Surrogate-assisted Genetic Programming 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. Genetic Programming (GP), as a powerful approach, has been widely used for automatically evolving priority rules for solving job shop scheduling problems. However, one of the main drawbacks of GP is the intensive computational requirements. This paper aims at investigating appropriate surrogates for GP to reduce its computational cost without sacrificing its performance in solving dynamic flexible job shop scheduling (DFJSS) problems. Firstly, adaptive surrogate strategy with dynamic fidelities of simulation models are proposed. Secondly, we come up with generation-range-based surrogate strategy in which homogeneous surrogates are used in predefined ranges of generations and heterogeneous surrogates are applied in different ranges of generations. The results show that these two surrogate strategies with GP are efficient regarding the computational costs, which are reduced by 22.9% to 27.2% and 32.6% to 36.0%, respectively. Moreover, the test performance shows that the proposed approaches can obtain rules with at least the similar quality to the rules obtained by the basic GP approach without using surrogates and GP with adaptive surrogates achieves significantly better performance in one out of six scenarios. This paper confirms the potential of using surrogates to solve DFJSS problems efficiently. To the best of our knowledge, this represents the first work of applying surrogate models to DFJSS.

Keywords: Surrogate · Dynamic flexible job shop scheduling · Genetic programming.

1 Introduction

In the basic version of job shop scheduling (JSS) problem, n jobs need to be scheduled on m machines, while trying to minimize the makespan. In this standard, for each job, there is a set of operations which need to be executed in a specific order and each operation can be processed at a specified machine. In essence, the former JSS is based on the assumption that only one machine is able to run a particular operation.

The flexible job shop scheduling (FJSS) problem is an extension to classical JSS problem, which breaks the above restriction through the constraint of resources uniqueness. That is, one operation can be processed on more than one machine, which leads itself to a more complex problem. In order to tackle the

FJSS problem, two decisions, which are a machine-specific decision and a job-specific decision, have to be made. The machine-specific decision is to allocate a ready operation to an appropriate machine while the job-specific decision aims to select one operation as the next to be processed when a machine becomes idle and there is operation in its queue. FJSS is NP-hard [4].

In practice, the environment is usually dynamic and jobs arrive in the job shop over time without prior information. Dynamic job shop scheduling (DJSS) was proposed for considering this situation. Exact optimization approaches such as mathematical programming [22] and heuristic search methods such as tabu search [21] and genetic algorithm [23] are often not suitable for solving DJSS problem due to their lack of ability to reflect in time. On the contrary, dispatching rules, as priority functions, have been widely adopted for solving DJSS problems [2, 8], due to the ability to react in real time. A comprehensive comparison among a large number of dispatching rules can be found in [26].

Dynamic flexible job shop scheduling (DFJSS) considers both the characteristics of FJSS and DJSS problems. DFJSS is more challenging since not only does the specific machine need to be determined but also the processing sequence of operations must be decided simultaneously along with the new arrival jobs over time. Naturally, two kinds of dispatching rules are needed in DFJSS, which are routing rules and sequencing rules, respectively. In this case, the quality of DFJSS schedule depends highly on how well the routing rule and the sequencing rule work together. 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 (two kinds of rules). Briefly speaking, the routing rule will be triggered to decide where to allocate the operation when new jobs arrive or one operation is finished and its subsequent operation becomes a ready operation. When a machine becomes idle and its queue is not empty, the sequencing rule will be triggered to determine which operation in its queue will be chosen to process next. However, dispatching rules are normally manually designed. That is, the design of dispatching rules is domain-dependent and time-consuming. In addition, manually designed dispatching rules are relatively simple and they are normally restricted to some specific assumptions [3, 7, 13] and have difficulties in handling complex practical scenarios [6, 10, 24].

As one of the popular evolutionary computation algorithms, Genetic Programming (GP) [14], especially tree-based GP, has been successfully applied to automatically evolve dispatching rules for job shop scheduling [10, 18, 24]. However, a challenge of using GP is the intensive computational requirements. In addition, simulation model is popular in job shop scheduling to measure the objective value and complex simulations will further increase computational costs. In our research, the higher fidelity of the simulation model, which means the setting is much more closer to the original model, the higher the accuracy. However, it also means that the computational costs are higher.

