Surrogate-assisted Genetic Programming with Simplified Models for Automated Design of Dispatching Rules

Su Nguyen, *Member, IEEE* and Mengjie Zhang, *Senior Member, IEEE* and Kay Chen Tan, *Fellow, IEEE*

***Abstract*—Automated design of dispatching rules for production systems has been an interesting research topic over the last several years. Machine learning, especially genetic programming (GP), has been a powerful approach to dealing with this design problem. However, intensive computational requirements, accuracy and interpretability are still its limitations. This paper aims at developing new surrogate assisted GP to help improving the quality of the evolved rules without significant computational costs. The experiments have verified the effectiveness and efficiency of the proposed algorithms as compared to those in the literature. Furthermore, new simplification and visualisation approaches have also been developed to improve the interpretability of the evolved rules. These approaches have shown great potentials and proved to be a critical part of the automated design system.**

***Index Terms*—scheduling, evolutionary design, hyper-heuristic, genetic programming**

I. Introduction

The current highly competitive markets require make-to-order companies to be flexible and adapt quickly to changes. While technologies are one of the key competitiveness, they usually require large investments. Another way to improve the productivity is to strive for better operations. Scheduling is one of the important tasks to efficiently utilise available manufacturing resources for better customer satisfaction. To be more reactive, dispatching rules are commonly used in the shop to decide processing orders of jobs. The advantages of dispatching rules are their simplicity (i.e. require low computational costs), ease for implementation, and understandability (i.e. provide good managerial insights). However, designing effective dispatching rules is not straightforward and usually requires many cycles of trial-and-error. In fact, a lot of dispatching rules have been proposed for different types of manufacturing systems and scheduling objectives. Unfortunately, the literature has demonstrated that there are no universal rules that are superior in all situations. Therefore, it is important to design specialised rules for particular situations. Computer simulation is usually used by researchers to evaluate the effectiveness of dispatching rules before they are applied to the real environments. Due to the complex nature of scheduling problems, many attributes from jobs and systems need to be considered; therefore, manually designing dispatching rules is time-consuming and still mainly relies on trial-and-error.

Recently, different automated design approaches have been proposed to generate dispatching rules [1]. Genetic programming (GP) [2], [3], [4] is currently the most popular approach for this task because of the flexibility of its representations, powerful search ability, and acceptable interpretability of evolved rules [5]. The basic idea of this approach is relative simple. GP is used to represent and generate rules based on the sets of terminals (i.e. attributes of jobs/systems) and functions (e.g. arithmetic operators); then discrete event simulation is automatically used to evaluate the performance of the evolved rules. Rules with better fitness/quality are more likely to be selected to reproduce new rules for the next generation.

Previous studies have shown the superiority of the rules evolved by GP and further improvements on representations and search mechanisms have also been made to enhance the quality of the evolved rules. Although the results are promising, the time for GP to evolve dispatching rules are still long, especially when we deal with large manufacturing systems (i.e. hours to days on an average computer). The most time consuming component is fitness evaluations with simulation. To overcome this problem, surrogate models have been developed. Hildebrandt and Branke [6] proposed a simple surrogate model based on historical data from GP search and similarity measures of rules. The experimental results show that surrogate assisted GP (SGP) outperforms the traditional methods and GP also converges faster with the support of the surrogate model [6]. Nevertheless, this surrogate model still has some drawbacks: (1) the accuracy of the surrogate model is not good enough, (2) the similarity measure can only be applied to the simple case where only sequencing decisions are considered, and (3) the length of rules evolved by SGP is still significantly large. It is noted that the accuracy of the surrogate models will influence the chance to obtain good rules. Meanwhile, the similarity measure restricted to sequencing decisions will influence the applicability of dispatching rules in practice (e.g. when dealing with flexible job shops, batching process). Finally, the large evolved rules are more difficult to interpret.

In this paper, we will study different surrogate models and analyse their advantages and disadvantages when dealing with dynamic job shop scheduling (DJSS). In DJSS, the shop has a set of machines and jobs will arrive randomly over time. In this case, a job is a sequence of operations, each of which is to be performed on a particular machine. In DJSS, the routes of jobs are fixed, but not necessarily the same for each job. Instead of using similarity measure to estimate fitness, we will directly use the simulation models of a simplified shop which is a lot more efficient than the original simulation model. The overall goal of this study is to improve SGP to evolve more competitive and interpretable dispatching rules. Following are our research objectives:

- Develop new surrogate models based on the simplified simulation model of the shop.
- Analyse the effectiveness of new SGP methods in terms of testing performance, program/rule length, and computational costs.
- Analyse the behaviours of SGP with different surrogate models to gain more insights for further improvements.
- Develop new post-evolution approach to simplifying and visualising evolved rules.

These research objectives aim to overcome the limitations of the current SGP methods. In the next section, we will provide a brief review of existing studies on genetic programming based hyper-heuristics (GPHH) for scheduling problems. Section III describes the simulation model used in this paper. Section IV presents different surrogate models used in this study and how they can be applied to GP. Experimental results are presented in Section V. Section VI further analyses the evolutionary process of proposed SGP methods to understand how effective rules are evolved. Simplification and visualisation of obtained rules are provided in Section VII. Conclusions and future developments are shown in Section VIII.

II. Background

This section will briefly review previous studies on dispatching rules. Then, we discuss some past studies on using GP and machine learning techniques for discovering new dispatching/scheduling rules in the literature. Finally, some recent advances in this field are presented.

A. Dispatching rules for DJSS

Studies on scheduling for job shop scheduling (JSS) mainly focus on applying mathematical programming and meta-heuristics [7] to find the optimal or near-optimal solutions for static instances which are the snapshots of manufacturing systems at certain decision moments. Theoretically, these solutions can be applied to the system given that there is no unexpected changes in the shop, e.g. arrivals of new jobs, machine breakdowns, and order cancellations. Unfortunately, it is not the case in real-world scheduling applications as disruptions are an natural feature in practice [8], [9], [10]. Moreover, the real-world scheduling problems can have a large number of jobs which cause computational difficulty for optimisation methods (both exact or meta-heuristics) [11], [12]. Therefore, dispatching rules are an suitable approach to dealing with these practical situations because they are computationally efficient and can react quickly to dynamic changes in the shop.

There have been hundreds of dispatching rules proposed in the literature to deal with different types of manufacturing environments. Normally a dispatching rule is characterised by a priority function that determines priorities of jobs waiting in the queue (the job with highest priority will be processed next). Dispatching rules are usually classified based on the information used to make scheduling decisions (e.g. static, dynamic, local, global) [7] and how these pieces of information are combined. The effectiveness of a dispatching rule depends on how it copes with the dynamic changes of the shops and the ability to take into account different factors that can affect the considered objective to be optimised. The comparisons of different dispatching rules have been continuously done in many studies [13], [14], [15]. Different from static JSS problems where performance of proposed algorithms can be easily examined by a set of static instances, the performance of dispatching rules for dynamic problems has to be evaluated based on discrete event simulation (DES) [16]. Theoretical and practical studies of dispatching rules use DES to compare the effectiveness and understand the behaviours the rules under different scenarios of DJSS (i.e. different utilization, shop configurations). Different objectives were also considered in these studies as they are the natural requirements in real world applications.

B. Automated design of dispatching rules

Scheduling rules and heuristics are commonly designed for a specific problem. However, designing a sophisticated rule requires a tedious trial-and-error process (i.e. design, test, modify). In recent years, there is a trend to apply machine learning and optimisation techniques to automate the design process [17], [18], [19], [20]. The proposed methods are commonly known as hyper-heuristics [21], which is a special heuristic to automatically *"select or generate heuristics to solve hard computational search problems"* [21], [22]. The existing hyper-heuristics for generating scheduling rules and heuristics can be classified based on their (1) *learning methods* and (2) *representations* [1].

The learning methods in this case can be either supervised or unsupervised. Supervised learning is mainly applied when the optimal (or the best practice) solutions are known. Many well-known machine learning techniques have been applied for supervised learning to discover dispatching rules such as neural networks [23], logistic regression [24], decision trees [18]. The drawback of supervised learning is that the training set (optimal solutions) may not be available as exact optimisation methods are usually suitable for very small instances only. On the other hand, unsupervised learning is more flexible because we only need to know how well the obtained rules perform on the training set (i.e. problem instances in this case) and use this information to guide the search toward better rules in the search space. Unsupervised learning is currently a more popular approach to automatic design of dispatching rules. An advantage of unsupervised learning is that different representations (e.g. linear/parameter-based, tree-based, grammar-based) can be employed to generate rules. Many unsupervised methods have been developed in the literature based on genetic algorithms [25], [26], genetic programming [5], [27], [28], [29], [30], [31] and neural networks [32].

To generating dispatching rules, either fixed-length parametric representations or variable-length grammar-based representations can be used [1]. In the first set of representations, dispatching rules are commonly in a simple and fixed-structure format such as weight sum (of attributes) [25], [26] and neural networks [23]. Meanwhile, the grammar-based representations specifies how the individual components can be assembled to yield a valid priority function. Representations (tree-based, linear-based, or graph-based) of GP are one of the typical examples of grammar based representations. As compared to

the fixed-length representations, the search based grammar-based representations are much more complicated. For a comprehensive review of automated design methods for designing production scheduling heuristics can be found in [1].

C. Genetic programming for designing dispatching rules

Recently, GP has become popular in the field of hyper-heuristics and it is known as genetic programming based hyper-heuristics (GPHH) [33]. Because GP is able to represent and evolve complex programs or rules, it naturally becomes an excellent candidate for heuristic generation. GPHH has also been applied to evolve dispatching rules for scheduling problems [31], [34], [35]. Several GPHH methods have also been proposed for JSS problems. Atlan et al. [36] applied GP for JSS problems. However, the focus of their paper is on finding the solution for a particular problem instance. Miyashita [28] examined three potential multi-agent models to evolve dispatching rules in multiple machine environments: (1) a homogeneous model where all machines share the same dispatching rule, (2) a distinct agent model where each machine employs its own evolved rule, and (3) a mixed agent model where two rules can be selected to prioritise jobs depending on whether the machine is a bottleneck. The experiments showed that the distinct agent model provided better results in the training stage compared to the homogeneous model but had some over-fitting problems. The mixed agent model was the most robust in all the experiments but required the prior-knowledge about the bottleneck machine, which can change in dynamic situations. To handle this issue, Jakobovic and Budin [37] proposed a new GP method called GP-3 to provide some adaptive behaviour for the evolved rules. In their method, GP is used to evolve three components of the rules including a decision tree and two dispatching rules for bottleneck and non-bottleneck machines. The purpose of the decision tree is to identify whether a considered machine is a bottleneck and decide which of the two evolved rules should be applied. The experiments showed that this method can provide better rules than a simple GP method. However, it is noted that the superior performance of GP-3 will depend on the bottleneck machines. If the load levels between machines in the shops are rather similar (existence of multiple bottleneck machines), the information/output from the decision tree in GP-3 may not be very helpful. Nguyen et al. [5] investigated different representations of dispatching rules with GP. The experiments showed that a mixed representation based on decision-tree like representation and arithmetic representations provided the best the results. However, they mainly examined their rules on static scheduling instances.

