Automatic Design of Scheduling Policies for Dynamic Multi-objective Job Shop Scheduling via Cooperative Coevolution Genetic Programming

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

***Abstract*—A scheduling policy strongly influences the performance of a manufacturing system. However, the design of an effective scheduling policy is complicated and time-consuming due to the complexity of each scheduling decision as well as the interactions among these decisions. This paper develops four new multi-objective genetic programming based hyper-heuristic (MO-GPHH) methods for automatic design of scheduling policies including dispatching rules and due-date assignment rules in job shop environments. Besides using three existing search strategies NSGA-II, SPEA2 and HaD-MOEA to develop new MO-GPHH methods, a new approach called Diversified Multi-Objective Cooperative Coevolution (DMOCC) is also proposed. The novelty of these MO-GPHH methods is that they are able to handle multiple scheduling decisions simultaneously. The experimental results show that the evolved Pareto fronts represent effective scheduling policies that can dominate scheduling policies from combinations of existing dispatching rules with dynamic/regression-based due-date assignment rules. The evolved scheduling policies also show dominating performance on unseen simulation scenarios with different shop settings. In addition, the uniformity of the scheduling policies obtained from the proposed method of DMOCC is better than those evolved by other evolutionary approaches.**

***Index Terms*—Genetic Programming, job shop scheduling, hyper-heuristic, dispatching rule.**

NOMENCLATURE

JSS	job shop scheduling
DJSS	dynamic job shop scheduling
DR	dispatching rule
CDR	composite dispatching rule
DDAR	due-date assignment rule
SP	scheduling policy
GP	genetic programming
GPHH	genetic programming based hyper-heuristic
MO-GPHH	multi-objective GPHH
SPEA2	strength Pareto evolutionary algorithm 2
NSGA-II	non-dominated sorting genetic algorithm II
HaD-MOEA	harmonic distance based multi-objective evolutionary algorithm
DMOCC	diversified multi-objective cooperative co-evolution

Su Nguyen, Mengjie Zhang, and Mark Johnston are with the Evolutionary Computation Research Group at Victoria University of Wellington, PO Box 600, Wellington, New Zealand. Kay Chen Tan is with the Department of Electrical and Computer Engineering, National University of Singapore, 4 Engineering Drive 3, 117576, Singapore.

NOTATION

p_j	total processing time of job j
d_j	due-date of job j
w_j	weight of job j
m_j	number of operations of job j
r_j	release time of job j
C_j	completion time of job j
f_j	flowtime of job j
$\hat{f}_j$	estimated (predicted) flowtime of job j
e_j	error in flowtime estimation of job j
σ	considered operation
δ	machine that processes σ
$p(\sigma)$	processing time of operation σ
$\hat{f}_o$	estimated operation flowtime of σ

I. INTRODUCTION

JOB shop scheduling is a crucial issue in the made-to-order manufacturing industry because it directly influences the performance of manufacturing systems. In the literature, it has been commonly assumed that JSS is equivalent to sequencing, which determines the order in which waiting jobs are processed on a set of machines in a manufacturing system [1]. A large number of studies on JSS mainly focus on the sequencing part [2], [3], [4], [5], [6], [7], [8]. The aim of these studies is to find an optimal schedule to satisfy a specific criterion (e.g. minimising makespan), given a set of jobs waiting to be processed. However, sequencing is only one of several scheduling decisions within a comprehensive scheduling system. One of the other important activities in JSS is due-date assignment (DDA), sometimes referred to as the estimation of job flowtimes. The objective of this activity is to determine the due-dates for arriving jobs by estimating the job flowtimes (the time taken from the arrival until the completion of a job), and therefore DDA strongly influences the delivery performance, i.e., ability to meet the promised delivery dates of a job shop [9].

Most studies on job shop scheduling only consider one of the many decisions and fix the others in order to reduce the complexity of the scheduling problems. These approaches are valid when there is no interaction among the scheduling decisions, which is often not the case for real world applications. Although JSS has been popular for decades and investigation of the interactions among various decisions is essential for the development of effective and comprehensive scheduling systems, studies on the interactions among different scheduling decisions are rather limited. These studies [10], [11], [12], [13] mainly examined the performance of simple combinations of different existing dispatching rules (DRs) and due-date assignment rules (DDARs). One of the reasons for the lack of research in this direction is that dealing with each scheduling decision is already difficult; and thus considering multiple scheduling decisions simultaneously will be even more complicated. To tackle this problem, there is a need to develop new methodologies for improving the scheduling decisions and their interactions, which should also be able to cope with the dynamic features of JSS problems.

