Neural Network Surrogate Based on Binary Classification for Assisting Genetic Programming in Searching Scheduling Heuristic

Ruiqi Chen, Yi Mei, Fangfang Zhang(✉), and Mengjie Zhang

Centre for Data Science and Artificial Intelligence & School of Engineering and Computer Science, Victoria University of Wellington, Wellington 6140, New Zealand
{ruiqi.chen,yi.mei,fangfang.zhang,mengjie.zhang}@ecs.vuw.ac.nz

Abstract. Dynamic job shop scheduling (DJSS) is an NP-hard optimisation problem requiring real-time responses to newly arrived jobs. Scheduling heuristics generated by genetic programming can make high-quality decisions in DJSS, but evaluating these heuristics through simulations is very time-consuming. To speed up evaluations, surrogate models based on machine learning have been developed to predict fitness values. However, existing surrogates are primarily simplistic machine learning models with overly simplified input features, which overlook important characteristics of scheduling heuristics. To enhance prediction accuracy, we propose a new feature representation that comprehensively represents the behaviour of a scheduling heuristic. Additionally, a neural network binary classifier is employed as the surrogate model to learn the complex patterns in the proposed feature representation. Experimental results indicate that the proposed algorithm can find better scheduling heuristics and converge faster compared to the existing algorithms. Further analysis reveals both the new feature representation and the neural network binary classification-based surrogate model enhance the prediction accuracy and contribute to the performance improvement.

Keywords: Dynamic job shop scheduling · Genetic programming · Surrogate · Neural network

1 Introduction

Dynamic job shop scheduling (DJSS) necessitates constant adaptation to frequent job arrivals, requiring immediate dispatch decisions. Genetic Programming (GP) has been widely used for automatically evolving scheduling heuristics for DJSS [4,14,15]. GP treats scheduling rules as individuals that are improved generation by generation through evolutionary principles. The fitness evaluation of these individuals requires time-consuming scheduling simulations. To expedite the evaluation process, various surrogate-assisted GP methods have been developed [6,13]. Instead of running expensive simulations, they use computationally cheaper surrogate models to predict fitness values. Many competitive surrogates are built using machine learning (ML) models [2,8]. However, existing state-of-the-art machine learning surrogates primarily use simplistic models, such as K-Nearest Neighbours (KNN). These models have limited ability to learn the complex relationships in the data. Additionally, the input features are often over-simplified through extensive feature engineering, resulting in the loss of crucial information needed for accurate predictions. Furthermore, existing state-of-the-art surrogates are primarily regression models that directly predict fitness values. Since the GP algorithm changes the scheduling instance each generation (known as instance rotation [8]), the fitness values of individuals across different generations are incomparable. Consequently, incorporating individuals from different generations into the training data is challenging for existing methods, making it difficult to fully utilise all evaluated individuals.

To address the above issues, we propose a new feature representation, the raw-type Phenotypic Characterisation (PC), to comprehensively represent the behaviours of the individual and employ a neural network (NN) as the surrogate model to capture the complex patterns in the raw-type PC. Additionally, the surrogate operates in a binary classification manner by comparing pairs of individuals and predicts the superior one. In this mechanism, evaluated individuals within the same generation are compared to form training data. Training data from different generations are comparable because the relative relationships between pairs of individuals remain stable and are less affected by instance rotation. Consequently, the surrogate model can leverage more training data.

The primary aim of this research is to enhance the effectiveness and efficiency of GP in evolving high-quality scheduling rules through the proposed surrogate. To achieve this aim, the following research objectives are established:

1. Design a new feature representation that comprehensively reflects the behaviour of individuals, thereby assisting surrogate models in making accurate predictions.
2. Develop a surrogate model based on a NN binary classifier to predict the relative relationship between two scheduling rules, which not only effectively learns complex patterns from data but also leverages a larger volume of training data.
3. Conduct experiments to compare the proposed method with both basic GP and existing GP with KNN-based surrogates, analysing prediction accuracy and assessing the impact of data volume on the surrogate model.

2 Literature Review

2.1 Dynamic Job Shop Scheduling

Job shop scheduling is an NP-hard combinatorial optimisation problem that involves determining the processing order of a set of jobs on a set of machines. Each job has a due date and comprises a sequence of operations that must be executed in a predefined order. Each operation is processed on a specific machine for a duration without interruption. A machine can only process one operation at a time. This article focuses on the dynamic scheduling problem, where new

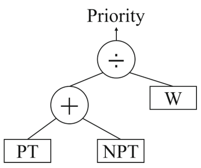


Fig. 1. An example of a scheduling heuristic.

jobs continuously arrive. The information for these jobs is unknown until they arrive [6]. In this paper, we consider minimising three common objectives [3]: mean tardiness (T_{mean}), weighted tardiness (WT_{mean}), and maximum tardiness (T_{max}).