Tay and Ho [30] performed a study on using GP for multi-objective JSS problems. In their method, three objectives are linearly combined (with the same weights) into an aggregate objective, which is used as the fitness function in the GP method. The experiments showed that the evolved rules are quite competitive as compared to simple rules but still have trouble dominating the best rule for each single objective. In another study, Hildebrandt et al. [27] explained that the poor performance of the rules evolved by Tay and Ho [30] is caused by the use of a *linear* combination of different objectives and the fact that the randomly generated instances cannot effectively represent the situations that happen in a long term simulation. For that reason, Hildebrandt et al. [27] evolved dispatching rules by training them on different simulation scenarios and only minimised the mean flow time. Some aspects of the simulation models were also discussed in their study. The experimental results showed that the evolved rules were quite complicated but effective as compared to other existing rules. Moreover, these evolved rules are also robust when tested with another environment. However, their work did not consider how to handle multiple conflicting objectives. Nguyen et al. [38] proposed a cooperative coevolution GPHH for multi-objective dynamic JSS problems. In that work, the due dates of new jobs are assumed to be assigned internally and two scheduling rules (dispatching rule and due date assignment rule) are simultaneously considered in order to develop effective scheduling policies. While the representation of the dispatching rules is similar to those in other GP methods, the operation-based representation [39] is used to represent the due date assignment rules. The results showed that the evolved scheduling policies can outperform scheduling policies from different combinations of existing dispatching rules and due-date assignment rules in different simulation scenarios.

In another study, Beham et al. [40] utilise parallel technologies to evolve dispatching rules for a flexible job shop with a large terminal and function sets. They develop three new GP methods based on island models and SASEGASA [41] in which rules are evolved in multiple subpopulations. The results show that the SASEGASA method can cope better with the states of exception in the simulation than island based methods. Pickardt et al. [42] proposed a two-stage approach to evolving dispatching rule sets for semiconductor manufacturing. In the first stage, GP is used to evolve general dispatching rules. The best obtained dispatching rule is combined with a list of benchmark dispatching rules to generate a set of candidate rules. In the second stage, a $\mu + \lambda$ evolutionary algorithm (EA) [42] is used to select the most suitable dispatching rule in the set of candidate rules for each work centre in the shop. The experiments in this paper compare the performance of the two-stage hyper-heuristics with the pure GP and EA hyper-heuristics. The results show that the three hyper-heuristics outperformed benchmark dispatching rules and the two-stage hyper-heuristics produced significantly better performance than the other two hyper-heuristics.

To cope with expensive fitness evaluations, surrogate models are used. Hildebrandt and Branke [6] investigated two surrogate models for evolving dispatching rules to minimise mean flowtime. In their approach, a large number of individuals are generated through genetic operations and the fitness of these rules is *approximated* by using the fitness of the most similar rules generated in the previous generations. Then, only rules with the top approximated fitness are *selected* for the next generation and receive *real* fitness evaluations. The experimental results showed that surrogate-assisted GP (SGP) is more effective than the simple GP method. Specifically, SGP can converge to good dispatching rules faster than GP given that the same computational budget is used.

GP has gradually proven to be a powerful approach to

automated design of dispatching rules. However, in order to make GP attractive in practice, more issues have to be considered and addressed. First, the efficiency of the proposed GP methods needs to be improved so that the managers/researchers can easily examine their hypotheses and conduct large experiments. Also, efficiency is important in order to deal with large scale and complex manufacturing systems. Second, we need to improve the accuracy of GP algorithms to find better dispatching rules especially when the search space is large. Finally, evolved rules have to be understandable to the users (i.e. managers and operators). Manual analyses are commonly used in the literature [27], [5] but this approach is impractical when we consider a large number of rules (e.g. investigate their trade-offs [43]). Recent efforts have been made to tackle these issues and promising results are shown. Branke et al. [32] visualised their priority indices as functions of incorporated attributes. However, this visualisation technique is only possible for a very limited number of attributes. In general, it is important to develop a set of new automated analysis techniques that are capable of dealing with a large number of attributes while providing key insights into the behaviours of the evolved rules.

D. Surrogate-assisted evolutionary algorithms

It is not uncommon for engineering applications to deal with expensive fitness evaluations. For time-consuming simulation-based evaluations, reduced models have been proposed to decrease the computational times in some applications such as simulation of wave [44], circuits and micromachined devices [45], [46], and real-time fluids [47]. While the results are very promising, they are mainly developed for very specialized processes which are difficult to be generalised.

Using pure evolutionary computation (EC) approaches when fitness evaluations are expensive is not efficient because they usually require a large number of fitness evaluations. To cope with this challenge, many surrogate models have been developed [48], [49] for fitness approximations. Popular models are polynomials [50], [51], artificial neural networks [52], [53], [54], support vector machine [55], and nearest neighbor [56], [57]. Ensemble techniques have also been employed to improve the accuracy of surrogate models by combining different models or select the most suitable models based on the evolutionary process [58], [59], [60], [61].

Jin [48] provided a comprehensive survey of surrogate-assisted EC methods in the literature and highlights key components in these methods. He classified surrogate methods based on the levels of approximation (problem approximation, functional approximation, and evolutionary approximation), and incorporation mechanism (migration, initialization, genetic operators, and fitness approximation). When using approximate models for fitness approximation, it is important to make sure that the algorithms can converge to the global optimum or near-optimum (not the false optimum from the approximate models). As a result, approximate models should be used along with the original fitness function. To remain efficient, model management is applied to control when to apply approximate models and when to use the original fitness function. Many strategies for model management can be used such as no control, fixed evolution control, and adaptive evolution control based on the fidelity of the approximate models.

III. Simulation model of DJSS

All experiments in this paper are based on the simulation model of a symmetrical job shop, which has been used in previous studies on dispatching rules [14], [43], [32]. Here are the simulation configurations:

- 10 machines
- Each job has 2 to 14 operations (re-entry is allowed)
- Processing times follow discrete uniform distribution $U[1, 99]$
- Job arrivals follow Poisson process
- Due date = current time + allowance factor $\times$ total processing time (allowance factor of 4 is used in our experiments)
- Utilisation of the shop is 85%, 95%
- No machines break-down; preemption is not allowed
- Weights of jobs are assigned based on the $4 : 2 : 1$ rule [62], [63] (this setting was inspired by Pinedo and Singer [63], which showed that approximately 20% of the customers are very important, 60% are of average importance and the remaining 20% are of less importance).

In each simulation replication, we start with an empty shop and the interval from the beginning of the simulation until the arrival of the 500^{th} job is considered as the warm-up time and the statistics from the next completed 5000 jobs [15] will be used to calculate performance measures. Three scheduling performance measures examined in our experiments are (1) mean flowtime, (2) mean tardiness, (3) total weighted tardiness. Although this simulation model is relatively simple, it still reflects key issues of real manufacturing systems such as dynamic changes and complex job flows. This section only considers a shop with high utilisation (85% and 95%) and tight due date (allowance factor of 4) because scheduling in this scenario is more challenging, and therefore easier to demonstrate the usefulness of GP. In order to reliably measure the effectiveness of evolved rules, a large number of simulation replications are usually needed (e.g. 30 to 50 simulation replications are usually needed to accurately estimate the performance of rules in the scenario described here). However, using simulation to evaluate the fitness of the evolved rules is also the most time-consuming part in GP for JSS. Therefore, only a small number of replications are usually used for fitness evaluations during the *training* process. As suggested by Hildebrandt et al. [27], we will use only one replication (corresponding to one random seed) for each fitness evaluation; however, we will change the replication (use a different random seed for simulation) when moving to a new generation. This strategy has been shown to be beneficial to prevent GP from overfitting to certain situations (replications). Two sets, each with 50 simulation replications, are used for determining *full training performance* and *testing performance*. The *full training set* is used to verify effectiveness of evolved rules and the *test set* is used to examine the performance of rules on unseen situations. In this dynamic case, we cannot identify

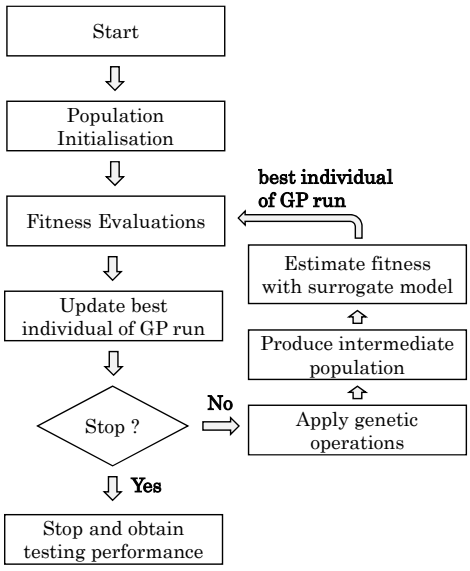


Fig. 1. Overall algorithm of SGP.

the best rule with the training performance or fitness functions, because they are changing across generations and they cannot accurately estimate the effectiveness of evolved rules.

IV. Proposed SGP methods

To handle the above issues, this paper will focus on developing new surrogate models to improve the efficiency and effectiveness of GP. In this section, we first explain how SGP work. Then, we develop new surrogate models and present the new SGP algorithm.

A. Basic SGP

Fig. 1 shows how SGP can evolve dispatching rules for a particular simulation scenario. Similar to most EC methods, SGP starts by randomly initialising a population (ramped-half-and-half). Each rule will be evaluated using simulation. The fitness of an evolved rule depends on the performance measure achieved by the rule when applied to the training replications (e.g. mean flowtime). The rule with the best fitness will be evaluated with the full training set (refer to [27] for detailed discussions of this strategy). If the best rule of the generation has better full training performance than the current best rule, it will be assigned as the current best rule of the run. If the stopping condition is met (i.e. maximum generation in GP), SGP will stop; otherwise, we perform the next steps to build the population for the next generation.

