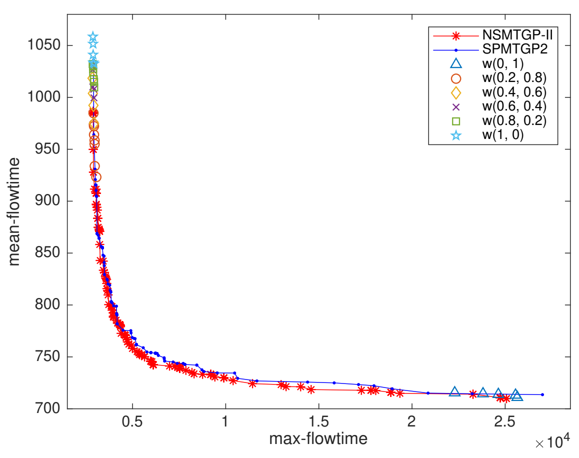
TABLE V
THE MEAN AND STANDARD DEVIATION VALUE OF **RATIO DIFFERENCE** BETWEEN TRAINING AND TEST FITNESSES OF NSMTGP-II, SPMTGP2 OVER 50 INDEPENDENT RUNS IN SIX SCENARIOS

Scenario	Mean(StdDev)	
	NSMTGP-II	*SPMTGP2*
1	**0.303(0.088)**	0.317(0.143)
2	**2.082(0.710)**	2.461(1.350)
3	**0.151(0.071)**	0.171(0.110)
4	1.952(0.636)	1.858(0.985)
5	0.009(0.001)	0.008(0.001)
6	0.035(0.006)	0.035(0.006)

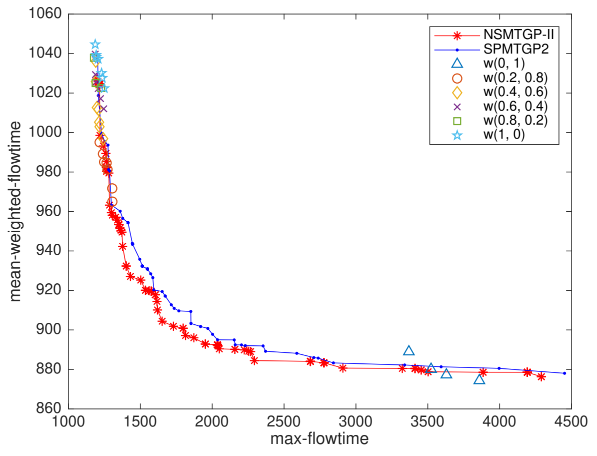
Table V shows the mean and deviation value of ratio difference obtained by NSMTGP-II and SPMTGP2 over 50 independent runs. We can see that there is no significantly difference between NSMTGP-II and SPMTGP2. However, the mean and standard deviation values obtained by NSMTGP-



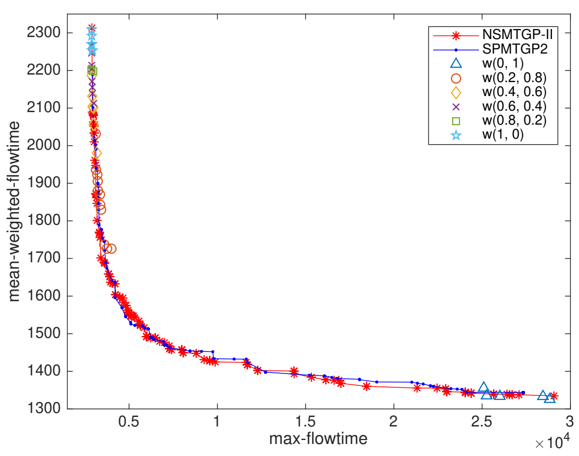
(a) Pareto fronts of NSMTGP-II, SPMTGP2 and WMTGP in scenario 1.