Surrogate assisted evolutionary computation with efficient computational models, known as surrogates, provides a promising means of handing complex applications. It has been proven to be very effective to reduce the computational cost of evolutionary computation [1, 25, 28]. The challenge is how to design appropriate surrogates with cheaper computational costs that can represent the original simulation models well. To the best of our knowledge, there has not been any work about using surrogates to improve the efficiency of GP to solve DFJSS problems to date.

1.1 Goals

To address the challenge above, this paper aims at reducing the computational cost of GP with surrogates but not at the expense of its performance. In particular, we have the following research objectives.

– Propose adaptive surrogates for GP (ASGP) approach to operate diverse surrogates with different fidelities in the search process.
– Design generation-range-based surrogates for GP (GSGP) that uses homogeneous (heterogeneous) surrogates in the same (different) predefined ranges of generations.
– Verify the effectiveness and efficiency of the proposed algorithms.
– Compare the learning processes of the proposed two algorithms with standard GP without surrogates.

1.2 Organization

The rest of the paper is organized as follows. In Section 2, a brief introduction of DFJSS, GP and surrogate-assisted GP are given. Detailed descriptions of the proposed surrogate strategies are illustrated in Section 3. The design of experiments is shown in Section 4 and results with analyses are provided in Section 5. Finally, Section 6 concludes the paper.

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 job_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 $\pi(O_{ij}) \in M$ with $\delta(O_{ij})$.

3) For each time, each machine can only process one operation.

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

For dynamic job shop scheduling, jobs arrive in the job shop over time and their information can only be known when they arrive. This characteristic is opposite to static job shop scheduling.

2.2 Genetic Programming

In recent years, GP has been widely used to automatically design dispatching rules for solving JSS problems. Tree-based GP is commonly used in many studies [15, 17]. In order to handle the characteristics of the DFJSS problem, GP with Cooperative Coevolution (CCGP) [27], which can evolve routing and sequencing rules simultaneously, is applied in this paper. It is worth mentioning that GP has some key advantages that make it stand out among varieties of evolutionary computation approaches in solving JSS problems.

Firstly, GP makes it possible to allow various heuristics to be presented as computer programs, due to its flexible representations. Secondly, GP, which has powerful search mechanisms, can search in heuristic search space to find optimal or near-optimal scheduling heuristics. Thirdly, GP has the ability of exploring both the structure and corresponding parameters at the same time. That is, GP is a domain-independent approach that can automatically generates computer programs to solve problems. As a population based evolutionary computation technique, the main steps of GP are shown in Algorithm 1.

Algorithm 1: Pseudo-code of GP

```
// Initialization
1 while $N_{ind}$ < Popsize do
2     foreach individual
3         Randomly initialize each individual by ramp half-and-half
4 end
5 $gen = 0$
6 while $gen < maxGen$ do
7     Evaluation: Evaluate the individuals by fitness function
8     Selection: Select individuals from population based on fitness value
9     Evolution: Generate new population by applying genetic operators
10        reproduction: copy the selected individuals to new population
11        crossover: create offsprings by swapping chosen subtrees of parents
12        mutation: create a offspring by mutating a chosen part of a parent
13    $gen = gen + 1$
14 end
15 return best individual
```

2.3 Surrogate-assisted Genetic Programming

Surrogate-assisted evolutionary computation applies efficient computational models, often known as surrogates. It was mainly motivated from reducing computational costs in evolutionary optimization of expensive problems, such as aerodynamic design optimization drug design [5] and [12], where complex computational simulations are involved. In essence, for surrogates, the key idea is to use simple, efficient and effective approximation to reflect original problems, thus to reduce the computational costs possibly without sacrificing the performance.

There are some surrogate models that have been proposed to reduce the computational costs of GP for JSS. Hildebrandt and Branke [9] proposed using a phenotypic characterization instead of genotype directly as input to the surrogate model. Nguyen investigated the performance of surrogate-assisted GP in 2014 [19] and proposed a hybrid GP algorithm with multi-fidelity fitness evaluations, which are full evaluation, lazy evaluation and evaluation with simplified model in 2018 [17]. However, these studies are all related to DJSS.