First, an intermediate population is created using genetic operations. This intermediate population has a larger population size as compared to the original population to increase the diversity in the population as well as improve the chance to find better rules. The fitness of all rules in the intermediate population is *approximated* by using the surrogate model. For example, Hildebrandt and Branke [6] proposed a surrogate model based on a decision vector, which estimates the fitness of an evolved rule by using the fitness of the most similar rules generated in the previous generations. SGP in [6] used fixed simulation replications and utilised individuals in the last two generations to approximate fitness of newly generated rules. For the surrogate model, the behaviour of an evolved rule is characterised by a *decision vector* based on a reference rule (2PT+WINQ+NPT) and the similarity of the two rules is measured by the distance of their corresponding decision vectors (see [6] for a detailed description). Nguyen et al. [64] further examined this model in the case with changing simulation replications (at each generation) and showed that the proposed method successfully improve the quality of evolved rules. In general, this surrogate model significantly enhances the performance of GP, and effectively utilises the historical search information, and the implementation is also straightforward. One disadvantage is that the dimension of decision vectors needs to be large enough to help differentiating evolved rules. It will become more tricky when we deal with complex manufacturing systems and special process. Also, this approach is designed for evolving dispatching rules. Thus, if there are more scheduling decisions taken into account (e.g. due date assignment, routing, order release), this surrogate model will not be suitable.

TABLE I
Terminal and function sets of GP

Symbol	Description
rrJ	time-in-system of job ($\mathtt{t} - releasetime$)
rRJ	job queuing time ($\mathtt{t} - readytime$)
RO	number of remaining operation within the job.
RT	work remaining of the job
PT	operation processing time
rDD	time to due date ($duedate - \mathtt{t}$)
RM	machine ready time
SL	slack of the job = DD − (t + RT)
WT	is the current waiting time of the job = $\max(0, \mathrm{t} - \mathrm{RJ})$
#	Random number from 0 to 1
NPT	processing time of the next operation
WINQ	work in the next queue
APR	average operation processing time of jobs in the queue
Function set	$+, -, \times$, %, min, max

*t is the time when the sequencing decision is made.

B. Representation

This paper uses the traditional GP tree to represent dispatching rules, the same as previous studies [5], [27], [31]. Table I shows the terminal set and function set used by GP to construct priority functions. Most attributes in the tables have been extensively used in the existing dispatching rules as well as GP for DJSS. For rrJ, rRJ, and rDD, they are commonly used in their absolute forms, i.e. release time, ready time, and due date. However, using these attributes in their absolute forms may cause some unexpected behaviours. For example, $-0.0001 \times releasetime - PR$ will behave like the shortest-processing-time rule when the release time of jobs are small (arrive at the early stage of the simulation), but will behave like the first-in-first-out rule when the release time increases (in the latter stage of the simulation). Therefore, we would like to use these attributes in their relative forms as shown in Table I. For the function set, four basic arithmetic operators and $\min / \max$ are used to construct composite dispatching rules (the protected division is similar to normal division but returns a value of 1 when division by 0 is attempted).

C. New surrogate models

In this paper, we develop a new surrogate model based on three criteria: (1) efficiency, (2) accuracy, and (3) generalisability. The first two criteria are to ensure that the proposed SGP can effectively find competitive rules efficiently. Regarding the generalisability, we expect that the proposed SGP can cope with various complex scheduling problems with minor modifications.

Rather than using the historical performance of evolved rules during the search for estimation [6], the surrogate model proposed in this study estimates fitness of the evolved rules by using a simplified simulation model of the original shop (see Section III). The goal here is to reduce the complexity of the original model up to the point where the simulation (evaluation) costs are low enough but still ensure an acceptable accuracy level. This study will examine two strategies for designing simplified models. First, the simplified model can be created by reducing the warmup and running time of the simulation [16]. This strategy does not require any modification to the original model. However, because of the simulation is shortened, the long term performance may not be accurately estimated and biased by the initial state of the system (i.e. empty shop in this case). The second strategy is to reduce the complexity of the original model by reducing the number of machines and the number of operations per job. By reducing the complexity of the shop, we can also reduce the warmup and running time of the simulation as the transition phase [16] (before the system becomes stable) will be shorter. Table II shows four simplified models investigated in this paper. *OriginalR1* is the original model in which the performance of rules are evaluated based on a single simulation replication. This model will be used as the fitness function in our SGP (see Fig. 1) to reduce the computational cost as it is shown to be helpful to improve the diversity of GP and the quality of evolved rules [27], [64]. Nevertheless, this model is still expensive and not suitable to be our surrogate model. *OriginalR1* is mainly considered here for reference purpose. *OriginalR1Short* is similar to *OriginalR1* but the simulation length is significantly reduced as shown in Table II (the first strategy). *HalfShop* and *MiniShop* are simplified versions (the second strategy) of the original shop where the number of machines/operations and simulation length is reduced (again, only one replication is used).

To select which is the most suitable to be the surrogate model for the scheduling problem under study, we will check how accurately the model measures the performance of scheduling rules. It is noted that these models cannot provide the *absolute* performance (e.g. the expected total weighted tardiness) of the original shop because they have been simplified and not statistically accurate (caused by a small number of replications and biases). Fortunately, what we need is the *relative* performance of evolved rules, i.e. which one is better, for genetic operations. Therefore, what we need is simply a way to identify the relationship between rules. Particularly, if rule a is better than rule b based on the *absolute* performance, the simplified model should also show that a is better than b. To know if the simplified model can effectively satisfy this property, we will check the rank-correlation between performance of rules using the original model and the performance of rules using the simplified model. Here is the proposed procedure:

1) Select a set of benchmark rules $B = \{\Delta_1, \Delta_2, \ldots, \Delta_N\}$
2) Apply B to the original model with a large number of simulation replications to obtain the *absolute* performance $\Upsilon = \{f(\Delta_1), f(\Delta_2), \ldots, f(\Delta_N)\}$ and the corresponding *rank* $\Upsilon_r = \{f_r(\Delta_1), f_r(\Delta_2), \ldots, f_r(\Delta_N)\}$ (sorted based on the descending order of $f(\cdot)$)
3) Apply the benchmark rules B to the simplified model to obtain the *relative* performance $\Psi = \{f'(\Delta_1), f'(\Delta_2), \ldots, f'(\Delta_N)\}$ and the corresponding *rank* $\Psi_r = \{f'_r(\Delta_1), f'_r(\Delta_2), \ldots, f'_r(\Delta_N)\}$ (sorted based on the descending order of $f'(\cdot)$)
4) Check the correlation between A and R, using the correlation coefficient:

$$\rho = \frac{\sum_{i=1}^{N} (f_r(\Delta_i) - \bar{f}_r)(f'_r(\Delta_i) - \bar{f}'_r)}{\sqrt{\sum_{i=1}^{N} (f_r(\Delta_i) - \bar{f}_r)^2 \sum_{i=1}^{N} (f'_r(\Delta_i) - \bar{f}'_r)^2}} \quad (1)$$

where is $\bar{f}_r$ and $\bar{f}'_r$ are the average performance across all benchmark rules from the original model and the simplified model respectively.

After the rank-correlation coefficients of all simplified models are determined, the one with the highest correlation will be used as the surrogate model for SGP. In this study, we use the set of benchmark dispatching rules are shown in Table III. These include various rules that are commonly found in the literature [14], [15], [43]. After applying the above procedure to the four models in Table II, the results are shown in Fig. 2.

It is clear that the *OriginalR1* model and the original model is well correlated. This observation is consistent with those from Hildebrandt et al. [27] in which they showed that using one simulation replication is sufficient for each generation of GP. The accuracy of *OriginalR1Short* model is worse than that of *OriginalR1*. Although the correlation coefficient is high, the ranking of benchmark rules are very different when *OriginalR1Short* model is used to determine the fitness. This is easy to understand as there are biases caused by short warmup phase and simulation length. The *HalfShop* model provides better *relative* performance as compared to *OriginalR1Short* given the same warmup phase and simulation length. It indicates that this model can well reflect the behaviour of the original model with lower complexity. The *MiniShop* model is overly simplified in this case and cannot approximate the fitness accurately. The results suggest that *HalfShop* model is the most suitable simplified model in our study.

It is noted that the surrogate (simplified) model is developed offline as a preprocessing step instead of online like the surrogate model proposed by [6]. The estimated fitness will be calculated directly from the simplified model instead of generating decision vector and match it with ones in the previous generations. The proposed procedure is not only helpful to select accurate and efficient simplified model but also general for different scheduling problems as it does not use any specific information about the scheduling process (e.g. the ranking of jobs).

TABLE II
SIMPLIFIED MODELS

Model	Description	#Machine	Max# of Operations	Simulation Length	Warmup Length
OriginalR1	The original model with one replication	10	14	5000	500
OriginalR1Short	Similar to OriginalR1 with short simulation time	10	14	500	100
HalfShop	The simplified shop is only half the scale of the original shop	5	7	500	100
MiniShop	The number of machines are kept minimum	2	4	250	50

TABLE III
BENCHMARK DISPATCHING RULES

SPT	shortest processing time	LPT	longest processing time
EDD	earliest due date	FDD	earliest flow due date
FIFO	first in first out	LIFO	last in first out
LWKR	least work remaining	MWKR	most work remaining
NPT	next processing time	WINQ	work in next queue
CR	critical ratio	AVPRO	average processing time/operation
MOD	modified due date	MOPNR	most operations remaining
SL	negative slack	Slack	slack
PW	process waiting time	RR	Raghu and Rajendran
ATC	apparent tardiness cost	COVERT	cost over time

OPFSLK/PT	operational flow slack per processing time
LWKR+SPT	least work remaining plus processing time
CR+SPT	critical ratio plus processing time
SPT+PW	processing time plus processing wating time
SPT+PW+FDD	SPT+PW plus earliest flow due date
Slack/OPN	slack per remaining operations
Slack/RPT+SPT	slack per remaining processing time plus processing time
PT+WINQ	procesting time plus work in next queue
2PT+WINQ+NPT	double processing time plus WINQ and NPT
PT+WINQ+SL	processing time plus WINQ and slack
PT+WINQ+NPT+WSL	PT+WINQ plus next processing time and waiting slack

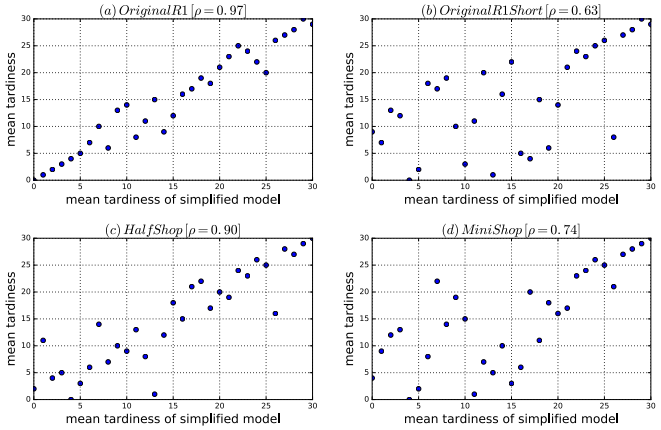


Fig. 2. Rank-correlation between simplified and original simulation models.