Genetic Programming (GP) [14], [15] is an evolutionary computation method which has been applied to evolve/train programs that are able to solve difficult computational problems [16], [17]. In the literature, GP has also been applied as a machine learning method to evolve dispatching rules for different machine scheduling environments [18], [19], [20], [21], [22], [23], [24]. In these studies, a dispatching rule is encoded by a GP tree and treated as a priority function to determine priorities for jobs waiting in the queues of machines. A set of instances were used to train the rules and the performance of the rules evolved by the proposed methods are evaluated by a set of test instances. The results from these studies showed that the evolved dispatching rules are very promising and outperform existing dispatching rules when tested on different scheduling instances. However, DDA was considered in these studies and the due-dates were generated randomly for the training/testing purpose only. In the work presented in this paper, GP is used as a hyper-heuristic method [25] for the automatic design of scheduling policies (SPs) which include sequencing/dispatching rules and due-date assignment rules for dynamic JSS problems. Unlike existing approaches, GP is suitable for designing SPs because of its flexibility to encode different scheduling rules in the representation. Moreover, as an evolutionary approach, GP can be applied to handle the multiple conflicting objectives of JSS problems. Another advantage of GP is that evolved scheduling policies are potentially interpretable, which is important and useful for understanding how the problem is solved by the evolved policies and how the trade-offs among the different objectives of JSS can be obtained.

This paper presents novel methodologies to design efficient SPs for solving dynamic multi-objective JSS problems via genetic programming based hyper-heuristic (GPHH) [26]. In order to address drawbacks of existing methods, three important aspects are considered in our proposed algorithms: (i) representations of different scheduling rules; (ii) evolutionary optimisation to evolve the trade-offs in SPs; and (iii) reusability (ability to be reused on new unseen problems [26], [25]) of the evolved SPs. The first aspect concentrates on how the scheduling rules can be represented and evaluated in GP. The second applies multi-objective evolutionary algorithms [27], [28], [29], [30], [31] to explore the Pareto front of the evolved SPs. In order to examine how training scenarios may influence reusability of the evolved rules on unseen situations, four multi-objective genetic programming based hyper-heuristic (MO-GPHH) methods are proposed to deal with the JSS problems.

In this paper, the following four research objectives will be presented: (1) developing MO-GPHH methods for automatic design of scheduling policies for dynamic job shop scheduling problems; (2) comparing the evolved scheduling policies with existing scheduling policies from combinations of existing dispatching rules and due-date assignment rules; (3) evaluating the reusability of the evolved scheduling policies; and (4) analysing the performance of the proposed MO-GPHH methods and the evolved scheduling policies.

Section II provides the relevant background of JSS problems and various GPHH methods. Section III describes the proposed MO-GPHH methods and the job shop simulation models used for training and testing. The experimental and comparison results of the evolved scheduling policies and the existing scheduling policies are presented in Section IV. The analysis of the proposed methods and the evolved scheduling policies are shown in Section V. Conclusions are drawn in Section VI.

II. Background

This section gives an overview of the JSS problem and GPHH with a special focus on scheduling problems.

A. Job Shop Scheduling

In JSS problems, a number of jobs are to be processed, each including one or more operations to be processed on a set of machines in the shop with a pre-defined order and processing times [32]. In practical situations, jobs can arrive at random over time and the processing times of these jobs are not known prior to their arrivals. There are many related decisions to make for jobs and machines in the shops such as due-date assignment, job order release and job scheduling. In this work, we focus on due-date assignment and job scheduling decisions; job release is simplified by immediately releasing jobs to the shop upon their arrival. An example of a job shop is shown in Fig. 1. In this figure, the due-date is assigned to a newly arriving job by some DDAR. Then, the job will be released to the shop and processed at the pre-determined machines. If the job is transferred to a machine when it is busy, the job will have to wait in the corresponding queue. Meanwhile, when a machine completes a job (or operation), the next job in the queue will be selected based on some sequencing/scheduling (dispatching) rule to be processed next.

