Feature Construction in Genetic Programming Hyper-Heuristic for Dynamic Flexible Job Shop Scheduling

Daniel Yska
Victoria University of Wellington
New Zealand
d.yska249@gmail.com

Yi Mei
Victoria University of Wellington
New Zealand
yi.mei@ecs.vuw.ac.nz

Mengjie Zhang
Victoria University of Wellington
New Zealand
mengjie.zhang@ecs.vuw.ac.nz

ABSTRACT

Genetic Programming Hyper-Heuristic (GPHH) has shown success in evolving dispatching rules for dynamic Flexible Job Shop Scheduling (FJSS). In this paper, we focus on feature construction to improve the effectiveness and efficiency of GPHH, and propose a GPHH with Cooperative Co-evolution with Feature Construction (CCGP-FC). The experimental results showed that the proposed CCGP-FC could improve the smoothness of the convergence curve, and thus improve the stability of the evolutionary process. There is a great potential to improve the FC methods, such as filtering the meaningless building blocks, and incorporating with feature selection to improve the terminal set.

CCS CONCEPTS

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

KEYWORDS

genetic programming, hyper-heuristic, dynamic scheduling

ACM Reference Format:
Daniel Yska, Yi Mei, and Mengjie Zhang. 2018. Feature Construction in Genetic Programming Hyper-Heuristic for Dynamic Flexible Job Shop Scheduling. In *GECCO '18 Companion: Genetic and Evolutionary Computation Conference Companion, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3205651.3205741

1 INTRODUCTION

Flexible Job Shop Scheduling (FJSS) [4] aims to complete a set of jobs (i.e. a sequence of operations) with a set of machines to minimise some objective(s) such as makespan and flowtime. Each machine can process only one operation at a time, and each operation can only be process by a set of candidate machines (called *options*).

In dynamic FJSS, unexpected job arrivals may occur while the schedule is being executed, and the schedule must be adjusted to respond to the environmental change. *Dispatching rule* is a very promising strategy to this end. In dynamic FJSS, there are two types of rules, i.e. *routing rule* that selects an option from all the options once an operation becomes ready, and *sequencing rule* that selects an operation out of a machine's queue whenever it becomes idle. The two rules work together to make real-time decisions.

GECCO '18 Companion, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5764-7/18/07.
https://doi.org/10.1145/3205651.3205741

Genetic Programming Hyper-heuristics (GPHH) is an effective method to automatically evolve effective rules. For FJSS, we proposed a Cooperative Co-evolution GP (CCGP) [4] to co-evolve the routing and sequencing rules, which significantly outperformed the existing GP (e.g. [3]) which fixed the routing rule intuitively.

This paper aims to further improve the effectiveness of CCGP by feature construction. We construct high-level features from the low-level raw features. By using the constructed feature, better rules can be found more easily. A new feature construction method is proposed, and the constructed features are analysed.

2 PROPOSED APPROACH

The proposed GPHH algorithm with CC and Feature Construction (CCGP-FC) has the following three stages.

(1) Run the `NiSuCCGP` method [1] to obtain a diverse set of promising rules.
(2) Conduct feature construction based on the obtained rules, and add the constructed features into the terminal sets.
(3) Run `CCGP` [4] using the extended terminal sets to get the final rules.

In CCGP-FC, the novel contribution is on the second stage. In this paper, we propose different feature construction mechanisms.

The feature construction is similar to the feature selection in [1]. Given a diverse set of promising rules, we first extract all the depth-2 subtrees from all the rules as candidate building blocks. Then, we calculate the *contribution* of each subtree to each rule using the formula in [1]. Finally, we select the most important subtrees based on their contributions to the rules.

Filtering. Note that there can a large number of subtrees, making the contribution calculation very time consuming. To improve efficiency, we design a filter to remove the subtrees that are not worth calculating the contribution. We remove a subtree from the candidate set if (1) it can be simplified (e.g. $\max\{\mathtt{PT}, \mathtt{PT}\}$), or (2) it is a meaningless combination between terminals (as defined in [2]).

Important Subtree Selection. The selection is a weighted voting process. Each rule identifies the useful subtrees to itself, and votes for them (i.e. giving them its voting weight, which is proportional to its fitness). There are two open questions in this process: *(a) how each rule identifies the useful subtrees*, and *(b) how to select the most important subtrees based on the total voting weights they receive*. In this paper ,we develop a number of strategies to this end.

For identifying useful subtrees, we propose using the k-means clustering method (e.g. cluster the contributions into "large" and "small" if $k = 2$).

For selecting important subtrees, we propose (1) a threshold that is proportional to the total voting weight of all the rules, (2) k-means clustering, and (3) top-k strategies.

Table 1: The algorithms to be compared experimentally.

