Phenotype Based Surrogate-Assisted Multi-objective Genetic Programming with Brood Recombination for Dynamic Flexible Job Shop Scheduling

Fangfang Zhang[1]✉, Yi Mei[1], Su Nguyen[2], Mengjie Zhang[1]
[1] School of Engineering and Computer Science, Victoria University of Wellington, New Zealand
{fangfang.zhang, yi.mei, mengjie.zhang}@ecs.vuw.ac.nz
[2] Centre for Data Analytics and Cognition, La Trobe University, Victoria 3086, Melbourne, Australia
P.Nguyen4@latrobe.edu.au

2022 IEEE Symposium Series on Computational Intelligence (SSCI) | 978-1-6654-8768-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/SSCI51031.2022.10022169

***Abstract*—Dynamic flexible job shop scheduling (DFJSS) is an important combinatorial optimisation problem with a large number of real-world applications such as component production in manufacturing. Genetic programming (GP), as a hyper-heuristic approach, has been widely used to learn scheduling heuristics for DFJSS. Brood recombination has been shown its effectiveness to improve the performance of GP by generating more offspring and preselecting only promising individuals into the next generation. However, evaluating more individuals requires more computational cost. Phenotype based surrogate models have been successfully used with GP to speed up the evaluation in single-objective dynamic job shop scheduling. However, its effectiveness on multi-objective dynamic job shop scheduling is unknown. To fill this gap, this paper proposes a novel surrogate-assisted multi-objective GP based on the phenotype of GP individuals for DFJSS. Specifically, we first use phenotypic vector to represent the behaviour of GP individuals in DFJSS. Second, K-nearest neighbour based surrogates are built according to the phenotypic characterisations and multiple fitness values of the evaluated individuals. Last, the built surrogate models are used to predict the fitness of newly generated offspring in GP with brood recombination. The results show that with the same training time, the proposed algorithm can achieve significantly better scheduling heuristics than the compared algorithm. The analyses of population diversity, feature importance, and the number of non-dominated individuals have also shown the effectiveness of the proposed algorithm in different aspects.**

***Index Terms*—Surrogate, Phenotype, Genetic Programming, Brood Recombination, Dynamic Flexible Job Shop Scheduling.**

I. Introduction

Job shop scheduling is the process of organising the production of jobs, i.e., products or services, with limited resources such as machines in manufacturing [1] and gates in airports [2]. In traditional job shop scheduling, a set of machines need to process a number of jobs. Each job consists of a number of operations, and each operation can be processed by a predefined machine. Dynamic flexible job shop scheduling (DFJSS), as a variation of job shop scheduling, is closer to real-world applications, where each operation can be processed by more than one machine. This indicates that we need to make two decisions simultaneously in DFJSS, i.e., machine assignment when a new job comes to the job shop, and operation sequencing when a machine becomes idle and there are operations in its queue. Furthermore, the objectives of DFJSS vary with different requirements, for example, one might want to minimise max-flowtime to deliver products to customers on time [3], while one may prefer to minimise mean-flowtime to reduce the total production cost. It is not uncommon that these objectives are potentially or partially conflicting in DFJSS such as minimising max-flowtime and mean-flowtime. This is naturally a multi-objective problem.

Exact methods such as dynamic programming [4], aim to solve an optimisation problem to optimality. However, they are not efficient and are only applicable to static and small scale problems. Heuristic methods such as genetic algorithms [5], can handle large scale problems well. However, they are not efficient to react to dynamic events since they face rescheduling problems. Scheduling heuristics [6], [7] such as shortest processing time, as priority functions, have been widely used to make schedules by prioritising the machines or operations for job shop scheduling. However, they are normally manually designed, which needs expertise which is not always available. In addition, the designed scheduling heuristics are normally specific to certain scenarios, and we need to design new scheduling heuristics for other scenarios. Genetic programming (GP) [8], as a hyper-heuristic approach [9]–[12], has been popularly used to learn scheduling heuristics automatically due to its flexible representation. In other words, we do not need to define the structures of the scheduling heuristics in advance which are often unknown. GP can automatically learn the structure and the coefficients to form scheduling heuristics. In DFJSS, a scheduling heuristic consists of a routing rule for machine assignment, and a sequencing rule for operation sequencing. These two rules are needed to be learned simultaneously due to their close collaboration in making decisions in DFJSS [13].

Brood recombination aims to improve the effectiveness of GP by generating more offspring (e.g., $> popsize$, population size), and selects only promising individuals to the next generation [14], [15]. However, generating more offspring means the requirement of more computational resources due to the extra individual evaluations. Surrogate techniques have been widely used in evolutionary computation algorithms to

978-1-6654-8768-9/22/$31.00 ©2022 IEEE

reduce their training time. However, commonly used surrogate techniques such as radial basis function [16] are not applicable for GP in DFJSS where simulation technique is normally used to mimic the dynamic environment of job shop scheduling. What we have for learning surrogate model with GP for DFJSS is a number of GP individuals, i.e., trees (genotype), and their fitness. Phenotypic characterisation was proposed to represent GP individuals by measuring their behaviour, which represents tree-based GP individuals by vectors (phenotype) [17]. In addition, the phenotypic characterisations along with the fitness of individuals were used to build surrogate models with K-nearest neighbour (KNN) to predict the fitness of newly generated offspring by brood recombination mechanism [17]. However, this is only used for optimising single-objective dynamic job shop scheduling problem. It is still not clear whether the phenotype based surrogate technique with KNN can predicate multiple fitness for multi-objective optimisation properly or not. To the best of our knowledge, there is no study on surrogate-assisted GP for multi-objective DFJSS.