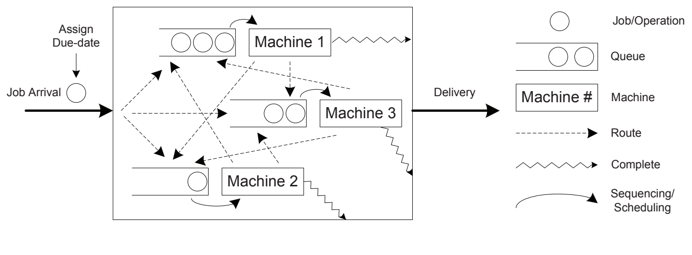


Fig. 1. Job Shop Scheduling (shop with 3 machines).

1) Due-date assignment: Due-date assignment decisions are made whenever jobs (customer orders) are received from customers [33]. Due-dates can be set exogenously or endogenously [32], [9]. In the former case, due-dates are decided by independent agencies (sellers, buyers). In this work, we focus on the latter case, in which the due-dates are internally set based on the characteristics of jobs and machines [32]. Basically, the due-date of a new job j is calculated as:

$$d_j = r_j + \hat{f}_j \tag{1}$$

The task of a DDAR is to assign a value to $\hat{f}_j$. In the ideal case, we want the estimated due-date d_j to be equal to the completion time of the job C_j. The miss due-date performance is normally measured by the error (lateness) between the completion time and due-date, i.e., $e_j = C_j - d_j = f_j - \hat{f}_j$.

Many DDARs have been proposed in the JSS literature. The early works of DDARs were mainly based on creating a simple model that employs aggregate information from the new job and the shop. Examples of these methods are Total Work Content (TWK) where $d_j = r_j + kp_j$, Number of Operations (NOP) where $d_j = r_j + km_j$, and Processing Plus Waiting (PPW) where $d_j = r_j + p_j + km_j$. In these methods, k is a coefficient that needs to be determined. Other more sophisticated models have also been proposed which incorporate more information about jobs and the shop to make better prediction of flowtimes. These include Job in Queue (JIQ), Job in System (JIS), Work in Queue (WIQ), Total Work and Number of Operations (TWK + NOP), Response Mapping Rule (RMR), and Operations-based Flowtime Estimation (OFE). The comparison results of these DDARs [10], [34], [35], [36], [37], [38] show that the DDARs which employ more useful information can lead to better performance. However, the drawback of these methods is that the results are depending on finding appropriate coefficients for factors used in the prediction models. Although linear regression models are a popular method to determine the coefficients, it restricts the DDARs to linear models only.

Because of the complexity and stochastic nature of dynamic job shops, nonlinear models are needed [35], which makes it computationally expensive to be solved by regression methods. Since the early 1990s, artificial intelligence methods have also been applied to deal with due-date assignment problems. These include neural networks [35], [39], decision trees [40], regression trees [41], and a regression-based method with case-based tuning [42], etc. Some dynamic DDARs without the need of finding optimal coefficients have also been proposed, such as Dynamic Total Work Content (DTWK), Dynamic Processing Plus Waiting (DPPW) [37], and ADRES [43].

2) Job sequencing: Over the last few decades, a large number of techniques have been applied to solve sequencing problems in job shops, ranging from simple heuristics to artificial intelligence and mathematical optimisation techniques [44]. The aim of these methods is to find an optimal schedule (e.g. minimising makespan) based on a given set of jobs. Since this is a NP-hard problem [45], mathematical programming approaches usually fail to produce optimal solutions within a reasonable computational time. There are many researches on applying meta-heuristics [46], [47] for solving scheduling problems in the last two decades, particularly for *static* JSS problems. The two popular directions in solving these problems are (1) local search based methods [2], [3], [5], [6], [8], [4], [48] and (2) evolutionary computation methods [49], [50], [51]. A comprehensive review of these methods is given in Ouelhadj and Petrovic [52] and Potts and Strusevich [53].