There are three major kinds of approximation, which are problem approximation, function approximation and evolutionary approximation [11]. Problem approximation tries to use approximate simple problem instead of the original problem. In function approximation, the fitness function is replaced by another approximate function. In evolutionary approximation, fitness evaluation can be estimated according to the fitness value of other individuals.

Our research is simulation-based and normally complex simulation models will lead to more complicated situations. In this paper, problem approximation is applied due to its simplicity and the characteristic of DFJSS problem.

3 The Proposed Surrogate Strategies

For surrogates, a common assumption is that higher fidelity models are generally more accurate at the expense of higher computational costs. In our research, the fidelity of models depends on the complexity of simulation models and the number of jobs is the key factor in determining their complexities. The simulation models with more jobs will get higher fidelities, but also suffer expensive computational costs.

It is worth noticing that approximation error in the surrogates does not always harm. Surrogates may contribute more to the evolutionary search than the original models because of the capable of smoothing the multimodal or noisy landscape of the complex problem.

Naturally, it is better to use simple surrogates in the early stage of the optimization and increase the quality of the surrogate as the search proceeds. The use of dynamic surrogates is able to improve the search speed. Our preliminary work shows that the HalfShop surrogate (set the number of jobs and machines to half of the corresponding value in original model) proposed in [20] which is used for DJSS, is not applicable for DFJSS. No prior knowledge is available to select appropriate surrogates for DFJSS problem. So, the key issue is how to design appropriate surrogates reasonably. Based on the assumption that the fidelity of the surrogate and the performance are positively correlated, two kinds of strategies are proposed in this paper.

3.1 Adaptive Surrogates

In this section, adaptive surrogates are proposed for genetic programming and the corresponding algorithm is named as ASGP. The basic idea is to deliberately enlarge accuracy of the surrogate models by building up a very simple surrogates at the early stage. As the evolutionary optimization proceeds, the accuracy of the surrogates increases gradually and smoothly expecting that the performance of approximated surrogate models is consistent with the original model.

Let N_{job} and N_{warmup} represent the number of jobs and warmup jobs, respectively. At the ith generation, the number of jobs and warmup jobs in the simulations during the fitness evaluation are denoted as $N_{job,i}$ and $N_{warmup,i}$. The expression of $N_{job,i}$ and $N_{warmup,i}$ are shown as Eq. (1) and Eq. (2), respectively.

$$N_{job,i} = \begin{cases} N_{job} * \frac{1}{maxGen-1} & gen = 0 \\ N_{job} * \frac{Gen}{maxGen-1} & 1 \leq Gen < maxGen \end{cases} \tag{1}$$

$$N_{warmup,i} = \begin{cases} N_{warmup} * \frac{1}{maxGen-1} & gen = 0 \\ N_{warmup} * \frac{Gen}{maxGen-1} & 1 \leq Gen < maxGen \end{cases} \tag{2}$$

In this way, the number of jobs will increase linearly. Thus, adaptive surrogates with multi-fidelity models, which range from low-fidelity to high-fidelity detailed surrogates as the generation increases, are created.

3.2 Generation-range-based Surrogates

For the ASGP, at each generation, different surrogate models are applied. In this section, generation-range-based surrogates is proposed for genetic programming (GSGP) to explore whether a fixed interval change can be more efficient.

In this paper, the number of jobs and warmup jobs of the original simulation model are set to 5000 and 1000, respectively. We set every ten generations into a range. In each range, the generations will share the same surrogate while in different ranges, the generations will use different surrogates. The setting details of different surrogates used in different generations are shown in Table 1.

Table 1. The setting of generation-range-based surrogates.

Generation ranges	$N_{job,i}$	$N_{warmup,i}$
$[0, 10)$	500	100
$[10, 20)$	1000	200
$[20, 30)$	1500	300
$[30, 40)$	2500	500
$[40, 50)$	5000	1000

4 Experiment Design

In this paper, we present the results of our experiments obtained by the three involved algorithms using three commonly used objectives, namely: (1) max-flowtime, (2) mean-flowtime, and (3) mean-weighted-flowtime. As mentioned earlier, the proposed algorithms are compared with CCGP [27] to verify its effectiveness and efficiency.