D. Overall algorithm

Fig. 3 shows the details of the our proposed SGP algorithm. In general, the basic components of this algorithms and the basic SGP discussed in Section IV-A are similar. The inputs of our models are the simulation model that we want to evolve dispatching rules (the original model) and the simplified model $\mathcal{S}$ presented in Section IV-C. Three types of fitness functions are used in different stages of the algorithms. As described in the previous section, the fitness $f(\Delta_i)$ is the real fitness (i.e. absolute performance) of evolved rules and $f'(\Delta_i)$ is the estimated fitness obtained with the simplified model $\mathcal{S}$. The fitness $f_g(\Delta_i)$ is the performance of rule Δ_i in a particular generation (with a specific replication π). Because of expensive simulation costs, using a large number of replications to obtain $f(\Delta_i)$ is impractical; therefore, we only use one replication per generation to evaluate the quality of evolved rules $f_g(\Delta_i)$. This strategy has been shown to be useful to improve the effectiveness of evolved rules and the diversity in the population [27], [64]. The rule with the best $f_g(\Delta_i)$ is then fully evaluated to obtain $f(\Delta_i)$. After an intermediate population generated based on the generation fitness $f_g(\Delta_i)$, the fitnesses of newly generated rules are quickly evaluated by using the simplified model $\mathcal{S}$. The use of these three fitness functions makes the algorithm slightly more complicated but they allows us to utilise the computational budgets more efficiently. The fitness $f'(\Delta_i)$ determines the rough quality of generated rules and helps the algorithm produce more potential rules. Meanwhile the fitness $f_g(\Delta_i)$ helps the algorithm identify the most potential rules for full evaluations and improve the diversity of SGP. Other parameters for the proposed SGP are shown in Table IV. These parameters have been used in our previous studies [43], [64] and tested in our pilot experiments. The intermediate population is k times larger than the original population to increase the diversity and improve the chance to find better rules as explained in the previous section. Three SGP versions investigated in this paper SGP_OS, SGP_HS, and SGP_MINI are based on the algorithm proposed in Fig.3 and the three simplified models, *OriginalR1Short*, *HalfShop*, and *MiniShop*, respectively.

TABLE IV
PARAMETER SETTINGS

Parameter	Description
Initialisation	ramped-half-and-half
Crossover/mutation/elitism rates	80%/15%/5%
Maximum depth	8
Number of generations	50
Population size	200
Size of intermediate population	200×5=1000
Selection	tournament selection (size = 5)

V. EXPERIMENTAL RESULTS

This section presents the results from the proposed SGP and other GP systems developed in the literature. We compare SGP_HS with SGP_OS and SGP_MINI to investigate whether the theoretical prediction is consistent with the empirical evidence. Also, we compare the proposed SGP methods with two GP methods proposed in the literature, GP and SGP_H, to demonstrate their effectiveness. GP is the common implementation for DJSS with the changing replication strategy [27], [64], and SGP_H is the surrogate assisted GP developed by Hildebrandt and Branke [6]. The five methods are compared in terms of the testing performance, the length of final rules/programs, and the running times. Each method performs 30 independent runs and the Wilcoxon signed-rank test with $\alpha = 0.05$ is used for our statistical significance test.

Inputs: simulation model of DJSS, simplified model $\mathcal{S}$
Output: the best evolved rule Δ^*

randomly initialise the population $P \leftarrow \{\Delta_1, \ldots, \Delta_n\}$
set $\Delta^* \leftarrow null$ and the fitness $f^* \leftarrow +\infty$
setup a set of replications $\mathcal{R}$ for full evaluation
$generation \leftarrow 0$, select a replication π
while $generation \leq maxGeneration$ **do**
    for all $\Delta_i \in P$ **do**
        evaluate $f_g(\Delta_i)$ by applying a Δ_i to π
        $f_g^* \leftarrow +\infty$ and $\Delta_g^* \leftarrow null$
        if $(f_g(\Delta_i) < f_g^*)$ **then**
            $\Delta_g^* \leftarrow \Delta_i$
            $f_g^* \leftarrow f_g(\Delta_i)$
        end if
    end for
    obtain $f(\Delta_g^*)$ by applying a Δ_g^* to $\mathcal{R}$
    if $(f(\Delta_g^*) < f_g^*)$ **then**
        $\Delta^* \leftarrow \Delta_g^*$
        $f^* \leftarrow f(\Delta_g^*)$
    end if
    $P' \leftarrow$ apply genetic operations to P †
    for all $\Delta_i \in P'$ **do**
        obtain $f'(\Delta_i)$ by applying a Δ_i to $\mathcal{S}$
    end for
    replace P with the top $|P|$ rules $\Delta_i \in P'$
    select a new π
    $generation \leftarrow generation + 1$
end while
return Δ^*

† $|P'| = k \times |P|$

Fig. 3. Proposed SGP algorithm.

The comparisons of five GP methods are shown in Fig. 4 to Fig. 9. For each figure, the first and second boxplots respectively show the testing performance measures and lengths (number of nodes) of rules obtained by the five GP methods. For presentation purposes, the total weighted tardiness is normalised in Figs. 8–9 and by dividing it by the number of (recorded) jobs.

A. Testing performance

Regarding the testing performance, it is clear that SGP methods outperform GP in all scenarios. The gaps between GP and SGP methods are about 3–5% for minimising mean tardiness, 5–6% for minimising maximum tardiness, and 10–20% for minimising total weighted tardiness. It seems that the surrogate methods are more powerful when the scheduling objectives are more complex. In general, SGP_HS and SGP_OS are significantly better than SGP_H in most cases except for the scenarios with utilisation of 85% and the objective is mean tardiness and maximum tardiness. However, this is easy to understand as these two scenarios are not too complicated (for tardiness related objective) and good rules can be found quite easily with SGP methods. SGP_HS and SGP_OS are very competitive and SGP_HS is significantly better than SGP_OS only in the scenarios with utilisation of 95% and maximum tardiness as the objective. SGP_MINI is quite competitive as compared to SGP_H but it is outperformed by SGP_HS and SGP_OS in the difficult scenarios with the utilisation of 95%.

B. Program length

In all scenarios, rules found by GP are significantly smaller than those found by SGP methods (except for SGP_MINI). However, as SGP methods are able to find better rules, it can be argued that the larger rules found by SGP is to help them cope with different complex situations. As compared to SGP_H, the proposed SGP methods discover more compact rules in most scenarios even in the cases when they outperform SGP_H in terms of testing performance. It shows that there can be many redundant genetic materials in rules evolved with SGP_H. It is interesting that the lengths of rules found by SGP_MINI and GP are not significantly different in most scenarios. One explanation is that the rules to dealing with the *MiniShop* model are not required to be complicated. Because genetic operations are applied based on the estimated fitness determined by *MiniShop*, the complexity of evolved rules is also controlled.

C. Running time

Comparing running times is a tricky issue for GP as compared to other evolutionary computation techniques as the programs/rules have variable length and the program evaluation costs (and fitness evaluations) also vary. In this study, all GP methods have the same computational budgets, i.e. the number of simulation replications as they are considered the most time consuming part in the algorithms. The differences in the running times are mainly caused by the lengths of evolved rules and the time to execute surrogate models. In all SGP methods, SGP_H and SGP_MINI are the fastest and not significantly different from GP in some scenarios. It is easy to understand as the times to execute surrogate models in these two methods are relatively short. SGP_HS and SGP_OS are the two most time consuming algorithms here because their surrogate models are more complicated. However, SGP_HS is significantly faster than SGP_OS in some scenarios (mostly because SGP_HS has evolved smaller rules).

The experimental results have demonstrated the effectiveness of our proposed SGP algorithms. In general, the proposed SGP methods are able to find better rules as compared to other methods in the literature. This suggests that the surrogate models proposed in this paper have successfully helped SGP identify potential rules. These results also show the importance of surrogate models, especially when dealing with complex scenarios. Another advantage of the proposed SGP methods is that it can help reduce the lengths of evolved rules. This is a crucial factor in order to make the evolved rules easy to understand and apply. Although running times of proposed SGP methods are longer, the facts that more compact and powerful rules are discovered have well justified the additional costs. Given that the real world systems can be a lot more complicated, these additional costs may become negligible. SGP_HS and SGP_OS are the two most promising methods in the experiments. SGP_HS seems to be slightly more competitive in terms of testing performance and running times.

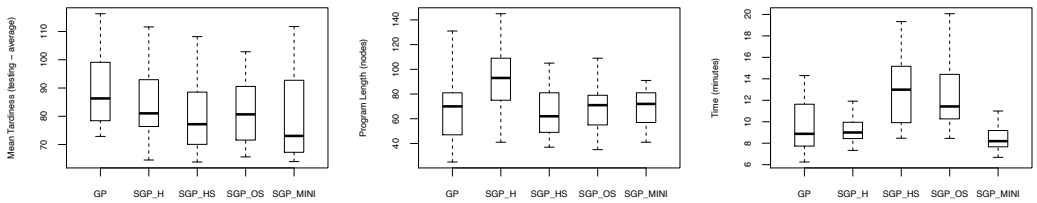


Fig. 4. Performance of GP methods - Minimise mean tardiness - Utilisation = 85%.

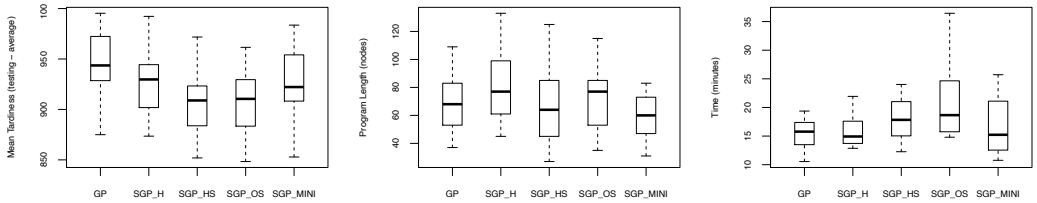


Fig. 5. Performance of GP methods - Minimise mean tardiness - Utilisation = 95%.

