A Preliminary Approach to Evolutionary Multitasking for Dynamic Flexible Job Shop Scheduling via Genetic Programming

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

Su Nguyen
Centre for Data Analytics and Cognition, La Trobe University, Melbourne, Australia
P.Nguyen4@latrobe.edu.au

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

ABSTRACT

Genetic programming, as a hyper-heuristic approach, has been successfully used to evolve scheduling heuristics for job shop scheduling. However, the environments of job shops vary in configurations, and the scheduling heuristic for each job shop is normally trained independently, which leads to low efficiency for solving multiple job shop scheduling problems. This paper introduces the idea of multitasking to genetic programming to improve the efficiency of solving multiple dynamic flexible job shop scheduling problems with scheduling heuristics. It is realised by the proposed evolutionary framework and knowledge transfer mechanism for genetic programming to train scheduling heuristics for different tasks simultaneously. The results show that the proposed algorithm can dramatically reduce the training time for solving multiple dynamic flexible job shop tasks.

CCS CONCEPTS

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

KEYWORDS

Evolutionary Multitasking, Knowledge Transfer, Genetic Programming Hyper-heuristics, Dynamic Flexible Job Shop Scheduling.

ACM Reference Format:
Fangfang Zhang, Yi Mei, Su Nguyen, and Mengjie Zhang. 2020. A Preliminary Approach to Evolutionary Multitasking for Dynamic Flexible Job Shop Scheduling via Genetic Programming. In *Genetic and Evolutionary Computation Conference Companion (GECCO '20 Companion), July 8–12, 2020, Cancún, Mexico.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3377929.3389934

1 INTRODUCTION

In flexible job shop scheduling, n jobs $J = \{J_1, J_2, ..., J_n\}$ need to be processed by m machines $M = \{M_1, M_2, ..., M_m\}$. Each job J_j has an arrival time $at(J_j)$ and a sequence of operations $O_j = (O_{j1}, O_{j2}, ..., O_{ji})$. Each operation O_{ji} can only be processed by one of its optional machines $\pi(O_{ji})$ and its processing time $\delta(O_{ji})$ depends on the machine that processes it. In dynamic flexible job shop scheduling (DFJSS) [4], not only the routing decision and sequencing decision need to be made simultaneously, but also dynamic events are necessary to be taken into account when making schedules. This paper focuses on one dynamic event that new jobs arrive at the job shop dynamically and stochastically.

Tree-based genetic programming (GP) [2], as a hyper-heuristic approach, has been successfully applied to evolve scheduling heuristics for JSS *automatically* [3, 5]. However, the training efficiency of GP in solving multiple job shop tasks is still limited. One reason is that the configurations of job shop scenarios vary from one scenario to an other, and the training processes of scheduling heuristics for different tasks are often treated independently under different environments.

Evolutionary multitasking [1] was proposed to address multiple related tasks simultaneously using a single representation and framework. The main feature is that the knowledge from different tasks can be transferred with each other during the evolutionary process. However, they are mostly applied to benchmarks with continuous, numeric optimisation problems rather than discrete, combinatorial optimisation problems. In DFJSS, the different job shop tasks with the same objective but with different configurations can be considered as different but related tasks to be optimised together. This paper introduces the idea from evolutionary multitasking but incorporates with GP as a hyper-heuristic algorithm to evolve scheduling heuristics for the related DFJSS tasks simultaneously. It is a new attempt for evolutionary multitasking to explore tree-based heuristic search space for discrete, combinatorial optimisation problems.

2 THE PROPOSED ALGORITHM

This paper defines the tasks with the same objective but with different utilisation levels as related tasks to be optimised together. A larger utilisation level leads to a more complex scheduling task. The algorithm groups the GP individuals for optimising different tasks by dividing the entire population into several subpopulations. The individuals in the same or different subpopulation are used to optimise the same or different task. Assuming k tasks (i.e. P_1, P_2, ... , P_k) are desired to be solved simultaneously, the population of GP is equally divided into k subpopulations (i.e. $Subpop_1$, $Subpop_2$, ... , $Subpop_k$) and each subpopulation aims to solve the corresponding task only. On one hand, the evolutionary process of each subpopulation is independent, and the individuals in different subpopulations are evolved respectively. On the other hand, different subpopulations assist with each other by sharing their knowledge with others. The output of a GP run consists of k best evolved rules (i.e. ind_1^*, ind_2^*, ... , ind_k^*).

The crossover with the parents from different subpopulations is used to transfer knowledge between different tasks instead of transferring the whole individuals. This paper defines a transfer ratio tr to control the frequency to transfer knowledge from other subpopulations, and simply transfer knowledge at each generation. If knowledge transfer mechanism is triggered ($rand <= tr$), the first parent $parent_1$ will be selected from the current subpopulation, and the other parent $parent_2$ will be selected from one of the other subpopulations. Only the offspring derived from $parent_1$ is kept to the next generation for the current subpopulation. Otherwise ($rand > tr$), two parents will be selected from the current subpopulation to produce two offspring for new subpopulation.

From a knowledge transfer perspective, for a subpopulation with a complex task, introducing knowledge from a simple task can speed up its convergence, since the evolved rules with simple task have good quality easier. For a subpopulation with a simple task, learning knowledge from a complex task can help increase the quality of individuals since the evolved rules with a complex task are more comprehensive. In general, the knowledge transfer is supposed to benefit all of the involved tasks.