In order to test the effectiveness and robustness of proposed algorithms, six simulation scenarios based on three objectives and two utilization levels ($3 * 2$) are involved. The goal of the proposed algorithms is to evolve dispatching rules in different scenarios more efficient without sacrificing their performance in solving DFJSS problems. Because of the lack of the best known objective value of the instances, benchmark routing and sequencing rules, which have been proven that can get better performance in specific objectives than the counterparts [8], are chosen. The details are shown in table 2.

Table 2. The benchmark rules used for different objectives.

Rule type	Objectives	Rule	Description
Routing rule	max-flowtime mean-flowtime mean-weighted-flowtime	LWIQ	Least Work in Queue
Sequencing rule	max-flowtime	FCFS	First Come First Serve
	mean-flowtime	SPT	Shortest Processing Time
	mean-weighted-flowtime	FCFS	First Come First Serve

In the training (test) process, the training (test) performance is the quality of evolved rules. The relative performance ratio, which was defined as the average normalized objective value (f) obtained by evolved rules over the value obtained by benchmark rules, is used to indicate the quality of rules. The equation of normalized objective value is shown as Eq. (3). For the results, the smaller the value, the better.

$$f = \frac{Obj(S(p_r, p_s, I_{train/test}))}{Obj^*(I_{train/test})} * \frac{1}{|I_{train/test}|} \tag{3}$$

In Eq. (3), the evolved routing rule and sequencing rule are denoted by p_r and p_s. The training or test set is represented by $I_{train/test}$. The number of simulations used for each objective in training or test process is represented by $|I_{train/test}|$. $S(p_r, p_s, I_{train})$ is the obtained schedule. The normalized objective value obtained by evolved rules and benchmark rules are denoted by $Obj(S(p_r, p_s, I_{train/test})$ and $Obj^*(I_{train/test})$.

4.1 Parameter Settings

In our experiment, the terminal set and function set in [16] are adopted. The details are shown in table 3. It is worth mentioned that "/" is the protected division that returns the largest double positive number if divided by 0.

Table 3. The terminal and function sets.

	Symbol	Description
terminals	NIQ	The number of operations in the queue
	WIQ	Current work in the queue
	MWT	Waiting time of a machine
	PT	Processing time of an operation on a specified machine
	NPT	Median processing time for the next operation
	OWT	The waiting time of an operation
	WKR	Median amount of work remaining for a job
	NOR	The number of operations remaining for a job
	W	Weight of a job
	TIS	Time in system
functions		$+, -, *, /, max, min$

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. In addition, 30 runs are executed, which assures that the results represent the average behavior instead of extreme situations. The learning process continues until the generation reaches the maximum number of generations, which is set to 51.

4.2 Simulation Configuration

For dynamic simulation, commonly used configuration is adopted [9, 16]. In the job shop, there are ten machines, which has been proven to be a good showcase for job shop environment. 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. Two utilization levels are 0.85 and 0.95. It is noted that, in order to improve the generalisation ability of the evolved rules for DFJSS problems, the seeds used to stochastically generate the jobs are rotated in the training process at each generation.

5 Results and Analyses

The surrogate-assisted GP in evolutionary search with respect to the test performance, the training time and the learning process of three algorithms are investigated and discussed. The $(-, +)$ marks show whether our proposed approaches converge significantly better or poorer that the basic approach in Wilcoxon rank sum test ($p \leq 0.05$), respectively. For the convenience of description, $< obj, uti >$ indicates the simulation scenarios, where obj and uti are the objective and the utilization level.

5.1 Test Performance of Evolved Rules

Table 4 shows the mean and deviation error of ASGP, GSGP and CCGP. Overall, ASGP and GSGP algorithms are no significantly worse than CCGP. The mean value obtained by ASGP are about equal with the value obtained by CCGP in all scenarios. It is noted that ASGP significantly outperforms CCGP in scenario $< tmean, 0.85 >$. This clearly shows the potential of using surrogates to improve the performance of GP. It also indicates that the surrogates (approximation models) may not be always harm.