Although there are some breakthroughs in the developments of exact and approximate approaches, these methods mainly focus on static problems in simplified job shop environments. Other approaches like genetic algorithms have been extended to solve these problems with some realistic constraints, but they are generally computationally inefficient. Moreover, as pointed out by McKay et al. [54], conventional operations research and artificial intelligence approaches are often not applicable in handling the dynamic characteristics of actual manufacturing systems, since they are fundamentally based on static assumptions. For this reason, simple dispatching rules have been used in practice because of their ability to cope with any dynamics when the shop changes. For dispatching rules, each job in the queue of the machine is assigned a priority and the job with the highest priority will be processed first. There have been a large number of rules proposed in the literature which can be classified into three categories [44]: (1) simple priority rules, which are mainly based on information related to the jobs; (2) combinations of rules that are implemented depending on the situation that exists on the shop floor; and (3) weighted priority indices which employ more than one piece of information about each job to determine the schedule. Composite dispatching rules (CDR) [55], [56] can also be considered as a version of rules based on weighted priority indices, where scheduling information can be combined in more sophisticated ways instead of linear combinations. Panwalkar and Iskander [57] provided a comprehensive survey on scheduling (dispatching) rules used in research and real world applications using a similar classification. Pinedo [56] also showed various ways to classify dispatching rules based on the characteristics of these rules. The dispatching rules in this case can be classified as *static* or *dynamic* rules, where dynamic rules are time dependent (e.g. minimum slack) and static rules are not time dependent (e.g. shortest processing time). Another way to categorise these rules is based on the information used (either local or global information) in making sequencing decisions. A *local* rule only uses information available at the machine where the job is queued. A *global* rule, on the other hand, may use the information from other machines.

The interaction between DRs and DDARs has also been studied [10], [11], [12], [13]. It has been noted that logical combinations of DRs and DDARs can enhance the performance of scheduling systems. Since the design of an effective DR or DDAR is already time-consuming and complicated, developing a comprehensive scheduling policy with rational design of DRs and DDARs is often even more challenging.

B. GPHH for Scheduling Problems

GP has been applied in the field of hyper-heuristics, which is known as GPHH [26]. Because GP is able to represent and evolve complex programs or rules, it is an excellent choice for generating heuristics. The GPHH has also been applied to evolve dispatching rules for scheduling problems. Dimopoulos and Zalzala [58] used GP to evolve dispatching rules for the problem of one-machine scheduling with a standard function set and a terminal set of scheduling statistics (processing time, release time, due date, number of jobs, etc.). The evolved dispatching rules are better than traditional rules even for some large and unseen instances. Jakobovic et al. [19] employed the same method to develop dispatching rules for the problem of parallel machine scheduling in both static and dynamic environments. However, the evolved rules obtained from these studies do not consider the effect of different representations in regard to the performance of the GP system. Moreover, the machine and system attributes are not considered in the evolved rules. To learn new dispatching rules for a single machine environment, Geiger et al. [59] presented a learning system that combines GP with a simulation model of an industrial facility. The proposed GP method is used to create the priority rule for a single machine in both static and dynamic environments. The terminal set of GP includes system attributes, job attributes, and machine attributes, while the function set consists of basic operators such as $+$, $-$, $\times$, protected division $\%$ and `If`. The paper also proposed a method to learn dispatching rules for multiple machine problems in which GP will evolve multiple trees simultaneously with modified crossover and mutation operators. The evolved rules are competitive with the optimal Johnson's rule in a simple two-machine environment. Geiger and Uzsoy [21] also applied this system to learn dispatching rules for batch processor scheduling with good results. For a stochastic single machine scheduling problem, Yin et al. [60] proposed a GP system employing a bi-tree structured representation scheme to deal with machine breakdowns. In this system, each scheduling heuristic is represented by two subtrees, which are used to calculate the priorities and inserted idle times of jobs, respectively. The empirical results under different stochastic environments showed that GP can evolve high-quality predictive scheduling heuristics.

Miyashita [61] developed an automatic method using GP to design customised dispatching rules for a job shop environment. The JSS problem is considered as a multi-agent problem where each agent represents a resource (machine or work station). Three multi-agent models were proposed and explored by the GP: (1) a homogeneous model where all resources share the same dispatching rules, (2) a distinct agent model where each machine employs its own evolved rules, and (3) a mixed agent model where two rules can be selected to prioritise jobs depending on whether the machine is a bottleneck. Although the multi-agent models produce good results, the use of these models depends on some prior-knowledge of the job shop environment, which may change in dynamic situations. A similar system was proposed by Atlan et al. [62] to find the solution of a particular problem instance. Jakobovic and Budin [63] applied GP to evolve dispatching rules for both single machine and job shop environments. The results for the single machine environment are shown to outperform the existing rules. For the job shop environment, a meta-algorithm was defined to show how the evolved rules can be used to construct a schedule. They also proposed an interesting approach to provide some adaptive behaviours for the evolved rules by presenting a GP-3 system that evolves three components, a discriminant function and two dispatching rules. The discriminant function aims at identifying whether the machine to be scheduled is a bottleneck. This function serves as a classifier for the binary classification problems. Based on the classification decision obtained from the discriminant function, one of the two dispatching rules will be selected to sequence jobs in the queue of the machine. Although the purpose of the discriminant function in this case is to identify the bottleneck machine, there is no guarantee that the classification can help to indicate the bottleneck machine or just some useful attributes of the shop or machines. The results show that the GP-3 system performed better than traditional GP with a single tree. However, the paper gave no demonstration or analysis of the evolved rules.