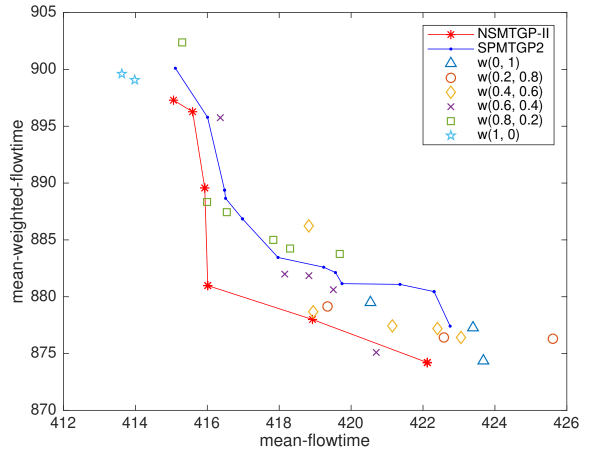
(b) Pareto fronts of NSMTGP-II, SPMTGP2 and WMTGP in scenario 2.



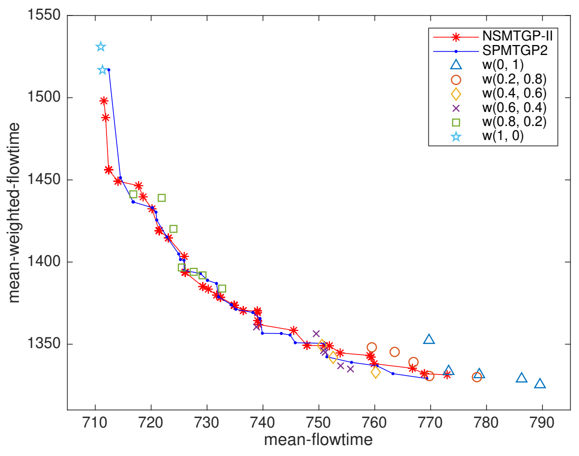
(c) Pareto fronts of NSMTGP-II, SPMTGP2 and WMTGP in scenario 3.



(d) Pareto fronts of NSMTGP-II, SPMTGP2 and WMTGP in scenario 4.



(e) Pareto fronts of NSMTGP-II, SPMTGP2 and WMTGP in scenario 5.



(f) Pareto fronts of NSMTGP-II, SPMTGP2 and WMTGP in scenario 6.

Fig. 7. Pareto fronts of NSMTGP-II, SPMTGP2 and WMTGP in different scenarios in training process.

II are smaller than that of SPMTGP2 in half scenarios. It means that the propsoed methods can evolve rules with good behaviour consistency, especially NSMTGP-II.

C. Insight of the Distribution of Evolved Rules

The evolved Pareto front (i.e. non-dominated solutions generated by the optimal solutions of 50 independent rules) by the proposed three approaches from scenario 1 to scenario 6 are shown in Fig. 7. In Fig. 7, for weights setting, the first (second) value is the weight for the objective indicated on X-axis (y-axis). It is obvious that the evolved Pareto front by NSMTGP-II is significantly better than SPMTGP2 and WMTGP (i.e. with different set of weights). It is interesting that weighted sum method get a good dispatching rule with the weight (0.6, 0.4) in scenario 6. It indicated in some case, weighted sum method can get better performance, however, it is not easy to predefine appropriate weights. In addition, it is more complex than NSMTGP-II and SPMTGP2 method because different independent runs should be conducted independently.

VI. Conclusions and Future Work

In this paper, we incorporated the strategies of NSGA-II and SPEA2 into GPHH with multi-tree representation to solve DFJSS. To the best of knowledge, this paper is the first time that the strategies of NSGA-II and SPEA2 was incorporated into GPHH with multi-tree representation to solve the DFJSS problem by evolving routing and sequencing rule simultaneously. In addition, we propose to use ratio difference as a metric to measure the consistency of rule behaviour. The experimental results showed that both the proposed methods can work well on DFJSS. Moreover, NSMTGP-II performs better than SPMTGP2 and the rules evolved by NSMTGP-II have promising rule consistency.

In the future, new strategies will be proposed to improve the performance of NSMTGP-II and new metric of rule behaviour consistency will be also investigated.

References

[1] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of operations research*, vol. 1, no. 2, pp. 117–129, 1976.

[2] R. E. Bixby, E. A. Boyd, and R. Z. Ríos-Mercado, Eds., *Integer Programming and Combinatorial Optimization, 6th International IPCO Conference, Houston, Texas, USA, June 22-24, 1998, Proceedings*, ser. Lecture Notes in Computer Science, vol. 1412. Springer, 1998.