2.2 Genetic Programming for DJSS

Scheduling heuristics enable real-time decision-making for job sequencing in DJSS. GP is a powerful method for evolving high-quality scheduling heuristics, renowned for its flexible representation that accommodates various heuristics within its search space. The tree-based structure is a commonly used representation in GP [6,9]. An example of a tree-based scheduling heuristic is depicted in Fig. 1. This heuristic maps features to priorities, with features gathered at the decision point serving as the terminal nodes (or leaves) of the tree. Functions act as interior nodes, and the priority is returned from the root node. In the example in Fig. 1, the priority of a job is calculated as (PT+NPT)/W, where PT and NPT are the processing time of the current operation and the next operation, respectively, and W is the weight of the job. The job with the highest priority is then selected for processing.

For evolving scheduling heuristics, GP follows a standard evolutionary process, and evaluate each candidate scheduling heuristic by applying it to DJSS decision-making simulations and measuring the steady-state scheduling performance. To this end, sufficiently long simulations are required, which is time consuming.

2.3 Surrogate-Assisted GP for DJSS

Surrogate techniques have been successfully utilised in GP to reduce the intense computational costs associated with evaluations. In the context of DJSS, surrogate models fall into two categories [9]: simplified simulation models and ML models.

Nguyen *et al.* [6] introduced surrogates based on simplified simulation models, reducing the scale of problems (i.e., fewer jobs and machines) to facilitate more cost-effective evaluations. This technique can be used to accelerate tasks such as feature selection for job shop scheduling [5]. Zhang *et al.* [12] further advanced this by proposing an adaptive surrogate strategy that dynamically adjusts the fidelity of simulation models across generations. The multifidelity-based surrogate is also employed to facilitate knowledge transfer through a collaborative mechanism [9].

The KNN surrogate with phenotypic characterisation, first introduced by Hildebrandt and Branke [2], is a highly influential ML surrogate model. This method posits that individuals with similar PC vectors likely exhibit analogous behaviours, thus having similar fitness. The similarity between PC vectors is quantified using their Euclidean distance. Zhang *et al.* [10] expanded this approach to multi-tree GP, evolving both routing and sequencing rules simultaneously for dynamic flexible job shop scheduling problems. Subsequent studies have confirmed the efficacy of KNN surrogate in multi-objective optimisations [7] and the knowledge transfer for multi-task learning [11].

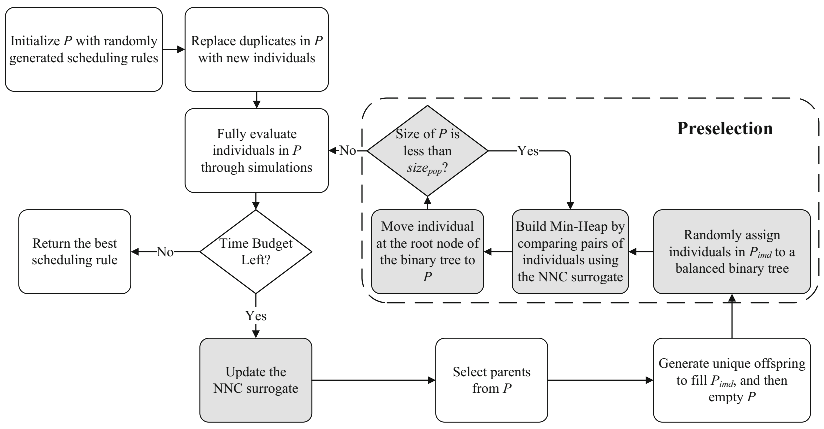

Initialize P with randomly generated scheduling rules
Replace duplicates in P with new individuals
Fully evaluate individuals in P through simulations
No
Size of P is less than size_pop?
Yes
Preselection
Move individual at the root node of the binary tree to P
Build Min-Heap by comparing pairs of individuals using the NNC surrogate
Randomly assign individuals in P_imd to a balanced binary tree
Return the best scheduling rule
No
Time Budget Left?
Yes
Update the NNC surrogate
Select parents from P
Generate unique offspring to fill P_imd, and then empty P

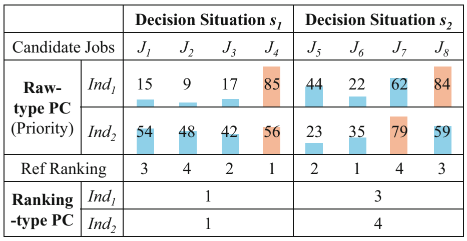
Fig. 2. Flowchart of the SGP-NNC algorithm.