Table 4. The mean and deviation error of the compared algorithms over 30 independent runs for six scenarios.

Index	**Scenario**	**ASGP**	**GSGP**	**CCGP**
1	$< tmax, 0.85 >$	0.640(0.034)	0.638(0.029)	0.642(0.035)
2	$< tmax, 0.95 >$	0.571(0.023)	0.565(0.018)	0.568(0.030)
3	$< tmean, 0.85 >$	0.772(0.012)**(-)**	0.768(0.008)	0.772(0.015)
4	$< tmean, 0.95 >$	0.734(0.023)	0.738(0.022)	0.731(0.015)
5	$< twt, 0.85 >$	0.778(0.030)	0.772(0.010)	0.774(0.018)
6	$< twt, 0.95 >$	0.773(0.023)	0.774(0.024)	0.774(0.037)

For GSGP, the mean value obtained are slightly smaller than CCGP in four out of six scenarios. In addition, the variances obtained by GSGP are smaller than CCGP in five out of six scenarios.

5.2 Training Time

Table 5 shows the computational costs (reductions produced by surrogates compared with CCGP) of the three algorithms. Overall, ASGP and GSGP need less computational requirements compared with CCGP. The average reductions produced by ASGP and GSGP are 25.7% and 34.4%, respectively.

The experimental results have confirmed that ASGP can reduce the computational costs by at least 22.9% in six scenarios. In both scenario 2 and scenario 3, the computational costs are reduced the most (27.2%). For GSGP, it is obvious that it can reduce more computational costs (from 32.6% to 36.0%) than ASGP (from 22.9% to 27.2%). It is not surprising because the average fidelity of ASGP is higher than GSGP. In addition, the computational costs are reduced the most (36.0%) in scenario 1 while the least (32.6%) in scenario 5.

Table 5. The training time (reduction) of the compared algorithms over 30 independent runs for six scenarios.

Index	**Scenario**	**Training Time (seconds)**		
		ASGP	**GSGP**	**CCGP**
1	$< tmax, 0.85 >$	3399.8 (26.8%)	2969.9 (36.0%)	4642.8
2	$< tmax, 0.95 >$	3743.6 (27.2%)	3326.2 (35.3%)	5144.9
3	$< tmean, 0.85 >$	3302.5 (27.2%)	2935.0 (35.3%)	4538.5
4	$< tmean, 0.95 >$	3635.3 (25.0%)	3220.2 (33.6%)	4849.9
5	$< twt, 0.85 >$	3436.2 (22.9%)	3004.4 (32.6%)	4458.4
6	$< twt, 0.95 >$	3725.7 (24.8%)	3282.7 (33.8%)	4957.0

5.3 Insight the Learning Process

Fig. 1 shows the convergence curves of evolutionary processes of ASGP, GSGP and CCGP in the training process. The lines in Fig. 1 are the average normalized objective value from 30 independent runs. Although all GP methods start with the same population, the starting points are different because they use different surrogates. That is, CCGP get the value from surrogates with higher fidelities while ASGP and GSGP get the value from surrogates with lower fidelities.

It is noted that both ASGP and GSGP have higher fluctuations in all scenarios than CCGP, especially at the early stage of evolutionary process. For ASGP, the fidelities of surrogate models change more smoothly to handle the learning process gradually. It is expected to meet the need of training. It is interesting that Fig. 1 shows that ASGP and GSGP have basically the same trends in six scenarios. This indicates that the predefined ranges and settings of simulations in GSGP are representative for the learning process. In addition, after generation 40, ASGP and GSGP can achieve almost the same learning ability as CCGP, although they use surrogates with lower fidelities at previous generations.

Fig. 2 shows the convergence curves of the normalized objective value obtained by ASGP, GSGP and CCGP. It is obvious that CCGP can improve much faster at the beginning of the evolution in six scenarios. This benefits from the precise search with full simulations at the expense of computational costs. However, after generation 10 approximately, the test performance between these three algorithms does not differ obviously.