Tay and Ho [20] proposed a GP system to evolve dispatching rules for a multi-objective job shop environment. The multi-objective problem was converted into a single objective problem by linearly combining all the objective functions. The proposed GP program can be considered as a priority function which is used to calculate the priority of operations in the queue of a machine based on a set of static and dynamic variables. The set of test instances was randomly generated and it has been shown that the evolved dispatching rules outperformed other simple dispatching rules. However, the use of machine attributes in the priority function was not considered. Hildebrandt et al. [18] re-examined the system in different dynamic job shop scenarios, which showed that the rules evolved by Tay and Ho [20] are only slightly better than the earliest release date rule and worse than the performance of the shortest processing time rule. They explained that the poor performance of these rules is caused by the use of a linear combination of different objectives. In addition, the randomly generated instances cannot effectively represent the situations that happened in a long term simulation. Hildebrandt et al. [18] then evolved dispatching rules via four simulation scenarios (10 machines with two utilisation levels and two job types) and minimised the mean flow time only. The experimental results showed that the evolved rules are rather complicated but more effective as compared to other existing rules. Moreover, these evolved rules are robust when they are tested in another environment (50 machines and different processing time distributions). However, their work has not considered multiple conflicting objectives and the influence of other scheduling decisions on the performance of the evolved dispatching rules.

Designing an effective scheduling system is an important and complicated task, which will influence the entire manufacturing system. Moreover, all the scheduling decisions and their interactions should be considered in order to ensure the success of the scheduling system in real world applications. This paper will present a novel GP approach for automatically designing scheduling policies including due-date assignment rules and dispatching rules for dynamic job shop environments. The focus of this work is on the representations of scheduling rules evolved by the GP system, the evolutionary search approaches to handle multiple conflicting objectives in JSS, and the reusability of the evolved scheduling policies.

III. Multi-objective GPHH methods for dynamic JSS problems

This section describes the new GPHH methods for evolving scheduling policies, which include rules for due-date assignment and sequencing decisions in dynamic job shop environments. We will first show how DDARs and DRs are represented by individuals in GP. Then, new MO-GPHH methods based on NSGA-II, SPEA2, HaD-MOEA and a proposed cooperative coevolution method are applied to deal with the dynamic JSS problems. These MO-GPHH methods are different in the way the Pareto fronts of non-dominated scheduling policies are explored. Lastly, the job shop simulation model used for the training and testing will be described.

A. Representations

1) Due-date Assignment Rules: The task of a DDAR is to determine the flowtime (i.e. due-dates by using equation (1)) by employing information from jobs and the shop. In this work, we evolve operation-based DDARs which will indirectly calculate job flowtimes through the accumulation of operation flowtimes in order to enhance the accuracy of the flowtime estimation by employing more detailed operation information. Here, we use a GP tree [14] to represent mathematical combinations of these pieces of information. The function set will include four standard mathematical operators $+$,$-$,$\times$, and protected division $\%$ (similar to normal division but returns a value of 1 when division by 0 is attempted), along with a conditional function `If` to allow GP to evolve sophisticated DDARs. Function `If` takes three arguments and if the value from the first argument is greater than or equal to zero, `If` will return the value from the second argument; otherwise `If` will return the value from the third argument. The terminal set for synthesising DDARs is shown in Table I. In this table, `SOTR` and `SAPR` are calculated based on the 20 previous jobs processed at machine δ. `SAR`, on the other hand, is calculated based on arrivals of the last 100 jobs. When the number of jobs is less than the sample size, `SOTR`, `SAPR` and `SAR` are calculated as an average over the available jobs.