KNN is the most commonly used ML model for building surrogates in the aforementioned studies. However, the KNN surrogate is limited in its ability to capture complex, non-linear relationships within the data, as it relies purely on distance metrics. Additionally, KNN is sensitive to irrelevant or redundant features, requiring extensive feature engineering. The potential of employing more sophisticated ML models to directly learn from raw data remains unexplored.

3 Proposed Method

3.1 Framework

We propose a novel surrogate-assisted GP algorithm based on a NN Classifier, referred to as SGP-NNC. This algorithm features novel preselection and surrogate update processes within its framework (as shown in Fig. 2, where the components distinct from the existing surrogate-assisted GP are highlighted in dark boxes). In this framework, we first generate random scheduling rules to initialise the population P with size $size_{pop}$. Individuals with identical PC vectors are identified as duplicates and replaced with randomly regenerated ones. We evaluate the individuals in P through scheduling simulations to get their real fitness.



		Decision Situation s_1				Decision Situation s_2			
Candidate Jobs		J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
Raw-type PC (Priority)	Ind_1	15	9	17	85	44	22	62	84
	Ind_2	54	48	42	56	23	35	79	59
Ref Ranking		3	4	2	1	2	1	4	3
Ranking -type PC	Ind_1	1				3			
	Ind_2	1				4			

Fig. 3. Example of calculating the raw-type and ranking-type PC.

The PC vectors and real fitness of these individuals are used as training data to update the NNC surrogate. Subsequently, we conduct tournament selection to choose parents and produce offspring through crossover, mutation, and elitism (copying top individuals in P). We continue generating new offspring, retaining only those with unique PC vectors, until the intermediate population, which is considerably larger than P, is fully filled. Population P is then emptied, and top individuals in P_{imd} are selected using the min-heap technique. A min-heap is a binary tree data structure where the value of parent nodes is less than or equal to the value of its children. Individuals in P_{imd} are first randomly assigned to a balanced binary tree. We then build the min-heap (detailed in [1]) by comparing pairs of individuals using the NNC surrogate to predict the better individual. The desired individual can be found at the root node of the binary tree. This selection process is repeated until the top $size_{pop}$ individuals are selected.

This preselection approach is markedly distinct from existing surrogates that sort the entire P_{imd} based on the fitness values of individuals. The min-heap offers a time complexity advantage over sorting algorithms when we only need to find the top subset and do not require ordering the remaining elements.

3.2 New Phenotypic Characterisation Representation

The behavioural features of an individual are extracted as the PC vector and serve as the input to the NNC surrogate model. To represent the behaviour of an individual, we apply them to a set of predefined decision situations and concatenate the priority values into the PC vector. Figure 3 presents a simple example with two decision situations, each containing four candidate jobs. After analysing the situation, an individual assigns a priority value to each candidate. We visualise priority values with rectangular bars and highlight the highest priority in each decision situation in orange. Priority values comprehensively reflect the detailed assessment of the candidates, according to which the individual makes scheduling decisions. We propose using priority values as a new PC representation, referred to as the raw-type PC, as priorities are the raw output of an individual. In Fig. 3, the raw-type PC of Ind_1 is an 8-dimensional vector $\langle 15, 9, 17, 85, 44, 22, 62, 84\rangle$.

The commonly used existing PC representation neglects the detailed information in priority values and directly takes the scheduling decision (candidate

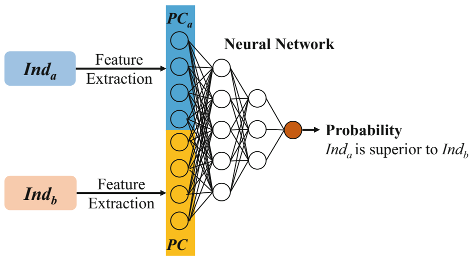


Fig. 4. Architecture of the NNC surrogate.

with the highest priority) as the behaviour of an individual. To quantify differences in decisions among different individuals, a manually designed reference rule is introduced. It provides a ref ranking for each candidate, and the ranking value corresponding to the highest priority is recorded as an element of the PC vector [2]. This PC representation is a vector of ref rankings, so we refer to it as the ranking-type PC. For instance, the ranking-type PC of Ind_1 in Fig. 3 is the vector $\langle 1, 3\rangle$.

Ranking-type PC utilises domain knowledge to refine the features as low-dimensional vectors. Although it is easy to process by ML models, it has two major flaws. Firstly, it heavily relies on the reference rule, which is a scheduling heuristic. The performance of the reference rule cannot be guaranteed, and features extracted by this method may vary when using different reference rules. However, PC is an inherent property of an individual and should not be influenced by external factors such as the setting of the reference rules. Secondly, ranking-type PC struggles to distinguish subtle differences in individual behaviours. Taking the two individuals in Fig. 3 as an example, ranking-type PC sees their behaviour in decision situation s_1 as identical since they both assign the highest priority to job J_4. However, there are subtle differences in their assessments of the candidates. Ind_1 assigns a considerably higher priority to J_4 than to other candidates, firmly choosing this candidate; by contrast, Ind_2 allocates similar priorities across all four candidates, indicating a lack of strong preference.