Algorithm	Useful subtree identification	Subtree selection
CCGPHH [4]	–	–
T-0.5TVW [1]	fixed threshold of 0.001	0.5 × total voting weight
2C-2C	2-means clustering	2-means clustering
2C-3C	2-means clustering	3-means clustering
2C-Top1	2-means clustering	top-1
2C-0.25TVW	2-means clustering	0.25 × total voting weight
3C-2C	3-means clustering	2-means clustering
3C-3C	3-means clustering	3-means clustering
3C-Top1	3-means clustering	top-1
3C-0.25TVW	3-means clustering	0.25 × total voting weight

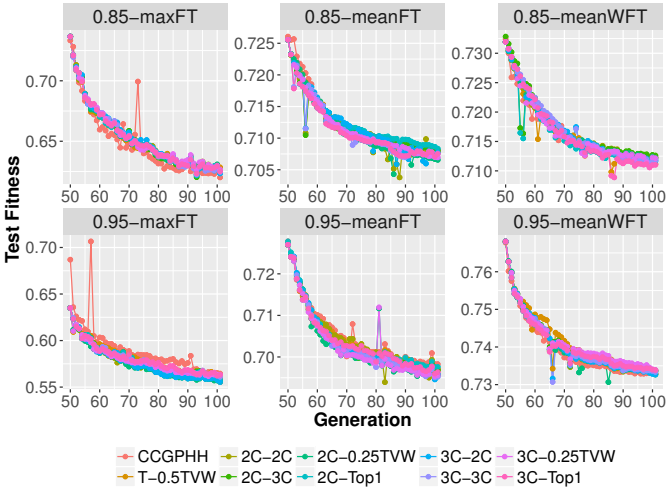

Figure 1: Convergence curves of the test performance of the compared CCGP algorithms.

As a result, we propose a series of algorithms, which are described in Table 1, along with the existing algorithms CCGP [4] and T-0.5TVW [1].

3 EXPERIMENTAL STUDIES

We evaluate the effectiveness of the algorithms in Table 1 on three objectives and two utilisation levels. The three objectives are (1) maximal flowtime, (2) mean flowtime and (3) mean weighted flowtime. The two utilisation levels are 0.85 and 0.95. The job shop configuration and GP parameter settings are the same as in [4]. For each scenario, the compared algorithms were run 30 times independently.

Test Performance. Fig. 1 shows the convergence curves of the compared CCGP algorithms in the 6 scenarios after the feature construction stage. In each generation, the point is is the average test fitness in that generation over 30 independent runs.

From the figure, one can see that the compared algorithms have almost the same convergence curves. We have also conducted Wilcoxon's rank sum test, and found no significant difference between the final results of the algorithms under $\alpha = 0.05$. However, it is observed that some CCGP-FCs (e.g. 3C-Top1) seemed to produce the smoother convergence curves than CCGPHH.

Table 2: Number of subtrees with and without filtering, for the sequencing and routing rules in the 6 test scenarios.

Objective	Util Level	Sequencing		Routing	
		No Filter	Filter	No Filter	Filter
Mean FT	0.85	50.93±8.87	25.17±6.58	55.60±9.49	28.47±5.59
	0.95	50.60±9.23	24.97±6.04	58.17±7.90	29.33±5.69
Max FT	0.85	49.23±11.13	25.20±6.16	65.23±14.48	34.10±7.44
	0.95	54.60±10.99	27.00±6.30	61.80±9.12	30.47±5.78
Mean WFT	0.85	40.97±9.06	19.53±5.54	52.67±10.54	25.60±6.07
	0.95	42.43±8.11	20.67±4.45	51.23±9.26	26.23±5.33

Effectiveness of Filtering. Table 2 shows the number of subtrees with and without the filtering in the 6 scenarios. From the table, it can be seen that the filtering method can remove about half of the subtrees for both the sequencing and routing rules. In other words, the filtering method leads to a dramatic improvement in efficiency of the feature construction.

Constructed Features. A constructed feature for the sequencing rule is `PT/W`. This high-level feature is equivalent to the weighted shortest processing time (WSPT) rule, which has shown effectiveness in minimising mean weighted flowtime.

A constructed feature for the routing rule is `PT * WIQ`. It is essentially the same as `WIQ`, since `PT` is the same for all the candidate machines. In other words, the proposed feature construction methods are not accurate enough, and there may still be noise in the extended terminal sets. As a result, the CCGP-FC methods did not manage to outperform CCGP without FC significantly.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel GPHH with cooperative coevolution and feature construction (CCGP-FC) for evolving both sequencing and routing rules of dynamic FJSS. Extending from the feature selection method in [1], we designed a number of FC mechanisms. The experimental results showed that although the proposed FC methods did not manage to improve the results, they improved the stability of the evolutionary process.

In the future, we will further improve the FC accuracy and reduce the noise in the terminal set. We will also consider combining feature selection and feature construction to obtain a more compact terminal set.

REFERENCES

[1] Y. Mei, S. Nguyen, B. Xue, and M. 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.

[2] Yi Mei, Su Nguyen, and Mengjie Zhang. 2017. Constrained Dimensionally Aware Genetic Programming for Evolving Interpretable Dispatching Rules in Dynamic Job Shop Scheduling. In *Asia-Pacific Conference on Simulated Evolution and Learning*. Springer, 435–447.

[3] J. Tay and N. Ho. 2008. Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering* 54, 3 (2008), 453–473.

[4] 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.