The overall goal of this paper is to develop a phenotype based surrogate-assisted multi-objective GP with brood recombination for DFJSS. The proposed algorithm is expected to improve the performance of GP with brood recombination without increasing the computational cost. In particular, this paper has the following research objectives:

- Build phenotype based surrogate model for predicting multiple fitness of GP individuals generated with brood recombination.
- Develop a new surrogate-assisted GP for multi-objective DFJSS along with the built surrogate models via KNN.
- Analyse the effectiveness of the proposed algorithm in terms of the quality of learned scheduling heuristics and training efficiency.
- Investigate the characteristics of the proposed algorithms in terms of population diversity, feature importance, and the number of non-dominated solutions.

II. Background

A. Dynamic Flexible Job Shop Scheduling

For DFJSS, n jobs $\mathcal{J} = \{J_1, J_2, ..., J_n\}$ need to be processed by m machines $\mathcal{M} = \{M_1, M_2, ..., M_m\}$. Each job J_j has 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 $M(O_{ji}) \subseteq \pi(O_{ji})$ and its processing time $\delta(O_{ji}, M(O_{ji}))$ depends on the machine that processes it. This paper focuses on one dynamic event (i.e., continuously arriving new jobs) that the information of a job is unknown until it arrives. Below are the objectives to be optimised in this paper, where Fmax and Fmean, and Tmax and Tmean are two pairs of conflicting objectives [18].

- Max-flowtime (Fmax): $max_{j=1}^{n}\{C_j - r_j\}$
- Mean-flowtime (Fmean): $\frac{1}{n}\sum_{j=1}^{n}\{C_j - r_j\}$
- Max-tardiness (Tmax): $max_{j=1}^{n} max\{0, C_j - d_j\}$

TABLE I
AN EXAMPLE OF USING A SEQUENCING RULE TO CHOOSE AN OPERATION TO BE PROCESSED NEXT.

Operations	PT	W	Priority Value (PT/W)
O_{11}	15	1	15
O_{22}	40	4	10
O_{31}	80	4	20

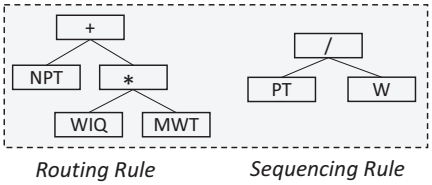

Fig. 1. An example of GP individual for DFJSS.

- Mean-tardiness (Tmean): $\frac{1}{n}\sum_{j=1}^{n} max\{0, C_j - d_j\}$

where C_j is the completion time of a job J_j, r_j is the release time of J_j, and d_j is the due date of J_j.

B. Genetic Programming for DFJSS

GP has been widely and successfully used to learn scheduling heuristics for DFJSS [19]. GP starts with a randomly initialised population with a number of individuals, i.e., *initialisation*. To know the quality of individuals, all the individuals are evaluated with a DFJSS simulation to get their fitness, i.e., *evaluation*. If the stopping criterion is met, the best learned individual found so far will be reported. Otherwise, GP will generate new individuals to the next generation based on the current population with *parent selection* method and genetic operators, i.e., elites, crossover, mutation, and reproduction. The newly produced individuals will be evaluated in the next generation. In such a way, GP improves scheduling heuristics generation by generation for DFJSS.

In this paper, we use GP with multi-tree representation to learn the routing rule and the sequencing rule simultaneously. An example of a GP individual can be found in Fig. 1, where NPT, WIQ, MWT, PT and W are features that will be introduced later. When a job comes, the routing rule will be used to allocate this operation to a machine. When a machine becomes idle, and there are operations in its queue, the sequencing rule will be used to select one operation to be processed next. Taking the sequencing rule as an example, Table I shows how to choose an operation to be processed next. The learned sequencing rule can be considered as a priority function PT/W. Assume there are three operations, i.e., O_{11}, O_{22}, and O_{31}, the priority values of them are firstly calculated according to PT/W. The operation with the smallest priority values O_{22} is selected to be processed next.

III. Phenotype Based Surrogate-Assisted Multi-objective GP with Brood Recombination

A. Overall of the Proposed Algorithm

Fig. 2 shows the flowchart of the proposed algorithm. The proposed algorithm has three different parts from the tradi-

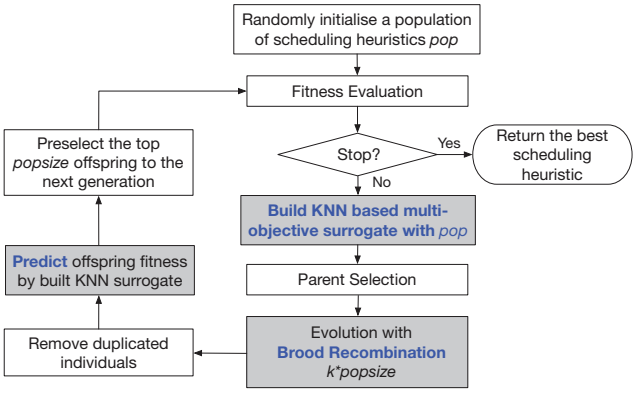

Fig. 2. The flowchart of the proposed algorithm.

Algorithm 1: Build phenotype based multi-objective KNN surrogates

Input : A population with evaluated individuals pop
Output: A KNN based surrogate model
1: set $knn = \phi$
2: **for** i = *1 to* $popsize$ **do**
3: $pc_i \leftarrow$ Calculate the phenotypic characterisation of ind_i
4: $fit_1, fit_2 \leftarrow$ get the fitness of ind_i
5: $knn \leftarrow knn \cup < pc_i, (fit_{i1}, fit_{i2}) >$
6: **end**
7: **return** knn

tional GP. First, after fitness evaluation, a KNN based multi-objective surrogate model is built with the current population pop. This is the key step to the proposed algorithm. Second, during the evolution process, brood recombination is adopted to generate a large number of offspring, i.e., $k*popsize$, where k is the brood size. The duplicated individuals with the same phenotypic characterisations are removed from the population. Third, the built surrogate is used to predict the fitness (multiple) of newly generated individuals. The individuals are ranked based on the estimated fitness by surrogate according to non-dominated sorting and crowding distance as in NSGA-II [20], and only the top $popsize$ individuals are selected to fill the population in the next generation.