The raw-type PC overcomes these flaws because it does not depend on reference rules, eliminates ambiguity in representing individual behaviours, and accurately reflects the nuances in individual behaviours.

3.3 Surrogate Model Based on Neural Network Binary Classifier

The proposed NNC surrogate model is shown in Fig. 4. It compares two individuals by horizontally concatenating their PC vectors (normalised using min-max scaling according to the minimum and maximum values of a PC vector) into a single vector which is then fed into a neural network. The output is a single value representing the probability that Ind_a is superior to Ind_b.

The updating of surrogate models is an online learning process where new data (evaluated individuals) arrives each generation. These updates enable the surrogate to adapt to constantly evolving individuals. Unlike KNN surrogate,

Table 1. Parameter settings for surrogate-assisted GP.

GP Parameter	Value	NNC surrogate parameter	Value
Population size ($size_{pop}$)	1000	Number of hidden layers	3
Intermediate Population size	3000	Number of neurons	64, 64, 32
Crossover / Mutation rate	90% / 10%	Activation	ReLU
Elitism	10	Learning rate	0.001
Parent selection	Tournament (size 7)	Optimiser	Adam
Maximum depth	8		

which must clear previous data and rebuild the surrogate from scratch, SGP-NNC can retain previously learned knowledge as it does not conflict with new data. This capability allows for the neural network parameters in SGP-NNC to be updated through incremental learning. To collect the training data, the evaluated individuals in each generation are compared with each other; if Ind_a has a better real fitness than Ind_b, the label for this pair is set to 0, otherwise, it is set to 1. The NN parameters are updated using stochastic gradient descent to minimise the binary cross-entropy loss.

4 Experimental Design

This paper adopts the widely recognised DJSS simulation configuration [3]. In each simulation scenario, jobs arriving continuously are processed by 10 machines. To ensure the simulation reflects a stable state of the job shop, we initially release 1000 jobs to "warm up" the system. Subsequently, we monitor the next 5000 jobs to evaluate the performance of dispatching rules. Jobs arrive at the job shop following a Poisson process with a rate $\lambda = \mu P_M / p$, where μ represents the average processing time of the machines, and P_M signifies the probability of a job visiting a machine. The utilisation level (p) is employed to control the busyness of the job shop [8]; a higher utilisation level corresponds to a busier job shop. Jobs are assigned varying weights, with 20%, 60%, and 20% of the jobs having weights of 1, 2, and 4, respectively. The number of operations per job follows a uniform discrete distribution ranging from 2 to 10. The processing times for these operations are drawn from a discrete distribution with values between 1 and 99.

Features of the job shop are extracted to support decision-making processes. This study adopts 16 commonly used features in DJSS, as detailed in [3]. We align our parameter settings with the mainstream settings for surrogate-assisted GP in DJSS [2,8,10], which are presented in Table 1. Following previous studies [8,10], we calculate the PC vector based on 40 predefined decision situations, each containing 7 candidate jobs. Consequently, the dimension of the ranking-type PC is 40, while the dimension of the raw-type PC is 280.

Table 2. Test performance of basic GP, SGP-random, SGP-KNNR, and SGP-NNC.

Scenario	Basic GP	SGP-random	SGP-KNNR	SGP-NNC
$\langle \mathrm{WT}_{\mathrm{mean}}, 0.85\rangle$	718.3(6.6)	713.6(3.3)(↑)	710.4(2.9)(↑↑)	710.2(2.5)(↑↑≈)
$\langle \mathrm{WT}_{\mathrm{mean}}, 0.95\rangle$	1673.6(23.6)	1663.3(22.1)(≈)	1646.2(17.8)(↑↑)	1635.9(13.2)(↑↑↑)
$\langle \mathrm{T}_{\mathrm{mean}}, 0.85\rangle$	395.5(4.3)	392.0(2.4)(↑)	389.4(1.4)(↑↑)	389.0(1.2)(↑↑≈)
$\langle \mathrm{T}_{\mathrm{mean}}, 0.95\rangle$	979.0(10.7)	973.5(6.6)(↑)	968.3(6.9)(↑↑)	964.6(4.7)(↑↑↑)
$\langle \mathrm{T}_{\mathrm{max}}, 0.85\rangle$	1425.9(31.5)	1410.4(33.6)(≈)	1375.1(29.9)(↑↑)	1372.0(25.6)(↑↑≈)
$\langle \mathrm{T}_{\mathrm{max}}, 0.95\rangle$	3263.4(131.8)	3194.9(97.3)(≈)	3107.3(102.4)(↑↑)	3057.0(94.3)(↑↑≈)
Average Rank	3.49	2.94	1.97	1.60