[3] W. L. Winston, M. Venkataramanan, and J. B. Goldberg, *Introduction to mathematical programming*. Thomson/Brooks/Cole Duxbury; Pacific Grove, CA, 2003, vol. 1.

[4] E. K. Burke, G. Kendall *et al.*, *Search methodologies*. Springer, 2005.

[5] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches," in *Handbook of metaheuristics*. Springer, 2010, pp. 449–468.

[6] K. Miyashita, "Job-shop scheduling with genetic programming," in *Proceedings of the 2nd Annual Conference on Genetic and Evolutionary Computation*. Morgan Kaufmann Publishers Inc., 2000, pp. 505–512.

[7] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Computational Intelligence*. Springer, 2009, pp. 177–201.

[8] S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: a survey with a unified framework," *Complex & Intelligent Systems*, vol. 3, no. 1, pp. 41–66, 2017.

[9] M. B. Bader-El-Den, R. Poli, and S. Fatima, "Evolving timetabling heuristics using a grammar-based genetic programming hyper-heuristic framework," *Memetic Computing*, vol. 1, no. 3, pp. 205–219, 2009.

[10] R. Hunt, M. Johnston, and M. Zhang, "Evolving less-myopic scheduling rules for dynamic job shop scheduling with genetic programming," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*. ACM, 2014, pp. 927–934.

[11] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic design of scheduling policies for dynamic multi-objective job shop scheduling via cooperative coevolution genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 2, pp. 193–208, 2014.

[12] F. Zhang, Y. Mei, and M. Zhang, "Genetic programming with multi-tree representation for dynamic flexible job shop scheduling," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 472–484.

[13] Y. Sun, G. G. Yen, and Z. Yi, "Igd indicator-based evolutionary algorithm for many-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, 2018.

[14] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[15] Y. Sun, G. G. Yen, H. Mao, and Z. Yi, "Manifold dimension reduction based clustering for multi-objective evolutionary algorithm," in *IEEE Congress on Evolutionary Computation*. IEEE, 2016, pp. 3785–3792.

[16] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[17] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 621–639, 2013.

[18] J. C. Tay and N. B. Ho, "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems," *Computers & Industrial Engineering*, vol. 54, no. 3, pp. 453–473, 2008.

[19] T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "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, 2010, pp. 257–264.

[20] D. Yska, Y. Mei, and M. Zhang, "Genetic programming hyper-heuristic with cooperative coevolution for dynamic flexible job shop scheduling," in *European Conference on Genetic Programming*. Springer, 2018, pp. 306–321.

[21] ——, "Feature construction in genetic programming hyper-heuristic for dynamic flexible job shop scheduling," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2018, Kyoto, Japan, July 15-19, 2018*, 2018, pp. 149–150.

[22] F. Zhang, Y. Mei, and M. Zhang, "Surrogate-assisted genetic programming for dynamic flexible job shop scheduling," in *Australasian Joint Conference on Artificial Intelligence*. Springer, 2018, pp. 766–772.

[23] W. Xia and Z. Wu, "An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems," *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 409–425, 2005.

[24] G. Moslehi and M. Mahnam, "A pareto approach to multi-objective flexible job-shop scheduling problem using particle swarm optimization and local search," *International Journal of Production Economics*, vol. 129, no. 1, pp. 14–22, 2011.

[25] X. Huang, Z. Guan, and L. Yang, "An effective hybrid algorithm for multi-objective flexible job-shop scheduling problem," *Advances in Mechanical Engineering*, vol. 10, no. 9, pp. 1–14, 2018.

[26] L. Zhang, L. Gao, and X. Li, "A hybrid genetic algorithm and tabu search for a multi-objective dynamic job shop scheduling problem," *International Journal of Production Research*, vol. 51, no. 12, pp. 3516–3531, 2013.

[27] S. Nguyen, M. Zhang, and K. C. Tan, "Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems," in *IEEE Congress on Evolutionary Computation*, 2015, pp. 2781–2788.

[28] J. Xiong, L.-n. Xing, and Y.-w. Chen, "Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns," *International Journal of Production Economics*, vol. 141, no. 1, pp. 112–126, 2013.

[29] X.-N. Shen and X. Yao, "Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems," *Information Sciences*, vol. 298, pp. 198–224, 2015.