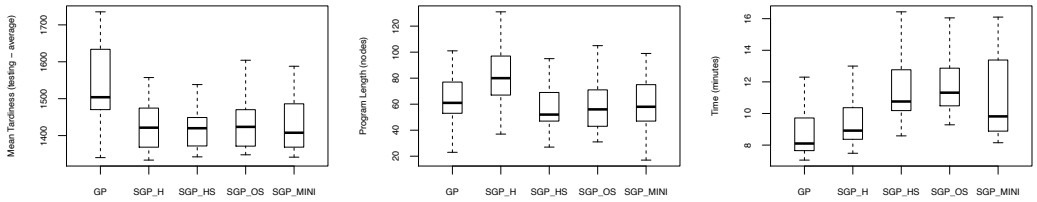


Fig. 6. Performance of GP methods - Minimise maximum tardiness - Utilisation = 85%.

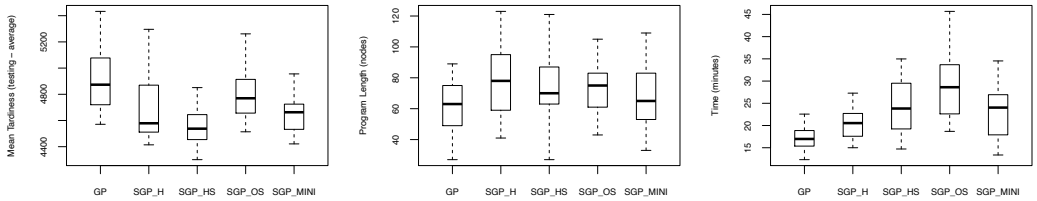


Fig. 7. Performance of GP methods - Minimise maximum tardiness - Utilisation = 95%.

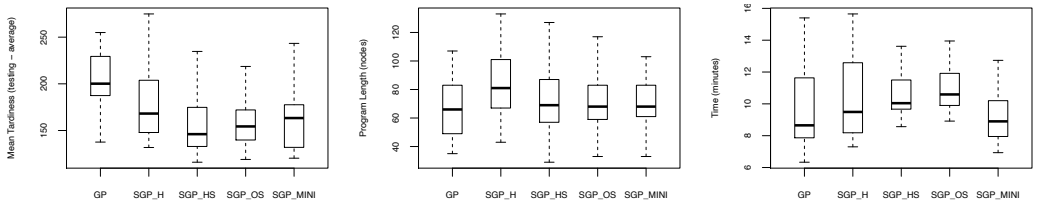


Fig. 8. Performance of GP methods - Minimise total weighted tardiness - Utilisation = 85%.

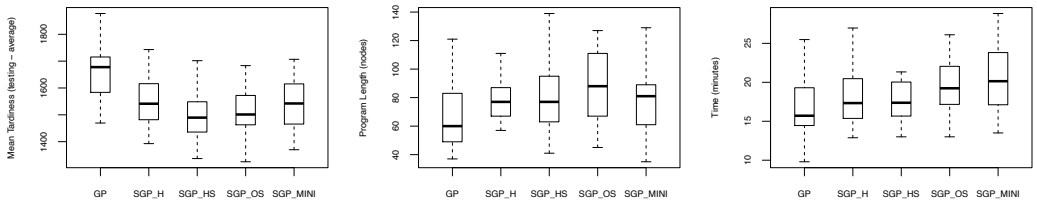


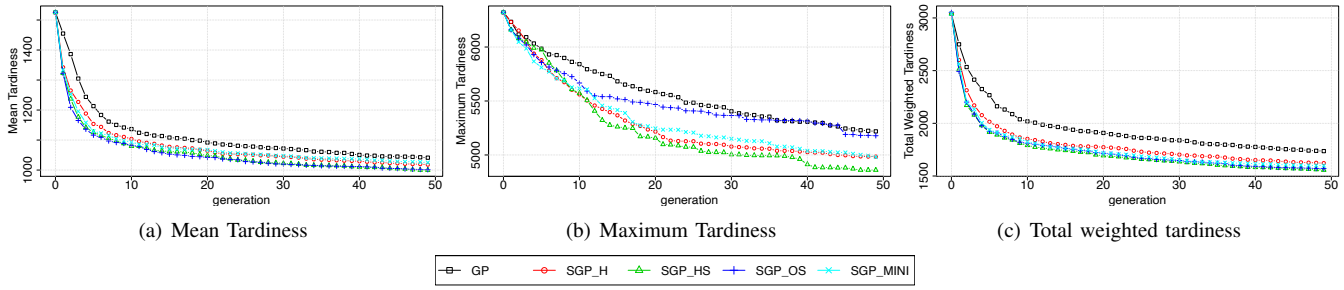
Fig. 9. Performance of GP methods - Minimise total weighted tardiness - Utilisation = 95%.

The next section will further investigate these methods to understand how they can effectively handle the problem.

VI. Further analysis

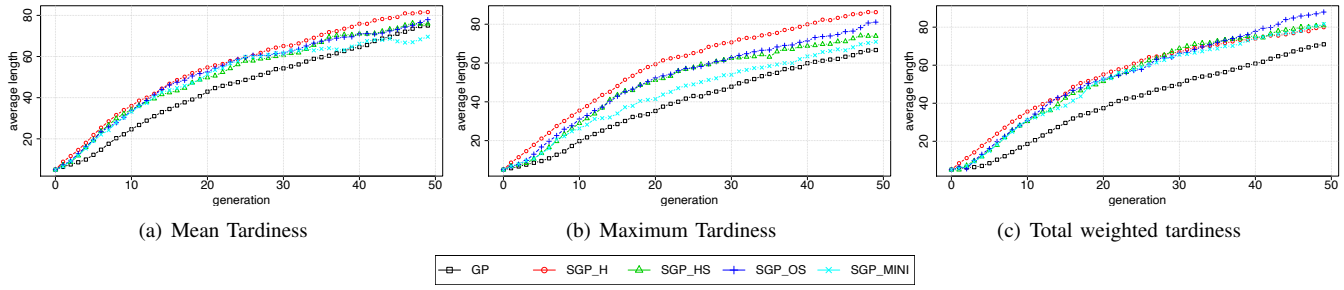
This section will look at the details during the evolution process of SGP methods. Fig. 10 and Fig. 11 show the average fitness $f^*(\Delta)$ and the average program length from all 30 independent runs and across generations when the utilisation is 95%.

The detailed results from Fig. 10 show that SGP methods can find good rules much faster than GP. It is easy to see that SGP methods only need half of the maximum generation to find the best solution obtained by GP. As discussed in the previous section, SGP_HS and SGP_OS make quite similar progress during the evolution (except for the case with maximum tardiness). It is noted that the progress of SGP methods is similar in the early stage of the evolution (the first ten generations); then their gaps become clear. This observation



(a) Mean Tardiness (b) Maximum Tardiness (c) Total weighted tardiness

Fig. 10. Fitness of rules across generations (utilisation = 95%).

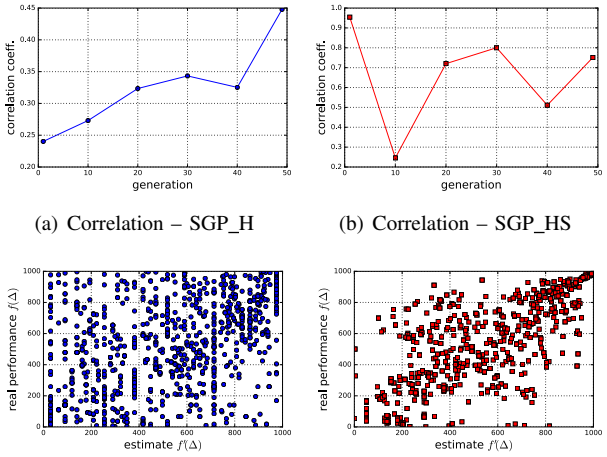


(a) Mean Tardiness (b) Maximum Tardiness (c) Total weighted tardiness

Fig. 11. Average length of rules across generations (utilisation = 95%).

is more obvious in Fig 10(b) when SGP_HS proceeds much better at the latter generations. Because the maximum tardiness requires a long simulation run to accurately estimate, SGP_OS is not appropriate as the simulation time of its simplified model is not long enough. This is a good example to show that choosing a suitable model to estimate fitness is extremely important.

SGP_H performs quite well in this case but it also progresses slowly at the later generations. This problem can occur since the surrogate model, similar to that of SGP_OS, becomes less accurate. To verify this assumption, we have compared the fitness estimated by the simplified model *HalfShop* and the absolute fitness/performance of SGP_H and SGP_HS. Because this analysis requires the real fitness of all generated rules which is very time-consuming (more than 2 days for one run), the result is only obtained for a particular run. Fig. 12 shows the rank-correlation between the estimated fitness and the real fitness during the evolution. In Figs 12(a) and (b), it is clear that the correlation coefficients in SGP_HS are much higher than those in SGP_H in most cases (noted that (a) and (b) use different scales for y axis). Actually, the correlation coefficients in SGP_H are low, which suggests that accuracy of its surrogate model is not high. An interesting pattern here is that the correlation coefficients of SGP_HS have the tendency to decrease in the early generations and go up at later generations. One possible explanation is that GP explores the search space intensively at the early stage and an accurate estimate will be difficult with a high diversity in the population (mostly bad programs/rules). In the later generations when more powerful rules are in the population, the relative performance between rules will be easier to determine. Figs 12(a) and (b) show the direct comparison between f_r and f'_r in the last generation. It is clear that SGP_H fails to estimate fitness accurately in this case (no clear correlation). SGP_HS estimates the fitness reasonably well. It explains why SGP_HS is very effective in this case.



(a) Correlation – SGP_H (b) Correlation – SGP_HS

(c) f vs. f' – SGP_H (GEN50) (d) f vs. f' – SGP_HS (GEN50)

Fig. 12. Accuracy of surrogate models - Minimise maximum tardiness.

Fig. 11 shows that the average program length grows quite fast at the early generations and its growing rate is smaller in the latter generations. For minimising maximum tardiness, it is very clear that SGP_H and SGP_OS evolves rules much larger than they should be, which make the fitness evaluations slower. SGP_HS performs quite well in most cases and it is able to maintain the length of evolved rules reasonably well. For minimising mean tardiness and maximum tardiness, the evolved rules of SGP_HS are as compact as GP. These analyses further support the experimental results and verify the effectiveness of the proposed SGP.