B. Phenotype Based KNN Multi-objective Surrogate Building

During the evolutionary process of GP for DFJSS, the available information for building surrogate models is the evaluated tree-based GP individuals and their fitness. This paper borrows the idea in [17] that uses phenotypic characterisation to represent tree-based GP individuals to vectors for building KNN based surrogate. Different from [17] that predicts one fitness for single-objective optimisation, this paper focuses on predicting multiple fitness values for multi-objective optimisation with two conflicting objectives in each scenario.

Algorithm 1 shows how to build phenotype based multi-objective KNN surrogate models. The input is a population with evaluated individuals. The built surrogate consists of a number of samples (line 5) which are represented as a union of $<pc_i, (fit_{i1}, fit_{i2})>$, where pc_i is the phenotypic characterisation of individual ind_i, and fit_{i1} and fit_{i2} are the two fitness of individual ind_i. It is straightforward to get the fitness of individuals. The key is to calculate the phenotypic characterisations of individuals.

TABLE II
AN EXAMPLE OF CALCULATING THE PHENOTYPIC CHARACTERISATION OF A SEQUENCING RULE.

Decision Situations	Reference Rule	PT/W	PC
$1(O_1)$	1	3	
$1(O_2)$	2	2	3
$1(O_3)$	③	1	
$2(O_1)$	3	2	
$2(O_2)$	①	1	1
$2(O_3)$	2	3	
$3(O_1)$	1	2	
$3(O_2)$	2	3	1
$3(O_3)$	③	1	

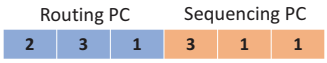

Fig. 3. The flowchart of the proposed algorithm.

1) Phenotypic Characterisation: As mentioned earlier, phenotypic characterisation is a vector used to measure the behaviour of GP individuals. Following the advice in [7], this paper randomly selects 20 routing decision situations and 20 sequencing decision situations with a size of 7 candidates, either 7 machines or operations. For routing/sequencing rule, it will be applied to all the routing/sequencing decision situations, and the ranks of the selected machines/operations in all decision situations are used to form the phenotypic characterisation. Continue with the example sequencing rule shown in Section II-B, Table II shows an example of calculating the phenotypic characterisation of the sequencing rule PT/W. First, we use a reference rule, i.e., WIQ (least workload in the queue) as the routing rule and SPT (shortest processing time) as the sequencing rule, to make decisions in different decision situations. For example, the operations O_1, O_2, O_3 are ranked as 1, 2, and 3 in the first scenario. Second, the examined sequencing rule PT/W is also used to rank the three operations, and it ranks O_3 as the most prior one. Third, we set the rank of the selected operation by reference rule which is 3 as the phenotypic characterisation of PT/W in the first decision situation. Using the same way, we can get the phenotypic characterisation values in all decision situations. The individuals with the same/different behaviour will have the same/different phenotypic characterisation(s). Thus, the individuals can be distinguished.

We calculate the phenotypic characterisation of the routing rule and the sequencing rule separately. Then, we combine the phenotypic characterisations of the routing rule and the sequencing rule as the phenotypic characterisation of a GP individual in DFJSS. Fig. 3 shows an example of the

C. Fitness Prediction

For fitness prediction of newly generated individuals by brood recombination, the phenotypic characterisations of the

individuals are calculated first, as shown in the previous section. Then, for a new individual, we will compare its phenotypic characterisation with all the phenotypic characterisations in the built surrogate model by using Euclidean distance. The fitness of an individual will be set as the same as the fitness of the most similar sample in the surrogate. Assume we have three samples in the built surrogate model for multi-objective optimisation with two objectives, as shown below.

$$knn = \begin{Bmatrix} <2 & 3 & 1 & 3 & 1 & 1>, & (100, 200) \\ <3 & 4 & 2 & 2 & 1 & 7>, & (150, 180) \\ <1 & 5 & 7 & 7 & 1 & 6>, & (200, 150) \end{Bmatrix}$$

If there is a new individual and its phenotypic characterisation is <2, 1, 2, 2, 3, 1>, we calculate the distances of its phenotypic characterisation with all the samples in the built surrogate, which are 3.16, 7.07, and 9.80, respectively. It is the most similar to the first sample, thus, its estimated fitness of two objectives are 100 and 200, respectively.

IV. Experiment Design

A. Simulation Model

This paper assumes that 5000 jobs need to be processed by ten machines [10]. Each job has a number of operations that follows a uniform discrete distribution between one and ten. Each operation has a number of candidate machines, which follows a uniform discrete distribution between one and ten. The processing time of each operation is set by uniform discrete distribution with the range [1, 99].

Different objectives and utilisation levels are used to generate scenarios for examining the algorithms. It is noted that *utilisation level* (p) is an essential factor to simulate different scenario environments. It is the proportion of time that a machine is busy. Jobs arrive continuously in the simulation according to a Poisson process with a rate λ. The expression is shown in Eq. (1). We can see that the utilisation level is managed by adjusting the rate λ in the Poisson process.

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

To improve the generalisation ability of the learned scheduling heuristics for DFJSS problems, the simulation used at each generation is rotated by setting a new random seed. A warm-up period of 1000 jobs is used to improve the accuracy of collected data. We collect data from the following 5000 jobs. The simulation keeps running until the 6000th job is finished.