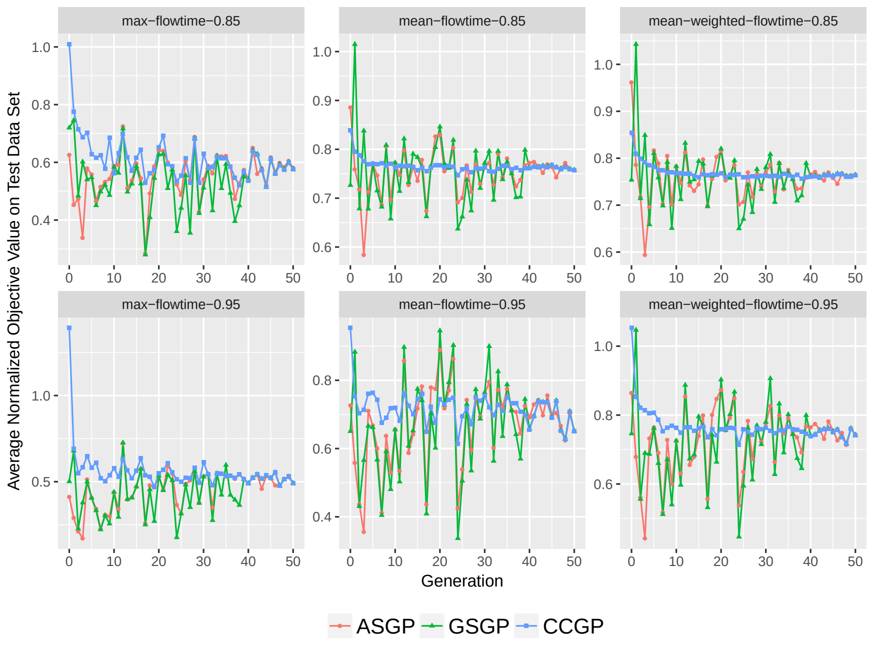

Fig. 1. The convergence curves of the fitness value obtained by ASGP, GSGP and CCGP in the training process.

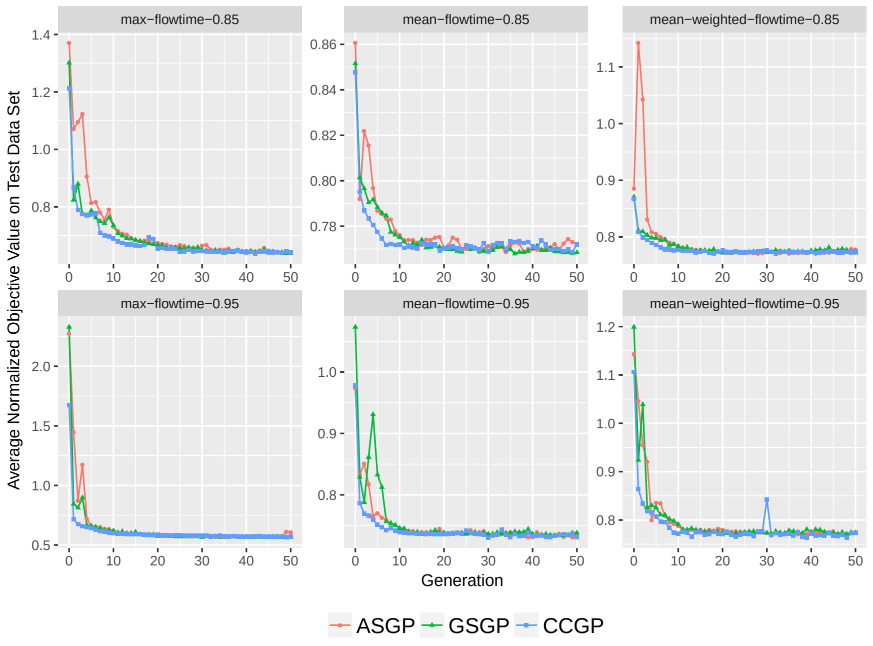

Fig. 2. The convergence curves of the normalized objective value obtained by ASGP, GSGP and CCGP in test process.

Overall, taking the computational cost and test performance into consideration, the proposed algorithms are more promising than CCGP.