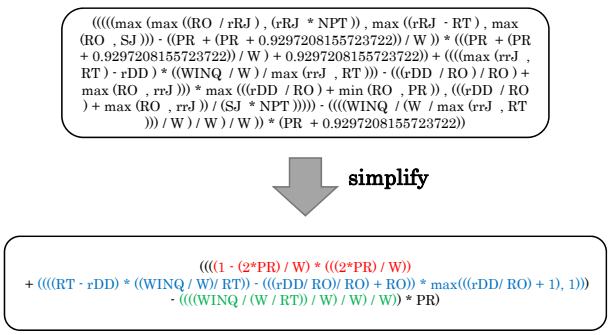


Fig. 13. Rule simplification.

VII. Simplification and Visualisation

Previous sections have provided insights about how SGP can evolve good rules. However, the rules evolved by proposed SGP methods tend to be larger than other rules evolved by standard GP. Large rules surely make the analyses and interpretation more difficult and restrict the applicability of SGP. In this section, we focus on gaining insights about evolved dispatching rules via simplification and visualisation.

A. Simplification

This issue has been investigated in other studies but it was mainly restricted to manual simplification. While this approach can make evolved rules more compact, it is quite a tedious process. To overcome this difficulty, we proposed here a new simple approach for simplification of dispatching rules:

1) Transverse through the program Δ from the root node
2) If the subtree has more than one node, replace it with the constant **1**. If the new rule Δ' is not significantly different from (or Δ' is better than) the original rule, $\Delta \leftarrow \Delta'$.
3) If current node is a constant, replace it with zero. If the new rule Δ' is not significantly different from (or Δ' is better than) the original rule, $\Delta \leftarrow \Delta'$.
4) Transverse to the next node and go to step 2.

To test this simplification approach, we use the best evolved rule for minimising total weighted tardiness as the example. The original rule and the simplified rule are shown in Fig. 13. It is obvious that the simplification process has eliminated redundant and complex components from the original rule. This simplification is simple and effective but it requires intensive search through the obtained rule. Thus, it is more suitable for the post-processing step rather than the online application during GP evolution.

B. Visualisation

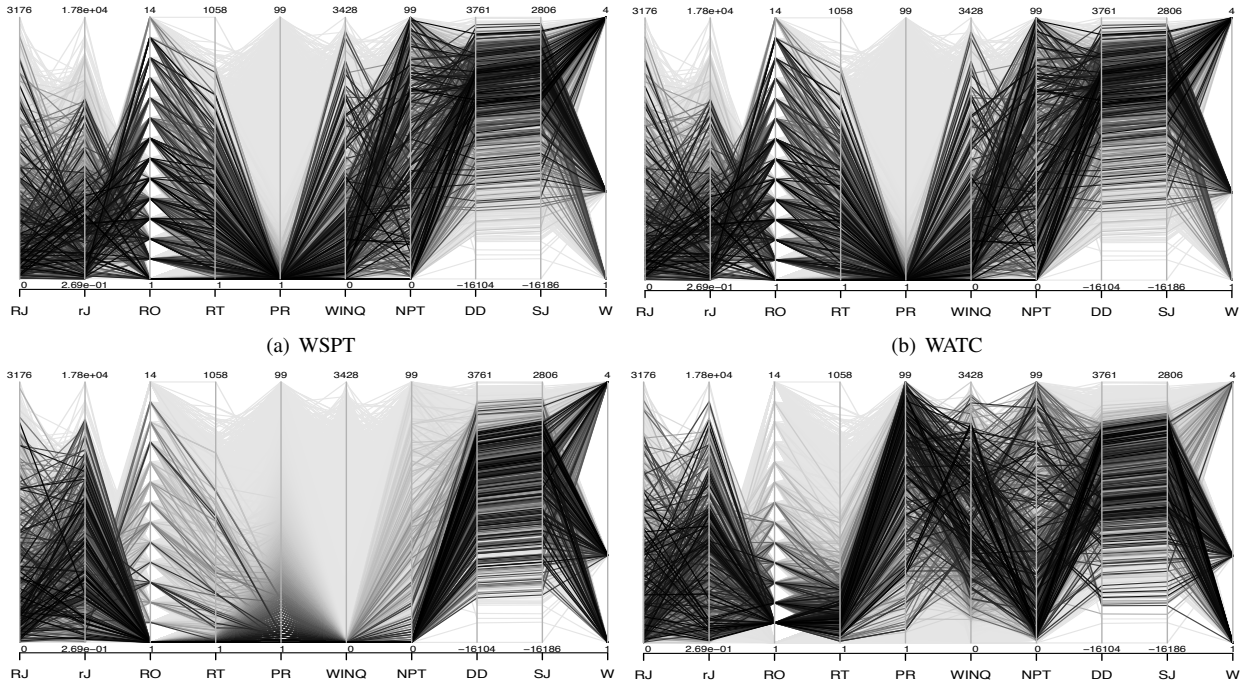
Although the simplified rule becomes more compact, it is still difficult to fully understand its behaviours. Working with the mathematical form of the evolved rules is tricky and sometimes not intuitive especially when these rules are discrete, non-smooth, and mutivariate. In this case, visualisation is needed to help explain the rules' behaviours. In this study, we use the parallel coordinate plot to visualise dispatching rules. To draw this plot, we will take a large sample of job attributes and their corresponding priority determined by the considered rule through simulation. Each attribute vector (for a particular job) will be represented by a line connecting all coordinates (for all attributes). The darkness of each line will depend the priority (higher priority will lead to darker line). To improve the readability, we normalise priorities p to $[0, 1]$ and the darkness will be determined by p^β. When we increase β, we will focus more on the attribute vectors with high priorities. Using the scenario with the utilisation of 95%, Fig. 14 shows the parallel coordinate plots of three well-known dispatching rules [7], weighted shortest processing time (WSPT[1]), weighted apparent tardy cost (WATC[2]), 2PT+WINQ+NPT, and the evolved rule in Fig.13 with $\beta = 30$. From the plots, it is easy to realise that the behaviours of WSPT and WATC are very similar even though WATC is much more complicated. The reason is that WATC will reduce to WSPT as the shop becomes crowded (high utilisation). In this case, the component in the exponential function become zero. This may not be easily observed based on the mathematical formula of WATC but it is very clear in the parallel coordinate plot. Similarly, the plot shows that 2PT+WINQ+NPT puts more emphasis on jobs with low PT (processing time), NPT and WINQ. The evolved rule is undoubtedly the most sophisticated one among the four rules presented in Fig. 14. Jobs with the highest priorities are the ones with low RT, RO, W and high PR. It is a bit counterintuitive as compared to WSPT and WATC. Late jobs (with negative rDD) are also given high priorities by this evolve rule. This property helps the evolved rule reduce the tardiness of jobs. However, the parallel coordinate plot is not sufficient to show the rule's behaviour.

To gain more insights about the rule, we will further investigate how the rule decides which job should have higher priorities. Using the sample in Fig. 14, we create a dataset for a classification problem. From the above sample, we randomly select two attribute vectors $v_a = (rrJ_a, rRj_a, \ldots, W_a)$ and $v_b = (rrJ_b, rRj_b, \ldots, W_b)$ and their corresponding priorities $priority_a$ and $priority_b$. The features in the dataset will include the relative values of attributes from jobs a and b, and the label is 1 if $priority_a > priority_b$ (0 otherwise). For example, if $W_a = 1$ and $W_b = 4$, the relative feature value will be $1/(1+4) = 0.2$ (if the two job attributes are zero, the relative value will be 0.5). For attributes with both positive and negative values such as SJ, the relative feature value is 0 when $SJ_a < 0$ and $SJ_b \geq 0$; or 1 when $SJ_a \geq 0$ and $SJ_b < 0$. If both SJ_a and SJ_b are positive, the relative feature value is $= SJ_a/(SJ_a+SJ_b)$. If the two values are negative, the relative feature value is $= SJ_b/(SJ_a + SJ_b)$. We apply decision tree learning to build the classifier for this dataset. The goal here is to use decision trees for identifying key attributes in the evolved rules and provides insights into their behaviours. The decision trees for WATC and the evolved rule are shown in Fig. 15 and Fig. 16 respectively. We use the maximum depth of 5 for these rules to make them more compact.

For WATC, the decision tree achieves very high accuracy

[1] $priority_{wspt} = \frac{w_i}{p_{ij}}$

[2] $priority_{watc} = \frac{w_i}{p_{ij}} exp(\max(\frac{d_j - t - p_{ij} - \sum_{q=j}^{N_i} W_{iq} + p_{iq}}{k\bar{p}}, 0))$ where w_i, p_{ij}, d_i, W_{iq}, $\bar{p}$ are the weight, processing time of operation j of job i, due date, waiting time, and average processing time respectively



(a) WSPT (b) WATC

(c) 2PT+WINQ+NPT (d) GPrule

Fig. 14. Parallel coordinate plots of dispatching rules (sample size = 10000).

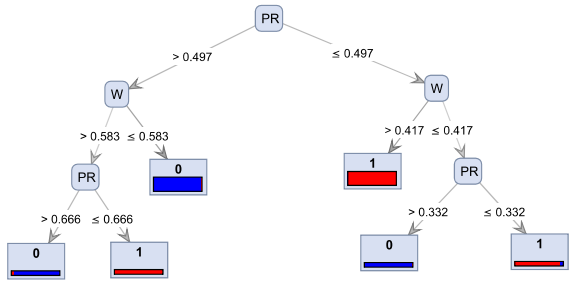


Fig. 15. Decision tree based on the WATC rule. For the two jobs a and b, the decision is **1** if a has a higher priority than b and **0** otherwise.

(about 99%). The behaviour of WSPT is also observed in the decision tree. Generally, jobs with low PR and high W will have higher priorities. Unfortunately, the evolved rules are much more complex to be represented by a decision tree. The decision tree generated with the dataset of the evolved rule only achieve a modest accuracy of 76%. However, we can see that the behaviour of WSPT is also partly reflected in the decision tree. Again, we cannot understand the evolved rule fully with the decision tree. One thing we can learn from the decision tree is which features/attributes are more likely to decide the priorities of jobs. For the evolved rules, they are RT, W, PR and WINQ. Opposite to what we have seen in Fig. 14(d), the attributes rDD are shown in the decision tree. The reason is that rDD is not the most powerful feature to decide which job has a higher priority (also there is no obvious pattern for rDD in Fig. 14(d)).