B. Comparison Design

This paper uses GP with multi-tree representation to learn the routing and sequencing rules simultaneously [13]. The GP algorithm with the mechanism of NSGA-II [20] named NSGP-II is set as the baseline algorithm for comparison [18]. The proposed phenotype based surrogate-assisted multi-objective GP with Brood Recombination is named SA-NSGP-II.

The proposed algorithms are tested in six scenarios. The scenarios are the combinations of two pairs of objectives (i.e., Fmax and Fmean, and Tmax and Tmean) and three utilisation levels (i.e., 0.75, 0.85 and 0.95). The learned best rule is tested on 50 unseen instances, and the average objective value is reported as the test objective value of this best rule.

TABLE III
THE TERMINAL AND FUNCTION SETS.

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

TABLE IV
THE PARAMETER SETTINGS OF NSGP-II AND SA-NSGP-II.

Parameter	Value
Population size	1000
Method for initialising population	ramped-half-and-half
Initial minimum/maximum depth	2 / 6
Maximal depth of programs	8
The number of elites	10
Crossover/Mutation/Reproduction rate	80% / 15% / 5%
Parent selection	Tournament selection with size 7
Number of generations	51
Terminal/non-terminal selection rate	10% / 90%
*brood size k	3

*: only for SA-NSGP-II

C. Parameter Settings

In our experiment, the terminal and function set are shown in Table III, following the setting in [21]. The "/" operator is a protected division, returning one if divided by zero. The other parameter settings of GP are shown in Table IV.

V. Results and Discussions

According to 30 independent runs, we use Wilcoxon rank-sum test with a significance level of 0.05 to verify the performance of the algorithms. In the following results, "↑", "↓", and "≈" indicate the corresponding result is statistically significantly better than, worse than, or similar to the compared one. We use hyper volume (HV) [22] and inverted generational distance (IGD) [23] to examine the proposed algorithms. A larger (smaller) HV (IGD) value signifies better performance. Since the true Pareto font is not available in our problem, we put all obtained best solutions of all algorithms together to get an approximated Pareto font as the true Pareto front for calculating IGD. We will use the term true Pareto front in the following paper.

A. Quality of Learned Scheduling Heuristics

Table V shows the HV and IGD of NSGP-II and SA-NSGP-II on training and test instances over 30 independent runs in six multi-objective scenarios. The results show that SA-NSGP-II is significantly better than NSGP-II on both training and test in all six scenarios. Specifically, for the training and test, the HV (IGD) of SA-NSGP-II is larger (smaller) than NSGP-II in all/most scenarios. We can also see that the proposed

TABLE V
THE MEAN (STANDARD DEVIATION) OF THE HV AND IGD ON **TRAINING INSTANCES AND TEST INSTANCES** OF NSGP-II AND SA-NSGP-II BASED ON 30 INDEPENDENT RUNS IN SIX MULTI-OBJECTIVE SCENARIOS WHICH ARE REPRESENTED BY THE OBJECTIVE AND UTILISATION LEVEL.

		Training				**Test**			
		HV		**IGD**		**HV**		**IGD**	
No.	Scenario	NSGP-II	SA-NSGP-II	NSGP-II	SA-NSGP-II	NSGP-II	SA-NSGP-II	NSGP-II	SA-NSGP-II
1	<Fmax-Fmean, 0.75>	0.82(0.06)	0.88(0.03)(↑)	0.08(0.04)	0.04(0.02)(↑)	0.79(0.07)	0.85(0.04)(↑)	0.10(0.04)	0.06(0.02)(↑)
2	<Fmax-Fmean, 0.85>	0.92(0.02)	0.94(0.01)(↑)	0.03(0.01)	0.02(0.01)(↑)	0.91(0.03)	0.92(0.01)(≈)	0.04(0.02)	0.03(0.01)(↑)
3	<Fmax-Fmean, 0.95>	0.89(0.02)	0.92(0.01)(↑)	0.04(0.01)	0.03(0.01)(↑)	0.65(0.02)	0.69(0.05)(↑)	0.04(0.01)	0.04(0.01)(↑)
4	<Tmax-Tmean, 0.75>	0.83(0.10)	0.89(0.08)(↑)	0.16(0.08)	0.12(0.05)(↑)	0.78(0.12)	0.81(0.09)(≈)	0.28(0.08)	0.21(0.08)(↑)
5	<Tmax-Tmean, 0.85>	0.86(0.06)	0.90(0.02)(↑)	0.08(0.03)	0.06(0.02)(↑)	0.84(0.08)	0.88(0.02)(↑)	0.08(0.05)	0.05(0.01)(↑)
6	<Tmax-Tmean, 0.95>	0.87(0.03)	0.91(0.01)(↑)	0.05(0.02)	0.03(0.01)(↑)	0.86(0.04)	0.90(0.02)(↑)	0.06(0.02)	0.05(0.02)(↑)

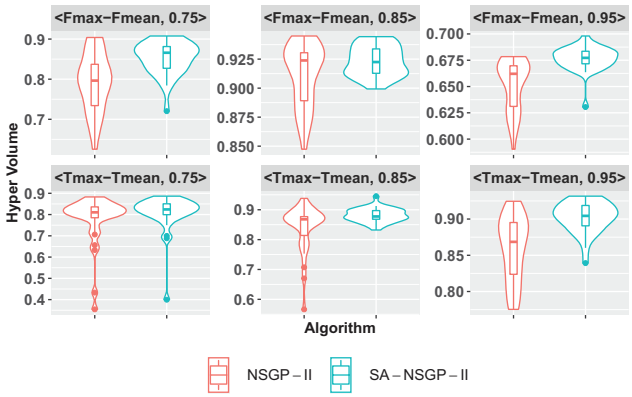

Fig. 4. The violin plots of the **HV** values of NSGP-II and SA-NSGP-II in six test multi-objective scenarios.