6 Conclusions and Future Work

In order to tackle the intensive computational requirements of genetic programming approach, this paper proposed two different kinds of strategies of surrogates for genetic programming to automatically design dispatching rules for dynamic flexible job shop scheduling. It is a preliminary attempt to apply surrogates in the DFJSS problem. The result show that both the adaptive surrogates and the generation-range-based surrogates managed to reduce computational costs without deteriorating the quality of the evolved rules and have the potential to get more promising dispatching rules.

In future research, it is important to further investigate different strategies surrogates to accelerate the effectiveness and efficiency. In this paper, only the level of problem approximation was applied. We will consider function approximation and evolutionary approximation to improve the effectiveness and efficiency of the GP approach.

References

1. Bhattacharya, M.: Reduced computation for evolutionary optimization in noisy environment. In: Proceedings of the 10th annual conference companion on Genetic and evolutionary computation. pp. 2117–2122. ACM (2008)
2. Blackstone, J.H., Phillips, D.T., Hogg, G.L.: A state-of-the-art survey of dispatching rules for manufacturing job shop operations. The International Journal of Production Research **20**(1), 27–45 (1982)
3. Bokhorst, J.A., Nomden, G., Slomp, J.: Performance evaluation of family-based dispatching in small manufacturing cells. International Journal of Production Research **46**(22), 6305–6321 (2008)
4. Brucker, P., Schlie, R.: Job-shop scheduling with multi-purpose machines. Computing **45**(4), 369–375 (1990)
5. Douguet, D.: e-lea3d: a computational-aided drug design web server. Nucleic acids research **38**(suppl_2), W615–W621 (2010)
6. Geiger, C.D., Uzsoy, R.: Learning effective dispatching rules for batch processor scheduling. International Journal of Production Research **46**(6), 1431–1454 (2008)
7. 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)
8. Haupt, R.: A survey of priority rule-based scheduling. Operations-Research-Spektrum **11**(1), 3–16 (1989)
9. Hildebrandt, T., Branke, J.: On using surrogates with genetic programming. Evolutionary computation **23**(3), 343–367 (2015)
10. 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)
11. Jin, Y.: A comprehensive survey of fitness approximation in evolutionary computation. Soft computing **9**(1), 3–12 (2005)

12. Jin, Y., Sendhoff, B.: A systems approach to evolutionary multiobjective structural optimization and beyond. IEEE Computational Intelligence Magazine **4**(3) (2009)
13. Kanet, J.J., Li, X.: A weighted modified due date rule for sequencing to minimize weighted tardiness. Journal of Scheduling **7**(4), 261–276 (2004)
14. Koza, J.R.: Genetic programming as a means for programming computers by natural selection. Statistics and computing **4**(2), 87–112 (1994)
15. 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)
16. 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)
17. 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)
18. 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)
19. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: Selection schemes in surrogate-assisted genetic programming for job shop scheduling. In: Asia-Pacific Conference on Simulated Evolution and Learning. pp. 656–667. Springer (2014)
20. Nguyen, S., Zhang, M., Tan, K.C.: Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules. IEEE transactions on cybernetics **47**(9), 2951–2965 (2017)
21. Nowicki, E., Smutnicki, C.: A fast taboo search algorithm for the job shop problem. Management science **42**(6), 797–813 (1996)
22. Papadimitriou, C.H., Steiglitz, K.: Combinatorial optimization: algorithms and complexity. Courier Corporation (1998)
23. Pezzella, F., Morganti, G., Ciaschetti, G.: A genetic algorithm for the flexible job-shop scheduling problem. Computers & Operations Research **35**(10), 3202–3212 (2008)
24. 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)
25. Ratle, A.: Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In: International Conference on Parallel Problem Solving from Nature. pp. 87–96. Springer (1998)
26. Sels, V., Gheysen, N., Vanhoucke, M.: A comparison of priority rules for the job shop scheduling problem under different flow time-and tardiness-related objective functions. International Journal of Production Research **50**(15), 4255–4270 (2012)
27. 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)
28. Zhou, Z., Ong, Y.S., Nair, P.B., Keane, A.J., Lum, K.Y.: Combining global and local surrogate models to accelerate evolutionary optimization. IEEE Transactions On Systems, Man and Cybernetics-Part C **37**(1), 66–76 (2007)