Based on the knowledge obtained from the parallel coordinate plot and the decision tree, we will analyse the evolved rule closely. To make the rule easier to understand, we rearrange the rule as in Fig.17. We see that the priority calculated by the evolved rule are the sum of three components. The first two components are variants of WSPT with different emphasis on W and PR. The second component also considers WINQ and RT and it is quite sensitive to W. The third component is the most complicated one consisted of many attributes. As RT is the most important attribute in the decision tree (Fig. 16), we will interpret the rule centered around this attribute.

If RT is equal to rDD, the third component will reduce to -(((rDD/ RO)/ RO) + RO)) * max(((rDD/ RO) + 1), 1)*PR. Because the RT is positive, rDD will be positive and the job is not late. Therefore the third component is negative and behave like a variant of SPT. However, as rDD is divided by RO^2, the impact of the third component will be small and the evolved rule will be mainly governed by the first and second components. If WINQ is also zero, only the first component will remain (will be similar to WSPT). If WINQ is positive, the rule will give high priorities for jobs with low WINQ, RT and high W. This helps the shop finish short jobs with high weights faster. The rule will behave similarly when RT is large and rDD is positive (but smaller than RT).

If RT is small and rDD is negative (the job is late), the third component will be positive and this component as well as the whole rule will be governed mainly by rDD. It means that jobs that are behind the due date further will have higher priorities. As W and PR are also in the third components when they are positive, their effects are reversed as compared to WSPT (as previously shown in Fig. 14(d)). A possible explanation is that it will help reduce the effects of WSPT when jobs become too late.

Our analyses have shown that the evolved rules possess very interesting and sophisticated properties to help minimise

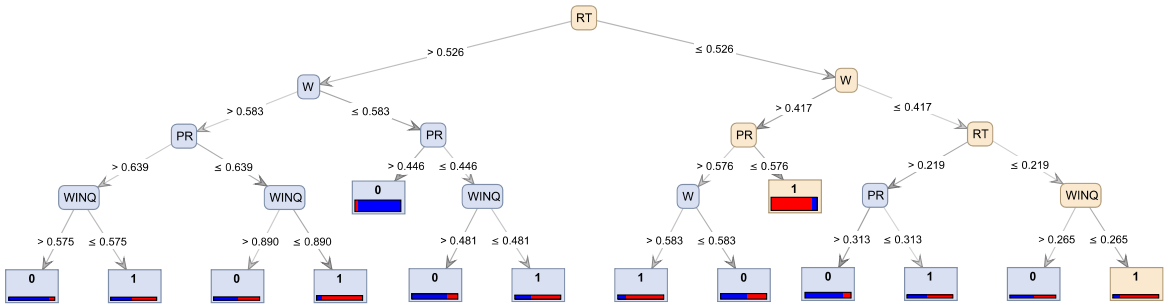


Fig. 16. Decision tree based on the evolved rule. For the two jobs a and b, the decision is **1** if a has a higher priority than b and **0** otherwise.

1:	- (2*PR*PR) / W * (((2*PR) / W))
2:	- WINQ*RT*PR/W^4
3:	+ ((((RT - rDD) * ((WINQ / W)/ RT)) - (((rDD/ RO)/ RO) + RO)) * max(((rDD/ RO) + 1), 1))*PR

Fig. 17. Rule simplification.

the total weighted tardiness. The simplification, visualisation, and decision trees have provided useful insights to help us interpret the rules. We believe that data analytics will play a very important role to enhance the applicability of evolved rules as well as GP. An automated design system should have more analytics tools to facilitate the analyses in the future.

VIII. Conclusions

Automated design of dispatching rules for production systems is an interesting and difficult task. Although, GP has achieved some encouraging results, building an applicable GP system for practical problems is often challenging due to issues such as flexibility, effectiveness, efficiency, and interpretability. In this paper, we have successfully developed a new surrogate assisted GP system for evolving dispatching rules. The novelty of this system lies on the surrogate models based on simplified simulation models of the original shop and the new SGP algorithm that combines multiple fitness functions. We have shown that the proposed algorithm can be extended to cope with different scheduling problems. The experimental results have also verified the effectiveness of our algorithms as compared to other results reported in the literature. The proposed algorithm outperforms other existing algorithms in terms of rule performance. In some cases, lengths of the evolved rules and running times of the algorithms are also reduced. The analyses have also demonstrated the accuracy of the proposed surrogate models and their behaviours through the evolution process of GP. As compared to the existing surrogate model, the new model based on a simplified simulation model of the original shop gives better estimates across generations.

In future studies, we want to investigate how our surrogate models can cope with more complex/practical environments. Surrogate models should be still useful in these case but they need to be used with more general and powerful search mechanisms in GP such as coevolution [28], [38], two-stage approach [42], and multiple-tree GP [37]. Also, surrogate models can be used to improve the quality of EAs [48] search so studies on these models for automated design of scheduling rules are still important. Moreover, it is important to develop a more general and powerful pre-processing subroutine to help improve the automation of the model selection steps. To improve the applicability and intepretability of GP for scheduling problems, visualisation and data analytics techniques have been applied. We have shown that these techniques are useful for the interpretation of the evolved rules. We believe that these techniques/tools should be part of the automated design process and more rigorous research should be carried out in order to develop a guideline for applying these techniques in a systematic way in practice.

References

[1] J. Branke, S. Nguyen, C. Pickardt, and M. Zhang, "Automated design of production scheduling heuristics: A review," *IEEE Transactions on Evolutionary Computation*, 2015, (to appear).
[2] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.
[3] U. Bhowan, M. Johnston, and M. Zhang, "Developing new fitness functions in gp for classification with unbalanced data," *IEEE Transactions on Cybernetics*, vol. 42, no. 2, pp. 406–421, 2012.
[4] D. Harvey and M. Todd, "Automated feature design for numeric sequence classification by genetic programming," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 4, pp. 474–489, 2015.
[5] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 621–639, 2013.
[6] T. Hildebrandt and J. Branke, "On using surrogates with genetic programming," *Evolutionary Computation*, 2014, in press (doi: 10.1162/EVCO_a_00133).
[7] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. New York: Springer, 2008.
[8] K. N. McKay, F. R. Safayeni, and J. A. Buzacott, "Job-shop scheduling theory: What is relevant?" *Interfaces*, vol. 18, pp. 84–90, 1988.
[9] M. E. Pfund, S. J. Mason, and J. W. Fowler, "Semiconductor manufacturing scheduling and dispatching: state of the art and survey of needs," in *Handbook of Production Scheduling*, 2006, vol. 89, pp. 213–241.
[10] S. C. Sarin, A. Varadarajan, and L. Wang, "A survey of dispatching rules for operational control in wafer fabrication," *Production Planning & Control*, vol. 22, no. 1, pp. 4–24, 2011.
[11] B. Chen and T. I. Matis, "A flexible dispatching rule for minimizing tardiness in job shop scheduling," *International Journal of Production Economics*, vol. 141, no. 1, pp. 360–365, 2013.
[12] A. Otto and C. Otto, "How to design effective priority rules: Example of simple assembly line balancing," *Computers & Industrial Engineering*, vol. 69, pp. 43–52, 2014.
[13] V. Sels, N. Gheysen, and M. Vanhoucke, "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*, vol. 50, no. 15, pp. 4255–4270, 2011.
[14] M. S. Jayamohan and C. Rajendran, "New dispatching rules for shop scheduling: a step forward," *International Journal of Production Research*, vol. 38, pp. 563–586, 2000.

[15] O. Holthaus and C. Rajendran, "Efficient jobshop dispatching rules: further developments," *Production Planning & Control*, vol. 11, no. 2, pp. 171–178, 2000.

[16] A. M. Law and D. M. Kelton, *Simulation Modeling and Analysis*, 3rd ed. McGraw-Hill Higher Education, 1999.

[17] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Automatic programming via iterated local search for dynamic job shop scheduling," *IEEE Transactions on Cybernetics*, vol. 45, no. 1, pp. 1–14, Jan 2015.

[18] X. Li and S. Olafsson, "Discovering dispatching rules using data mining," *Journal of Scheduling*, vol. 8, no. 6, pp. 515–527, 2005.

[19] Z. Ren, H. Jiang, J. Xuan, Y. Hu, and Z. Luo, "New insights into diversification of hyper-heuristics," *IEEE Transactions on Cybernetics*, vol. 44, no. 10, pp. 1747–1761, Oct 2014.

[20] N. Sabar, M. Ayob, G. Kendall, and R. Qu, "Automatic design of a hyper-heuristic framework with gene expression programming for combinatorial optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 19, no. 3, pp. 309–325, June 2015.

[21] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and J. R. Woodward, "A classification of hyper-heuristic approaches," in *Handbook of Metaheuristics*, 2nd ed., 2010, vol. 146, pp. 449–468.

[22] N. Sabar, M. Ayob, G. Kendall, and R. Qu, "A dynamic multiarmed bandit-gene expression programming hyper-heuristic for combinatorial optimization problems," *IEEE Transactions on Cybernetics*, vol. 45, no. 2, pp. 217–228, Feb 2015.

[23] T. Eguchi, F. Oba, and S. Toyooka, "A robust scheduling rule using a neural network in dynamically changing job-shop environments," *International Journal of Manufacturing Technology and Management*, vol. 14, no. 3–4, pp. 266–288, 2008.

[24] H. Ingimundardottir and T. P. Runarsson, "Supervised learning linear priority dispatch rules for job-shop scheduling," in *Learning and Intelligent Optimization*, ser. LNCS, 2011, vol. 6683, pp. 263–277.

[25] M. Kapanoglu and M. Alikalfa, "Learning IF-THEN priority rules for dynamic job shops using genetic algorithms," *Robotics and Computer-Integrated Manufacturing*, vol. 27, no. 1, pp. 47–55, 2011.

[26] M. Kofler, S. Wagner, A. Beham, G. Kronberger, and M. Affenzeller, "Priority rule generation with a genetic algorithm to minimize sequence dependent setup costs," in *Computer Aided Systems Theory — EUROCAST 2009*, ser. LNCS, 2009, vol. 5717, pp. 817–824.

[27] T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Towards improved dispatching rules for complex shop floor scenarios — a genetic programming approach," in *GECCO'10*, 2010, pp. 257–264.

[28] K. Miyashita, "Job-shop scheduling with genetic programming," in *GECCO'00*, 2000, pp. 505–512.

[29] D. Jakobović and L. Budin, "Dynamic scheduling with genetic programming," in *Genetic Programming*, ser. LNCS, 2006, vol. 3905, pp. 73–84.

[30] J. C. Tay and N. B. Ho, "Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems," *Computers & Industrial Engineering*, vol. 54, no. 3, pp. 453–473, 2008.