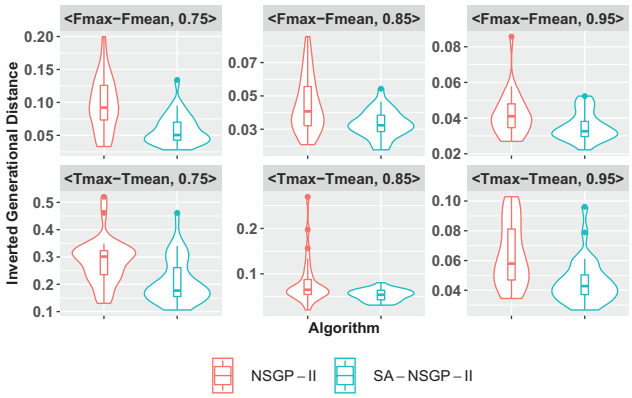

Fig. 5. The violin plots of the **IGD** values of NSGP-II and SA-NSGP-II in six test multi-objective scenarios.

algorithm SA-NSGP-II has good generalisation ability, and its performance on training and test instances is consistent. These findings show the effectiveness of the proposed phenotype based surrogate-assisted multi-objective GP on DFJSS.

Fig. 4 shows the violin plots of the HV values of NSGP-II and SA-NSGP-II in six unseen multi-objective scenarios. From the distributions of the HV values, we can see that SA-NSGP-II can obtain larger values than that of NSGP-II. This indicates that SA-NSGP-II has a better volume of the dominated portion of the objective space, which shows the superiority of SA-NSGP-II from the perspective of the coverage of obtained solutions. Fig. 5 shows the violin plots of the IGD values of NSGP-II and SA-NSGP-II in six unseen multi-objective scenarios. From the distributions of IGD values, we can see that SA-NSGP-II achieves smaller IGD than NSGP-II. This means the obtained solutions of SA-NSGP-II are closer to the true Pareto-front, which shows the effectiveness of SA-NSGP-II from the perspective of the obtained solution quality.

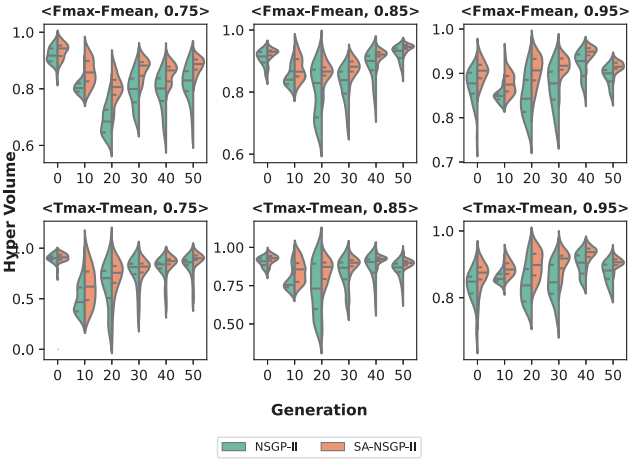

Fig. 6. The violin plots of the HV values of NSGP-II and SA-NSGP-II on the training instances at six generations, i.e., generations 0, 10, 20, 30, 40, and 50, in six multi-objective scenarios.

B. Learned Scheduling Heuristics in the Evolutionary Process

Fig. 6 and Fig. 7 show the violin plots of the HV values of NSGP-II and SA-NSGP-II during the training process in six multi-objective scenarios. We investigate the generations from an early stage to a later stage, i.e., generations 0, 10, 20, 30, 40, and 50, during the evolutionary process. We can see that the proposed algorithm SA-NSGP-II performs better than NSGP-II from an early stage until a later stage from the perspective of HV and IGD. This verifies the effectiveness of SA-NSGP-II for learning competitive scheduling heuristics in DFJSS by investigating the evolutionary process.

C. Learned Pareto Fronts

Fig. 8 shows the learned Pareto fronts with the medium HV value of NSGP-II and SA-NSGP-II in six multi-objective

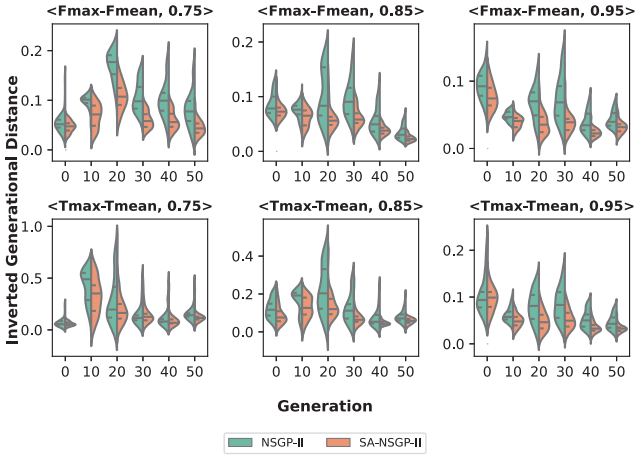

Fig. 7. The violin plots of the IGD values of NSGP-II and SA-NSGP-II on the training instances at six generations, i.e., generations 0, 10, 20, 30, 40, and 50, in six multi-objective scenarios.

TABLE VI
THE MEAN (STANDARD DEVIATION) OF TRAINING TIME (IN MINUTES) OF NSGP-II AND SA-NSGP-II OVER 30 INDEPENDENT RUNS OF SIX MULTI-OBJECTIVE SCENARIOS.