5 Experiment Results

5.1 Test Performance

In the following experiments, we test the proposed SGP-NNC algorithm on six scheduling scenarios: $\langle \mathrm{WT}_{\mathrm{mean}}, 0.85\rangle$, $\langle \mathrm{WT}_{\mathrm{mean}}, 0.95\rangle$, $\langle \mathrm{T}_{\mathrm{mean}}, 0.85\rangle$, $\langle \mathrm{T}_{\mathrm{mean}}, 0.95\rangle$, $\langle \mathrm{T}_{\mathrm{max}}, 0.85\rangle$, and $\langle \mathrm{T}_{\mathrm{max}}, 0.95\rangle$. The scheduling objectives include $\mathrm{WT}_{\mathrm{mean}}$, $\mathrm{T}_{\mathrm{mean}}$, and $\mathrm{T}_{\mathrm{max}}$; and the utilisation levels of the job shop are 0.85 or 0.95. The test set for each scenario consists of 30 scheduling instances unseen by the model. All scenarios are minimisation problems, so lower objective values indicate better performance. In previous studies, basic GP needed to evolve for 51 generations to find high-quality scheduling rules [8,9]. Since the training time varies when using surrogates, we set the average training time of basic GP with 51 generations as the time budget for all the compared algorithms [9].

In existing studies, the state-of-the-art machine learning surrogate employs KNN regression and ranking-type PC to predict fitness values [2]. We refer to the GP algorithm assisted by this surrogate as SGP-KNNR. The performance of the proposed SGP-NNC algorithm is compared with that of Basic GP and SGP-KNNR. Unlike SGP algorithms, basic GP lacks a preselection process that removes duplicate individuals. To assess the net improvement from the prediction accuracy of surrogates, we introduced a comparison algorithm, SGP-random, which follows the same procedures as other SGPs but randomly predicts fitness values. Table 2 shows the mean and standard deviation of objective values for the four algorithms over 30 independent runs, analysed using the Wilcoxon rank-sum test and the Friedman test with a significance level of 5%. "(↑)" and "(↓)" denote results significantly better or worse than competitors, respectively, while "(≈)" indicates no significant differences. For example, in scenario $\langle \mathrm{WT}_{\mathrm{mean}}, 0.85\rangle$, SGP-NNC performs significantly better than Basic GP and SGP-random and performs similarly to SGP-KNNR. Additionally, we present the average rank of each algorithm across all scenarios.

The results show that the proposed SGP-NNC algorithm outperforms all other algorithms, with the best average rank of 1.60. When compared with the state-of-the-art SGP-KNNR algorithm, SGP-NNC exhibits lower mean objective values and smaller standard deviations across all six scenarios. In scenarios $\langle \mathrm{WT}_{\mathrm{mean}}, 0.95\rangle$ and $\langle \mathrm{T}_{\mathrm{mean}}, 0.95\rangle$, the performance of SGP-NNC is significantly

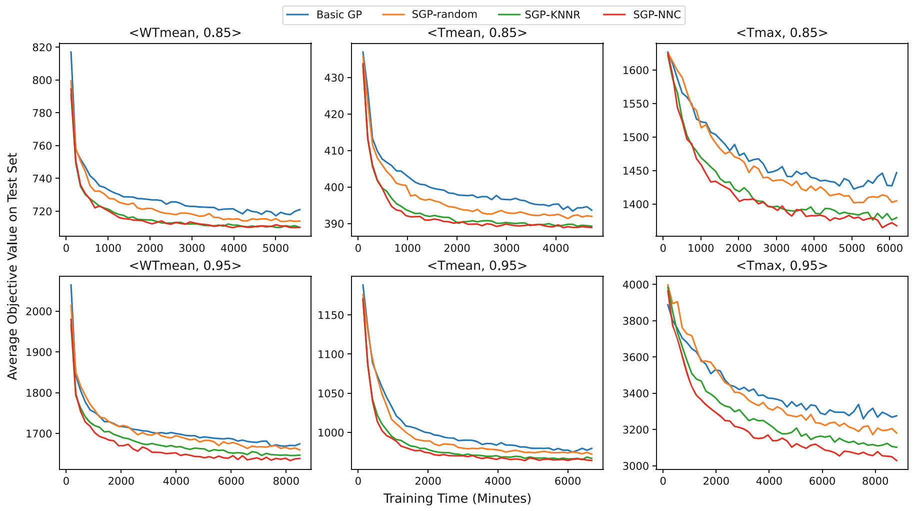


Fig. 5. Average objective value curves of Basic GP, SGP-random, SGP-KNNR, and SGP-NNC on the test set over 30 independent runs.

better than SGP-KNNR. SGP-random shows significant improvement over Basic GP in three of six scenarios, demonstrating that the process of removing duplicates enhances algorithm performance by maintaining population diversity. Both SGP-NNC and SGP-KNNR significantly outperform SGP-random in all six scenarios, indicating that accurate predictions of promising individuals help GP find better scheduling rules.