[31] C. D. Geiger, R. Uzsoy, and H. Aytuğ, "Rapid modeling and discovery of priority dispatching rules: an autonomous learning approach," *Journal of Scheduling*, vol. 9, no. 1, pp. 7–34, 2006.

[32] J. Branke, T. Hildebrandt, and B. Scholz-Reiter, "Hyper-heuristic evolution of dispatching rules: a comparison of rule representations," *Evolutionary Computation*, 2014, in press (doi: 10.1162/EVCO_a_00131).

[33] E. K. Burke, M. R. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. R. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Computational Intelligence*, 2009, vol. 1, pp. 177–201.

[34] C. D. Geiger and R. Uzsoy, "Learning effective dispatching rules for batch processor scheduling," *International Journal of Production Research*, vol. 46, no. 6, pp. 1431–1454, 2008.

[35] W.-J. Yin, M. Liu, and C. Wu, "Learning single-machine scheduling heuristics subject to machine breakdowns with genetic programming," in *CEC'03*, 2003, vol. 2, pp. 1050–1055.

[36] L. Atlan, J. Bonnet, and M. Naillon, "Learning distributed reactive strategies by genetic programming for the general job shop problem," in *Proceedings of the Seventh Florida Artificial Intelligence Research Symposium*, D. D. Dankel, II and J. H. Stewman, Eds., 1994.

[37] D. Jakobović, L. Jelenković, and L. Budin, "Genetic programming heuristics for multiple machine scheduling," in *Genetic Programming*, ser. LNCS, 2007, vol. 4445, pp. 321–330.

[38] 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.

[39] ——, "Genetic programming for evolving due-date assignment models in job shop environments," *Evolutionary Computation*, vol. 22, no. 1, pp. 105–138, 2014.

[40] A. Beham, S. Winkler, S. Wagner, and M. Affenzeller, "A genetic programming approach to solve scheduling problems with parallel simulation," in *Proceedings of the 2008 IEEE International Parallel & Distributed Processing Symposium*, 2008, pp. 1–5.

[41] M. Affenzeller and S. Wagner, "Sasegasa: A new generic parallel evolutionary algorithm for achieving highest quality results," *Journal of Heuristics*, vol. 10, no. 3, pp. 243–267, May 2004.

[42] C. W. Pickardt, T. Hildebrandt, J. Branke, J. Heger, and B. Scholz-Reiter, "Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems," *International Journal of Production Economics*, vol. 145, no. 1, pp. 67–77, 2013.

[43] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Dynamic multi-objective job shop scheduling: a genetic programming approach," in *Automated Scheduling and Planning*, 2013, vol. 505, pp. 251–282.

[44] K.-Y. Hashimoto, "A reduced model for fast and accurate simulation of surface acoustic wave devices," in *IEEE International in Ultrasonics Symposium (IUS)*, July 2013, pp. 1399–1402.

[45] Z. Bai, "Krylov subspace techniques for reduced-order modeling of large-scale dynamical systems," *Applied Numerical Mathematics*, vol. 43, no. 12, pp. 9–44, 2002.

[46] M. Rewieski and J. White, "Model order reduction for nonlinear dynamical systems based on trajectory piecewise-linear approximations," *Linear Algebra and its Applications*, vol. 415, no. 23, pp. 426–454, 2006.

[47] A. Treuille, A. Lewis, and Z. Popović, "Model reduction for real-time fluids," *ACM Trans. Graph.*, vol. 25, no. 3, pp. 826–834, Jul. 2006.

[48] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Comput*, vol. 9, no. 1, pp. 3–12, 2005.

[49] ——, "Surrogate-assisted evolutionary computation: Recent advances and future challenges," *Swarm and Evolutionary Computation*, vol. 1, no. 2, pp. 61–70, 2011.

[50] J. Branke and C. Schmidt, "Faster convergence by means of fitness estimation," *Soft Computing*, vol. 9, no. 1, pp. 13–20, 2005.

[51] V. Oduguwa and R. Roy, "Multi-objective optimisation of rolling rod product design using meta-modelling approach," in *GECCO '02*, 2002, pp. 1164–1171.

[52] Y.-S. Hong, H. Lee, and M.-J. Tahk, "Acceleration of the convergence speed of evolutionary algorithms using multi-layer neural networks," *Engineering Optimization*, vol. 35, no. 1, pp. 91–102, 2003.

[53] A.-C. Zvoianu, G. Bramerdorfer, E. Lughofer, S. Silber, W. Amrhein, and E. P. Klement, "Hybridization of multi-objective evolutionary algorithms and artificial neural networks for optimizing the performance of electrical drives," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 8, pp. 1781 – 1794, 2013.

[54] M. Farina, "A neural network based generalized response surface multiobjective evolutionary algorithm," in *CEC '02*, vol. 1, May 2002, pp. 956–961.

[55] I. Loshchilov, M. Schoenauer, and M. Sebag, "Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy," in *GECCO '12*, 2012, pp. 321–328.

[56] T. Runarsson, "Constrained evolutionary optimization by approximate ranking and surrogate models," in *PPSN VIII*, ser. LNCS, 2004, vol. 3242, pp. 401–410.

[57] L. Fonseca, I. Barbosa, and A. Lemonge, "A similarity-based surrogate model for expensive evolutionary optimization with fixed budget of simulations," in *CEC'09*, May 2009, pp. 867–874.

[58] T. Goel, R. Haftka, W. Shyy, and N. Queipo, "Ensemble of surrogates," *Structural and Multidisciplinary Optimization*, vol. 33, no. 3, pp. 199–216, 2007.

[59] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 3, pp. 329–355, June 2010.

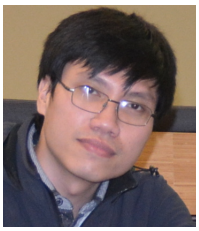
[60] D. Lim, Y.-S. Ong, Y. Jin, and B. Sendhoff, "A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation," in *GECCO'07*, 2007, pp. 1288–1295.

[61] X. Zhou, Y. Ma, Y. Tu, and Y. Feng, "Ensemble of surrogates for dual response surface modeling in robust parameter design," *Quality and Reliability Engineering International*, vol. 29, no. 2, pp. 173–197, 2013.

[62] S. Kreipl, "A large step random walk for minimizing total weighted tardiness in a job shop," *J. of Scheduling*, vol. 3, pp. 125–138, 2000.

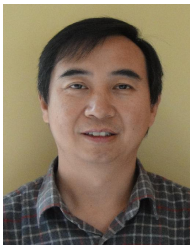
[63] M. L. Pinedo and M. Singer, "A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop," *Naval Research Logistics*, vol. 46, no. 1, pp. 1–17, 1999.

[64] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Selection schemes in surrogate-assisted genetic programming for job shop scheduling," in *Simulated Evolution and Learning*, 2014, vol. 8886, pp. 656–667.



Su Nguyen (M'13) received the B.E. degree in Industrial and Systems Engineering from the Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam, in 2006, the M.E. degree in Industrial Engineering and Management from the Asian Institute of Technology (AIT), Bangkok, Thailand, in 2008, and the Ph.D. degree in Operations Research and Data Analytics from Victoria University of Wellington (VUW), Wellington, New Zealand, in 2013.

He is currently the Research Fellow in Evolutionary Computation Research Group (ECRG), VUW and the lecturer at Hoa Sen University (HSU), Vietnam. Prior to VUW, he was a Research Associate in Industrial and Manufacturing Engineering at the School of Engineering and Technology, AIT. His primary research interests include evolutionary computation, discrete-event simulation and their applications in operations planning and scheduling.



Mengjie Zhang (M04-SM10) received the B.E. and M.E. degrees from Artificial Intelligence Research Center, Agricultural University of Hebei, Hebei, China, and the Ph.D. degree in computer science from RMIT University, Melbourne, VIC, Australia, in 1989, 1992, and 2000, respectively.

Since 2000, he has been with the Victoria University of Wellington, Wellington, New Zealand, where he is currently Professor of Computer Science, Head of the Evolutionary Computation Research Group, and the Associate Dean (Research and Innovation) in the Faculty of Engineering. His current research interests include evolutionary computation, particularly genetic programming, particle swarm optimization, and learning classifier systems with application areas of image analysis, multiobjective optimization, classification with unbalanced data, feature selection and reduction, and job shop scheduling. He has published over 350 academic papers in refereed international journals and conferences.

Prof. Zhang has been serving as an Associated Editor or Editorial Board Member for five international journals (including IEEE Transactions on Evolutionary Computation, Evolutionary Computation Journal) and as a Reviewer of over 20 international journals. He has been serving as a Steering Committee Member and a Program Committee Member for over 80 international conferences. He has supervised over 50 postgraduate research students. He is the Chair of the IEEE CIS Evolutionary Computation Technical Committee, a member of the IEEE CIS Intelligent Systems and Applications Technical Committee, a Vice-Chair of the IEEE CIS Task Force on Evolutionary Computer Vision and Image Processing, a Vice-Chair of the IEEE CIS Task Force on Evolutionary Computation for Feature Selection and Construction, a member of IEEE CIS Task Force of Hyper-heuristics, and the Founding Chair for IEEE Computational Intelligence Chapter in New Zealand.

Kay Chen Tan (SM'08-F'14) received the B.Eng. degree (First Class Honors) in electronics and electrical engineering and the Ph.D. degree from the University of Glasgow, Glasgow, Scotland, U.K., in 1994 and 1997, respectively.

He is currently an Associate Professor with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore. He has published over 100 journal papers, over 100 papers in conference proceedings, co-authored five books, and co-edited four books.

Dr Tan is a Fellow of IEEE for the class of 2014. He served as the General Co-Chair for the IEEE Congress on Evolutionary Computation, Singapore, in 2007, and the General Co-Chair for the IEEE Symposium on Computational Intelligence in Scheduling, Tennessee, in 2009. He has been an IEEE Distinguished Lecturer of the IEEE Computational Intelligence Society since 2011. He is currently the Editor-in-Chief of the IEEE Computational Intelligence Magazine. He also serves as an Associate Editor and Editorial Board member of over 15 international journals. He was a recipient of the 2012 IEEE Computational Intelligence Society Outstanding Early Career Award. He was a recipient of the Recognition Award in 2008 from the International Network for Engineering Education and Research (iNEER). He was a recipient of the NUS Outstanding Educator Award in 2004, the Engineering Educator Award in 2002, 2003, and 2005, the Annual Teaching Excellence Award in 2002, 2003, 2004, 2005, and 2006, and the Honor Roll Award in 2007.