TABLE I
Terminal sets for DDARs (σ is the considered operation, and δ is the machine that process σ)

`N`	number of jobs in the shop
`SAR`	sampled arrival rate
`APR`	average processing times of jobs in queue of the machine that processes σ
`OT`	processing time of σ
`LOT`	time for δ to finish the leftover job
`OTR`	percentage of jobs in queues of δ require less processing time less than `OT`
`SOTR`	percentage of sampled jobs processed at δ that require less processing time less than `OT`
`QWL`	total processing time of jobs in queue of δ
`SAPR`	sampled average processing time of jobs processed at δ
`RWL`	total processing time of jobs that need to be processed at δ
`W`	weight of the considered job
`PEF`	partial estimated flowtime
`#`	random number from 0 to 1

Instead of using the function obtained from the GP tree to directly estimate job flowtime, $\hat{f}$, the output of this function is used to estimate the operation flowtime $\hat{f}_o$ of each operation σ of the new job, starting from the first operation. When $\hat{f}_o$ is obtained, a condition is checked to see whether σ is the last operation. If it is not the last operation of the new job, $\hat{f}_o$ will be used to update the partial estimated flowtime (PEF), which will also be used as a terminal in the GP tree. Then, the next operation σ and the machine δ processing σ are used as the inputs for the GP tree to estimate the next operation flowtime $\hat{f}_o$. In the case that the flowtime of the last operation has been estimated, $\hat{f}_o$ will be added to the current PEF to obtain the estimated job flowtime $\hat{f}$. The evaluation scheme for a DDAR is shown in Fig. 2. The use of PEF (initially zero for the first operation) in the terminal set of DDARs also provides a chance to predict the changes of the system, given that the partial estimated flowtime is well predicted.

2) Dispatching Rules: The composite dispatching rules (CDR) [55], [56] will be represented by GP trees and will be evolved in this work. Basically, a CDR is a priority function that uses different pieces of information from jobs and machines to assign priorities to jobs in queues. The operation/job with the highest priority is scheduled to be processed next on the machine. The function set used for DDARs is also applied here along with `min`, `max` and `abs`, which commonly appear in existing CDRs. The terminal set for CDRs is shown in Table II. Terminals in the upper part provide information about the considered job. The last six terminals are used to provide information about the machine and shop status, where Λ is the set of operations/jobs currently in the shop that have the considered machine m on their routes; and K and I (subsets of Λ) are the sets of all operations/jobs that have and have not yet been processed by m, respectively ($\Lambda = K \cup I$). We call a machine *critical* if it has the greatest total remaining processing time $\sum_{\sigma \in I} p(\sigma)$ and a machine is called *bottleneck* if it has the largest workload $\sum_{\sigma \in \Omega} p(\sigma)$ where Ω is the set of jobs in queue of the considered machine. An example of a CDR is shown in Figure 3.

B. A Cooperative Coevolution MO-GPHH for DJSS

As discussed earlier, this work aims to evolve scheduling policies that include two key components, i.e., due-date assignment rules and dispatching rules. While the representation of the rules has been discussed previously, we need to specify how these rules are evolved in our proposed GPHH methods. In this work, two approaches are examined. First, a GP individual contains two GP trees for the two rules as presented above. In this case, each individual is equivalent to a scheduling policy. The scheduling policy is evaluated by applying the first tree as a DDAR when a new job arrives at the job shop to assign a due-date to that job. Meanwhile, the second tree is applied when a machine becomes idle and there are jobs in the queue of that machine to find the next job to be processed. For the first approach, we apply NSGA-II [29], SPEA2 [64], and HaD-MOEA [65] to explore the Pareto front of non-dominated scheduling policies similar to the common applications of these algorithms.

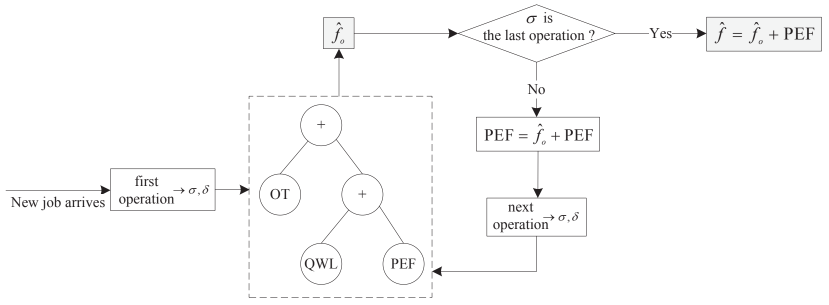

$\hat{f}_o$
σ is the last operation ?
Yes
$\hat{f} = \hat{f}_o + \text{PEF}$
No
$\text{PEF} = \hat{f}_o + \text{PEF}$
New job arrives
first operation $\rightarrow \sigma, \delta$
+
OT
+
QWL
PEF
next operation $\rightarrow \sigma, \delta$

Fig. 2. Operation-based DDAR.