No.	Scenarios	NSGP-II	SA-NSGP-II
1	<Fmax-Fmean, 0.75>	93(11)	95(13)(≈)
2	<Fmax-Fmean, 0.85>	94(14)	97(13)(≈)
3	<Fmax-Fmean, 0.95>	106(17)	108(16)(≈)
4	<Tmax-Tmean, 0.75>	95(20)	96(18)(≈)
5	<Tmax-Tmean, 0.85>	98(16)	100(18)(≈)
6	<Tmax-Tmean, 0.95>	107(16)	109(13)(≈)

scenarios. We can see that SA-NSGP-II obtains better Pareto fronts than NSGP-II in all the examined scenarios. The objective values obtained by SA-NSGP-II are smaller than NSGP-II. In other words, the scheduling heuristics learned by SA-NSGP-II can nominate most of the scheduling heuristics obtained by NSGP-II. This verifies the effectiveness of SA-NSGP-II in terms of the learned Pareto fronts.

D. Training Time

Table VI shows the mean and standard deviations of the training time (in minutes) of NSGP-II and SA-NSGP-II according to 30 independent runs in six multi-objective scenarios. The results show no significant difference in the training time between NSGP-II and SA-NSGP-II. However, SA-NSGP-II can achieve better performance than NSGP-II from an early stage. This shows the effectiveness and efficiency of SA-NSGP-II with the developed surrogates.

Summary: Overall, we can see that the proposed algorithm SA-NSGP-II can successfully handle the extra offspring evaluation problem of GP with brood recombination in DFJSS. This is realised by the proposed phenotype based multi-objective surrogate. No extra computational resources are required, although many individuals are evaluated for brood recombination. Furthermore, with the same training time, SA-NSGP-II can achieve a significant performance than NSGP-II

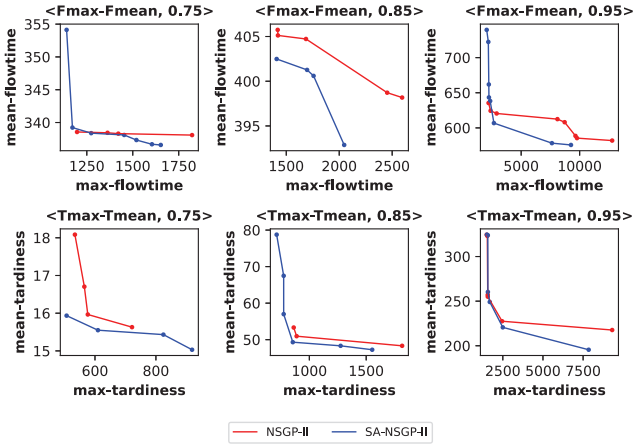

Fig. 8. The learned Pareto fronts of NSGP-II and SA-NSGP-II in six multi-objective scenarios.

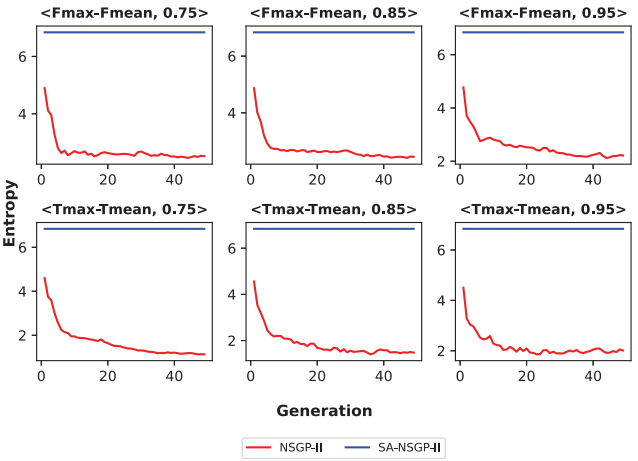

Fig. 9. The curves of entropy values of NSGP-II and SA-NSGP-II over 30 independent runs in six multi-objective scenarios.

from an early stage to a later stage in the evolutionary process. This verifies the effectiveness and efficiency of SA-NSGP-II.

VI. FURTHER ANALYSES

A. Population Diversity

Entropy is used to measure the diversity of individuals of GP in DFJSS. The entropy is calculated as $entropy = -\sum_{c \in C} \frac{|c|}{|inds|} log(\frac{|c|}{|inds|})$, where C is the set of clusters obtained by the DBScan clustering algorithm [24] with the phenotypic distance measure [17] and a cluster radius of zero based on the phenotypic characterisations of individuals in the whole population. A larger entropy value indicates a higher diversity of individuals for a task.

Fig. 9 shows the curves of entropy values of NSGP-II and SA-NSGP-II over 30 independent runs in six multi-objective scenarios. The results show that the diversity of NSGP-II reduces dramatically along with generations. On the contrary, SA-NSGA-II can have a good diversity level from an early stage, and can successfully maintain the population diversity

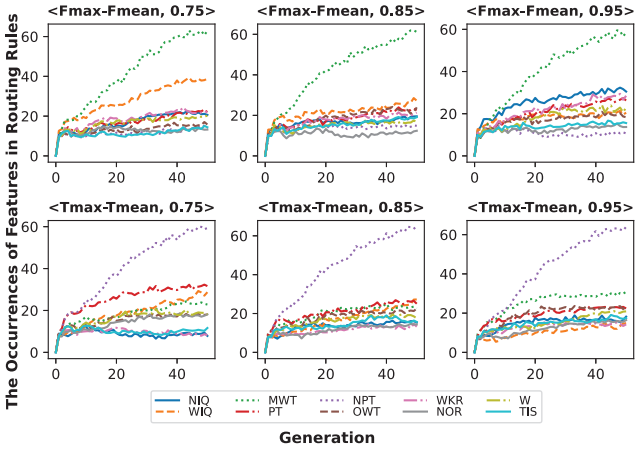

Fig. 10. The curves of the occurrences of features in learned routing rules.