3 EXPERIMENTS AND RESULTS

The terminal and function sets for GP, the parameter setting for GP, and the simulation in [4] are adopted. For simplicity, two tasks are solved simultaneously in this paper. The population of GP consists of two subpopulations, and the number of individuals are set to 512 for each subpopulation. This paper sets the transfer ratio tr to 0.6, since it is a reasonable setting according to our preliminary work. For the sake of convenience, Fmax and Fmean indicate max-flowtime and mean-flowtime, respectively. The tasks with the same objective but with different utilisation levels are solved simultaneously.

Two algorithms are compared in this paper. The GP with multi-tree representation [4] (MTGP) algorithm is selected as the baseline algorithm. The proposed algorithm based on multitasking, is named MTMTGP, since it involves *multi-tree representation GP (MTGP), and multitasking GP (MTGP)*. Note that MTGP works with one population with 1024 individuals while MTMTGP operates with two subpopulations with 1024 individuals (i.e. 512 individuals for each subpopulation). In addition, MTGP solves four tasks with four GP runs, while MTMTGP handles four tasks with two GP runs.

"–", "+", and "=" indicate the result is significantly better than, worse than or similar to its counterpart based on wilcoxon rank-sum test with a significance level of 0.05 over 30 independent runs.

3.1 Training Time

The training time (CPU time) is an important criterion to measure the efficiency of an algorithm. Less training time means that the algorithm is more efficient to get a solution for a problem. Table 1 shows that the training time of MTMTGP for solving two tasks in each multitasking scenario is dramatically reduced, and its total training time for two tasks is roughly half of that of MTGP. The main reason is that MTMTGP handles tasks simultaneously rather than independently.

Table 1: The mean (standard deviation) of the training time (in minutes) of MTGP and MTMTGP **for solving four tasks.**

		Task	MTGP	MTMTGP
scenario 1	task1	<Fmax,0.85>	64(9)	65(10)(–)
	task2	<Fmax,0.95>	67(12)	
scenario 2	task1	<Fmean,0.85>	61(11)	60(10)(–)
	task2	<Fmean,0.95>	64(13)	

Table 2: The mean (standard deviation) of the objective values of $\text{MTMTGP}_{0.85}$ **and** $\text{MTMTGP}_{0.95}$ **with and without knowledge transfer on test instances in four DFJSS tasks.**

Task	$\text{MTMTGP}_{0.85}$		$\text{MTMTGP}_{0.95}$	
	without	with	without	with
<Fmax,0.85>	1252.21(44.44)	1235.04(40.51)	/	/
<Fmax,0.95>	/	/	2018.93(93.22)	1965.24(38.45)**(–)**
<Fmean,0.85>	386.26(3.16)	384.67(1.17)**(–)**	/	/
<Fmean,0.95>	/	/	560.74(9.48)	552.39(4.90)**(–)**

3.2 The Effectiveness of Knowledge Transfer

Table 2 shows that the quality of the evolved rules with knowledge transfer are significantly better than its counterpart without knowledge transfer in three out of four tasks. Specifically, the performance of $\text{MTMTGP}_{0.85}$(with) is better than $\text{MTMTGP}_{0.85}$(without) in half tasks, while the performance of $\text{MTMTGP}_{0.95}$(with) is better than $\text{MTMTGP}_{0.95}$(without) in all tasks. This indicates that the knowledge obtained with the task with lower utilisation level is useful for the task with higher utilisation level, and vice versa.

4 CONCLUSIONS

The goal of this paper was to develop an efficient GP hyper-heuristic algorithm based on multitasking optimisation to evolve effective scheduling heuristics for distinct DFJSS tasks simultaneously. The efficiency and effectiveness of the proposed algorithm are verified by comparing the training time and the quality of evolved rules for each task. The preliminary results show that the proposed algorithm MTMTGP can dramatically reduce the computational time of GP, and seem to achieve comparable scheduling heuristics for solving multiple DFJSS tasks. The training time needed for solving the tasks in a multitasking scenario is less than half of its counterpart.

REFERENCES

[1] Abhishek Gupta, Yew-Soon Ong, Liang Feng, and Kay Chen Tan. 2017. Multiobjective Multifactorial Optimization in Evolutionary Multitasking. *IEEE Transactions on Cybernetics* 47, 7 (2017), 1652–1665.
[2] John R Koza and Riccardo Poli. 2005. Genetic programming. In *Search Methodologies*. Springer, 127–164.
[3] Su Nguyen, Mengjie Zhang, Mark Johnston, and Kay Chen Tan. 2014. Genetic Programming for Evolving Due-Date Assignment Models in Job Shop Environments. *Evolutionary Computation* 22, 1 (2014), 105–138.
[4] Fangfang Zhang, Yi Mei, and Mengjie Zhang. 2018. Genetic programming with multi-tree representation for dynamic flexible job shop scheduling. In *Proceedings of the Australasian Joint Conference on Artificial Intelligence*. Springer, 472–484.
[5] 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 *Proceedings of the Genetic and Evolutionary Computation Conference*. IEEE, 347–355.