An effective SGP algorithm is expected to exhibit high convergence speed. To analyse the efficiency of the algorithms in comparison, we recorded their best output (scheduling rule) at each generation and tested them on the test set. The average objective value curves of the four algorithms on the six scheduling scenarios are shown in Fig. 5. According to the figure, SGP-NNC converges much faster than Basic GP and SGP-random in all six scenarios. The SGP-KNNR algorithm shows comparable convergence speed, but the curves of SGP-NNC generally lie beneath those of SGP-KNNR in most scenarios except $\langle \mathrm{WT}_{\mathrm{mean}}, 0.85\rangle$. Its advantages are particularly evident in scenarios $\langle \mathrm{WT}_{\mathrm{mean}}, 0.95\rangle$ and $\langle \mathrm{T}_{\mathrm{max}}, 0.95\rangle$, indicating that, given the same amount of training time, SGP-NNC is most likely to yield the best scheduling rules.

5.2 Ablation Study

SGP-NNC contains two new components: the raw PC type and the NNC surrogate. Although the results demonstrate the superiority of SGP-NNC, it is essential to investigate the effect of the raw-type PC and the NNC surrogate separately. To this end, we pair ranking-type and raw-type PC with KNNR and NNC surrogates to form four variants: Ranking-KNNR, Ranking-NNC, Raw-

Table 3. Test performance of the four variants: Ranking-KNNR, Ranking-NNC, Raw-KNNR, and Raw-NNC.

Scenario	Ranking-KNNR	Ranking-NNC	Raw-KNNR	Raw-NNC
$\langle WT_{mean}, 0.85\rangle$	710.4(2.9)	711.0(3.9)(≈)	710.7(3.2)(≈≈)	710.2(2.5)(≈≈≈)
$\langle WT_{mean}, 0.95\rangle$	1646.2(17.8)	1645.0(15.0)(≈)	1640.0(14.1)(≈≈)	1635.9(13.2)(↑↑≈)
$\langle T_{mean}, 0.85\rangle$	389.4(1.4)	389.5(1.5)(≈)	388.9(1.5)(≈≈)	389.1(1.2)(≈≈≈)
$\langle T_{mean}, 0.95\rangle$	968.3(6.9)	967.4(5.4)(≈)	968.3(7.8)(≈≈)	964.6(4.7)(↑↑≈)
$\langle T_{max}, 0.85\rangle$	1375.1(29.9)	1384.6(23.0)(≈)	1377.1(26.5)(≈≈)	1372.0(25.6)(≈≈≈)
$\langle T_{max}, 0.95\rangle$	3107.3(102.4)	3126.4(74.8)(≈)	3066.7(82.5)(≈↑)	3057.0(94.3)(≈↑≈)
Average Rank	2.61	2.79	2.41	2.20

KNNR, and Raw-NNC. Their test performances on the six scheduling scenarios over 30 independent runs are shown in Table 3.

The p-value of the Friedman test for the four algorithms is 1.1E-4, far less than 0.05, indicating that the performances of the four variants differ and require further pairwise Wilcoxon rank sum tests. Raw-NNC performs best with an average rank of 2.20, suggesting that both raw-type PC and NNC surrogate contribute to the advantage of SGP-NNC. According to the average ranks, the two variants using raw-type PC generally outperform those using ranking-type PC, indicating that raw-type PC leads to the major performance improvement in the proposed algorithm. When comparing Raw-KNNR and Raw-NNC, although the Wilcoxon rank-sum test shows no significant difference between them, Raw-NNC achieves better mean performance in five out of six scenarios.

5.3 Prediction Accuracy Analysis

In this section, we conduct a further analysis on the prediction accuracy of different surrogate models. A clear performance difference among the surrogate models can be seen in scenario $\langle WT_{mean}, 0.95\rangle$. It is interesting to investigate if this performance difference is truly due to the prediction abilities of the different surrogates. To this end, we fully evaluate the individuals in P_{imd} across 10 GP runs and check if the good individuals are actually selected.

First, we rank individuals according to their real fitness, with the top 1000 being those desired to be selected. Figure 6(a) shows the proportion of individuals ranked at a specific place that are correctly selected by different surrogate models. From the figure, we observe that the better the individuals are, the higher the proportion they are selected. Raw-NNC exhibits higher prediction accuracy across the whole spectrum, indicating that the superior performance of SGP-NNC is attributed to the accurate predictions made by the surrogate.