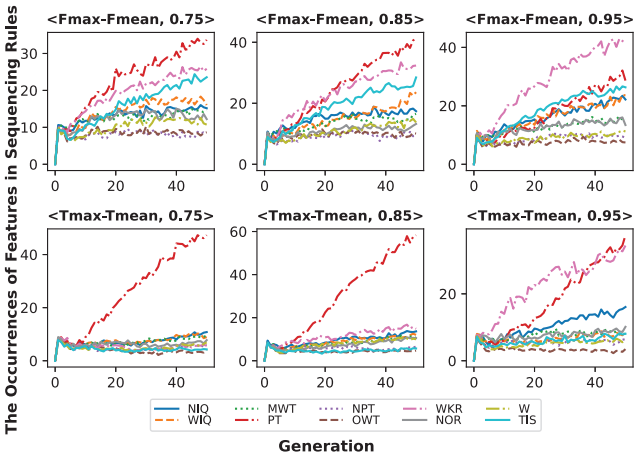

Fig. 11. The curves of the occurrences of features in learned sequencing rules.

during the evolutionary process. This is realised by large number of individuals provided by the brood recombination, and the duplicated individual removing strategy. It is noted that the individuals of SA-NSGP-II at all generations are unique which leads to a similar level of diversity across generations.

B. Occurrences of Features in Learned Scheduling Heuristics

To investigate the feature importance, we select the top 10 individuals at each generation to record the feature occurrences of the routing rule and the sequencing rule. Fig. 10 shows the curves of the occurrences of features in the learned routing rules along with generations. The results show that MWT is the most important feature for the scenarios with objectives Fmax and Fmean, while, NPT is the top feature for scenarios with objectives Tmax and Tmean. During the evolutionary process, GP algorithms can detect important features automatically and use more important features to generate individuals. In general, the feature importance of routing rules is sensitive to the objectives, and there is no big difference between the scenarios with the same objectives but different utilisation levels.

Fig. 11 shows the curves of the occurrences of features in the learned sequencing rules along with generations. The results show that the feature importance differs in different scenarios, either slightly or significantly. For the scenarios with objectives Fmax and Fmean (in the first row), we can see features PT, WKR, and TIS are the top three important features. The feature importance in scenario <Fmax-Fmean, 0.75> is similar to that of in scenario <Fmax-Fmean, 0.85>. However, there is a difference in the feature importance in scenario <Fmax-Fmean, 0.95>. PT is the most important feature in scenarios <Fmax-Fmean, 0.75> and <Fmax-Fmean, 0.85>, while WKR is the most important feature in scenario <Fmax-Fmean, 0.95>. We can see the feature importance is not only sensitive to the objective but also the utilisation level. These findings on feature importance for multi-objective optimisation are different from the observations on single-objective optimisation that the feature importance is sensitive to either the objective or the utilisation level [25]. Interested readers can refer to [25]. In terms of the scenarios with objectives Tmax and Tmean, PT is the most important feature in scenarios <Tmax-Tmean, 0.75> and <Tmax-Tmean, 0.85>, and the feature importance in these two scenarios are quite similar. In addition, PT is much more important than all other features. Besides, different from scenarios <Tmax-Tmean, 0.75> and <Tmax-Tmean, 0.85>, both PT and WKR are important for learning scheduling heuristics in scenario <Tmax-Tmean, 0.95>.

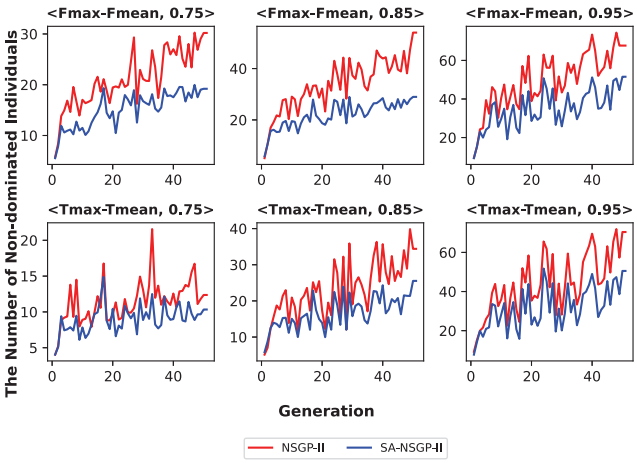

Fig. 12. The curves of the average number of non-dominated individuals of NSGP-II and SA-NSGP-II along with generations over 30 independent runs in six multi-objective scenarios.

C. Number of Non-dominated Individuals

Fig. 12 shows the curves of the average number of non-dominated individuals of NSGP-II and SA-NSGP-II along with generations over 30 independent runs in six multi-objective scenarios. Overall, the multi-objective GP algorithms, i.e., NSGP-II and SA-NSGP-II, learn an increasing number of non-dominated solutions along with generations. This shows the effectiveness of using evolutionary multi-

objective GP to learn scheduling heuristics for DFJSS. The results also show that the number of obtained non-dominated solutions by SA-NSGP-II is smaller than NSGP-II. Considering the good performance of SA-NSGP-II, we can know that SA-NSGP-II learns good and well-distributed solutions.

VII. Conclusions and Future Work

The goal of this paper was to develop an effective phenotype based surrogate-assisted multi-objective GP with brood recombination for DFJSS. The goal has been successfully achieved by proposing effective surrogate models that convert tree-based GP individuals to vectors, and use vectors to predict multiple fitness values of newly generated GP individuals with brood recombination. This is the first time that surrogate technique was designed for multi-objective GP to learn scheduling heuristics in DFJSS.