Figure 6(b) provides a detailed illustration of the 3000 offspring are selected by the proposed NNC surrogate. Each pixel in this heatmap represents a specific individual in a specific generation and rank, with the colour indicating the count of selections across 10 GP runs. The yellower the colour, the more frequently the individual is selected. From the figure, we observe that a large proportion of top-ranked individuals are selected by the NNC surrogate.

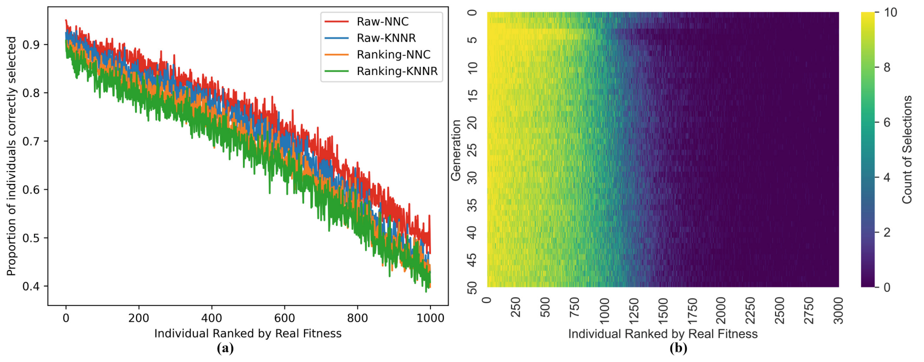


Fig. 6. Prediction accuracy: (a) Comparison of four surrogate variants (b) The heatmap of individual selection counts of the proposed NNC surrogate across 10 GP runs.

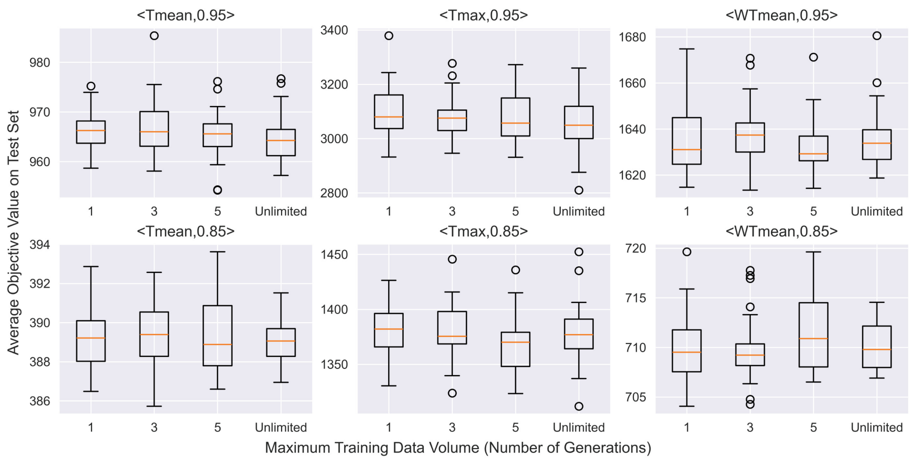


Fig. 7. Test performance of the SGP-NNC algorithm with different volumes of training data for the surrogate.

5.4 Impact of Training Data Volume

A notable characteristic of the proposed SGP-NNC algorithm is its ability to be trained with more data. To investigate the impact of training data volume on SGP-NNC, we conduct an experiment where the surrogate is trained on maximum of 1, 3, or 5 generations of data. When the training times of a NN reached this limit, its parameters are reinitialised. The test performances of SGP-NNC with different volumes of training data are shown in Fig. 7. The x-axis represents the maximum generations of data the surrogate is trained on, while the "Unlimited" serves as a control group where training data contains all the historical data. The y-axis represents the average objective value, with lower values being better.

From the results, we can see that the increase in training data slightly improves algorithm performance only in scenarios $\langle \mathrm{T}_{\mathrm{mean}}, 0.95\rangle$ and $\langle \mathrm{T}_{\mathrm{max}}, 0.95\rangle$. However, there is no apparent pattern in other scenarios, indicating that increasing the training data volume does not significantly benefit the algorithm. Nevertheless, the performance does not degrade when using more than one generation of data, suggesting that the binary classification mechanism effectively makes data from different generations compatible.

There are three possible reasons why increasing the training data does not improve algorithm performance as expected. First, the neural networks may have reached their model capacity during training, so additional data does not enhance their ability to distinguish good individuals. Second, training data from the current generation are most influential because newly generated offspring is most similar to individuals in the current generation. Therefore, even if the surrogate discards previously learned knowledge and trains on only one generation of data, it can still provide competitive performance. Third, not all training data are equally important; individuals from previous generations can be less important or redundant. Therefore, additional training data can introduce noise, hindering the performance of the surrogate.

6 Conclusion

The primary goal of this paper is to design an effective surrogate model that enhances the effectiveness of GP in discovering good scheduling rules. This goal is achieved through the newly proposed SGP-NNC algorithm. By using priority values as raw-type PC and employing a NN binary classifier as the surrogate model, SGP-NNC considerably improves the prediction accuracy of the surrogate. Extensive experiments across six scheduling scenarios show that SGP-NNC finds better heuristics and converges faster than basic GP and SGP-KNNR, though increased training data does not yield notable benefits.

The study addresses previous inadequacies in the PC representation and demonstrates the effectiveness of neural network surrogates. The proposed binary classification mechanism sheds new light on the surrogate design, which was originally dominated by fitness value regression. Further exploration of classification-based surrogates is expected to be promising. Additionally, the proposed surrogate is a general approach for predicting the performance of heuristics based on its behaviour. Similar ideas can be applied to other combinatorial optimisation problems, such as project scheduling and vehicle routing. Our future work will focus on identifying important training samples, better leveraging increased data volume, exploring improved preselection methods, and incorporating statistical models to further assist surrogate predictions.

References

1. Al-Jaloud, E.S., Al-Aqel, H.A., Badr, G.H.: Comparative performance evaluation of heap-sort and quick-sort algorithms. Int. J. Comput. Acad. Res. **3**(2), 39–57 (2014)
2. Hildebrandt, T., Branke, J.: On using surrogates with genetic programming. Evol. Comput. **23**(3), 343–367 (2015)
3. Huang, Z., Mei, Y., Zhang, F., Zhang, M.: Grammar-guided linear genetic programming for dynamic job shop scheduling. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1137–1145. ACM (2023)
4. Jakobović, D., Marasović, K.: Evolving priority scheduling heuristics with genetic programming. Appl. Soft Comput. **12**(9), 2781–2789 (2012)
5. Mei, Y., Nguyen, S., Xue, B., Zhang, M.: An efficient feature selection algorithm for evolving job shop scheduling rules with genetic programming. IEEE Trans. Emerg. Topics Comput. Intell. **1**(5), 339–353 (2017)
6. Nguyen, S., Zhang, M., Tan, K.C.: Surrogate-assisted genetic programming with simplified models for automated design of dispatching rules. IEEE Trans. Cybern. **47**(9), 2951–2965 (2017)
7. Zeiträg, Y., Figueira, J.R., Horta, N., Neves, R.: Surrogate-assisted automatic evolving of dispatching rules for multi-objective dynamic job shop scheduling using genetic programming. Expert Syst. Appl. **209**, 118194 (2022)
8. Zhang, F., Mei, Y., Nguyen, S., Tan, K.C., Zhang, M.: Instance-rotation-based surrogate in genetic programming with brood recombination for dynamic job-shop scheduling. IEEE Trans. Evol. Comput. **27**(5), 1192–1206 (2023)
9. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: Collaborative multifidelity-based surrogate models for genetic programming in dynamic flexible job shop scheduling. IEEE Trans. Cybern. **52**(8), 8142–8156 (2022)
10. Zhang, F., Mei, Y., Nguyen, S., Zhang, M.: Phenotype based surrogate-assisted multi-objective genetic programming with brood recombination for dynamic flexible job shop scheduling. In: IEEE Symposium Series on Computational Intelligence, pp. 1218–1225 (2022)
11. Zhang, F., Mei, Y., Nguyen, S., Zhang, M., Tan, K.C.: Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. IEEE Trans. Evol. Comput. **25**(4), 651–665 (2021)
12. Zhang, F., Mei, Y., Zhang, M.: Surrogate-assisted genetic programming for dynamic flexible job shop scheduling. In: Mitrovic, T., Xue, B., Li, X. (eds.) AI 2018. LNCS (LNAI), vol. 11320, pp. 766–772. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03991-2_69
13. Zhou, Y., Yang, J.J., Huang, Z.: Automatic design of scheduling policies for dynamic flexible job shop scheduling via surrogate-assisted cooperative co-evolution genetic programming. Int. J. Prod. Res. **58**(9), 2561–2580 (2020)
14. Đurasević, M., Jakobović, D.: Comparison of schedule generation schemes for designing dispatching rules with genetic programming in the unrelated machines environment. Appl. Soft Comput. **96**, 106637 (2020)
15. Đurasević, M., Jakobović, D.: Selection of dispatching rules evolved by genetic programming in dynamic unrelated machines scheduling based on problem characteristics. J. Comput. Sci. **61**, 101649 (2022)