The results show that with the same training time, the proposed algorithm can achieve significantly better performance than the compared algorithm in terms of the HV and IGD. This is realised by using proposed phenotype based surrogate models to preselect individuals from a larger number of newly generated individuals by GP with brood recombination to the next generation. The effectiveness of SA-NSGP-II was also verified by visualising the HV and IGD trends during the evolutionary process of GP, investigating the learned Pareto fronts and analysing the population diversities in different scenarios. Furthermore, the feature importance for both the routing rule and the sequencing rule is discussed. We find that feature importance in multi-objective optimisation is more complex than single-objective optimisation where the feature importance is not only related to the objective but also the utilisation level. By considering the effectiveness of the proposed algorithm SA-NSGP-II, the investigations on the number of non-dominated solutions indicate that the proposed algorithm SA-NSGP-II achieves good and well-distributed scheduling heuristics for DFJSS. This can provide end-users a set of good scheduling schedules to choose based on their preference.

In the future, we would like to study how the samples in the KNN surrogate models can affect the performance of the algorithm. In addition, we plan to investigate the performance of multi-objective GP algorithms on many-objective optimisation with more than three objectives.

References

[1] J. Zhang, G. Ding, Y. Zou, S. Qin, and J. Fu, "Review of job shop scheduling research and its new perspectives under industry 4.0," *Journal of Intelligent Manufacturing*, vol. 30, no. 4, pp. 1809–1830, 2019.

[2] S. Mokhtarimousavi, D. Talebi, and H. Asgari, "A non-dominated sorting genetic algorithm approach for optimization of multi-objective airport gate assignment problem," *Transportation Research Record*, vol. 2672, no. 23, pp. 59–70, 2018.

[3] 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.

[4] J. A. Gromicho, J. J. Van Hoorn, F. Saldanha-da Gama, and G. T. Timmer, "Solving the job-shop scheduling problem optimally by dynamic programming," *Computers & Operations Research*, vol. 39, no. 12, pp. 2968–2977, 2012.

[5] S. M. Sajadi, A. Alizadeh, M. Zandieh, and F. Tavan, "Robust and stable flexible job shop scheduling with random machine breakdowns: multi-objectives genetic algorithm approach," *International Journal of Mathematics in Operational Research*, vol. 14, no. 2, pp. 268–289, 2019.

[6] M. Durasević and D. Jakobović, "Selection of dispatching rules evolved by genetic programming in dynamic unrelated machines scheduling based on problem characteristics," *Journal of Computational Science*, vol. 61, p. 101649, 2022.

[7] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Instance rotation based surrogate in genetic programming with brood recombination for dynamic job shop scheduling," *IEEE Transactions on Evolutionary Computation*, 2022, Doi: 10.1109/TEVC.2022.3180693.

[8] J. R. Koza and R. Poli, "Genetic programming," in *Search Methodologies*. Springer, 2005, pp. 127–164.

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

[10] F. Zhang, Y. Mei, S. Nguyen, K. C. Tan, and M. Zhang, "Multitask genetic programming based generative hyper-heuristics: A case study in dynamic scheduling," *IEEE Transactions on Cybernetics*, 2021, Doi: 10.1109/TCYB.2021.3065340.

[11] X. Hao, R. Qu, and J. Liu, "A unified framework of graph-based evolutionary multitasking hyper-heuristic," *IEEE Transactions on Evolutionary Computation*, 2020. Doi: 10.1109/TEVC.2020.2991717.

[12] N. Pillay and R. Qu, *Hyper-heuristics: Theory and applications*. Springer, 2018.

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

[14] W. A. Tackett, "Recombination, selection, and the genetic construction of computer programs," Ph.D. dissertation, University of Southern California Los Angeles, 1994.

[15] M. Zhang, X. Gao, and W. Lou, "A new crossover operator in genetic programming for object classification," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 37, no. 5, pp. 1332–1343, 2007.

[16] R. G. Regis, "Particle swarm with radial basis function surrogates for expensive black-box optimization," *Journal of Computational Science*, vol. 5, no. 1, pp. 12–23, 2014.

[17] T. Hildebrandt and J. Branke, "On using surrogates with genetic programming," *Evolutionary Computation*, vol. 23, no. 3, pp. 343–367, 2015.

[18] F. Zhang, Y. Mei, and M. Zhang, "Evolving dispatching rules for multi-objective dynamic flexible job shop scheduling via genetic programming hyper-heuristics," in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2019, pp. 1366–1373.

[19] F. Zhang, S. Nguyen, Y. Mei, and M. Zhang, "Genetic programming for production scheduling: An evolutionary learning approach," in *Machine Learning: Foundations, Methodologies, and Applications*. Springer, 2021. DOI: 10.1007/978-981-16-4859-5, pp. XXXIII+338.

[20] 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.

[21] F. Zhang, Y. Mei, and M. Zhang, "Can stochastic dispatching rules evolved by genetic programming hyper-heuristics help in dynamic flexible job shop scheduling?" in *Proceedings of the IEEE Congress on Evolutionary Computation*. IEEE, 2019, pp. 41–48.

[22] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[23] C. A. C. Coello and M. R. Sierra, "A study of the parallelization of a coevolutionary multi-objective evolutionary algorithm," in *International Conference on Artificial Intelligence*. Springer, 2004, pp. 688–697.

[24] M. Ester, H. P. Kriegel, J. Sander, X. Xu *et al.*, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *kdd*, vol. 96, no. 34, 1996, pp. 226–231.

[25] F. Zhang, Y. Mei, S. Nguyen, and M. Zhang, "Guided subtree selection for genetic operators in genetic programming for dynamic flexible job shop scheduling," in *Proceedings of the European Conference on Genetic Programming*. Springer, 2020, pp. 262–278.