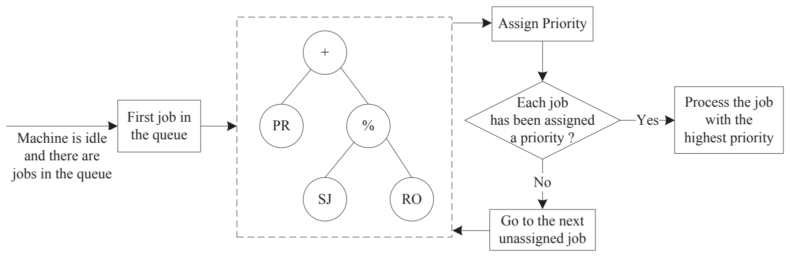

Assign Priority
+
PR
%
SJ
RO
Machine is idle and there are jobs in the queue
First job in the queue
Each job has been assigned a priority ?
Yes
Process the job with the highest priority
No
Go to the next unassigned job

Fig. 3. Example of GP tree as a CDR.

TABLE II
TERMINAL SET FOR CDR

`rJ`	job release time (arrival time)
`RJ`	operation ready time
`RO`	number of remaining operation of the job j.
`RT`	work remaining of the job
`PR`	operation processing time
`W`	weight of the job
`DD`	due date d_j
`RM`	machine ready time
`SJ`	slack of the job $j = \texttt{DD} - (\texttt{t} + \texttt{RT})$
`#`	Random number from 0 to 1
`WR`	workload ratio $= \frac{\sum_{\sigma \in \Omega} p(\sigma)}{\sum_{\sigma \in I} p(\sigma)}$
`MP`	machine progress $= \frac{\sum_{\sigma \in K} p(\sigma)}{\sum_{\sigma \in \Lambda} p(\sigma)}$
`DJ`	deviation of jobs in queue $= \frac{\min_{\sigma \in \Omega}\{p(\sigma)\}}{\max_{\sigma \in \Omega}\{p(\sigma)\}}$
`CWR`	critical workload ratio $= \frac{\sum_{\sigma \in \Omega^c} p(\sigma)}{\sum_{\sigma \in \Omega} p(\sigma)}$
`CWI`	critical machine idleness, `WR` of the critical machine
`BWR`	bottleneck workload ratio $= \frac{\sum_{\sigma \in \Omega^b} p(\sigma)}{\sum_{\sigma \in \Omega} p(\sigma)}$

*`t` is the time when the sequencing decision is made.
*Ω^b amd Ω^c are subsets of Ω that include jobs that will visit the bottleneck and critical machines in the future, respectively.

The second approach to evolving scheduling policies is to employ cooperative coevolution [66], [67], [68] to evolve two decision rules in two sub-populations. This approach is similar to the cooperative coevolution framework proposed by Potter and de Jong [69], in which the scheduling policy is the combination of an individual in a sub-population with a *representative* from the other sub-population, and some specialised operations are also employed here to help explore the Pareto front of the scheduling policies. In this work, we propose a new diversified multi-objective cooperative co-evolution (DMOCC) method based on the second approach. An overview of the proposed DMOCC is shown in Fig. 4. Here each sub-population (P_1 for DDARs and P_2 for DRs) represents one rule of the complete scheduling policy. For each individual $p_i^r \in P_r$, the objective values which determine the quality (fitness) of p_i^r are obtained by combining that individual with a representative from the other population to form a complete scheduling policy S. When a complete scheduling policy is applied to the job shop, the quality of that policy is characterised by the expected values of three performance measures: (1) makespan ($\texttt{C}_{\texttt{max}}$) [56]; (2) total weighted tardiness (`TWT`) [56]; and (3) mean absolute percentage error (`MPEA`) [43] (see Table V). $\texttt{C}_{\texttt{max}}$ and `TWT` are two popular performance measures for evaluating dispatching rules or scheduling methods while `MPEA` is used to indicate the accuracy of the due-date assignment rules. In this work, the scheduling policies are evolved such that these three performance measures are minimised.

Within DMOCC, we use the *crowding distance* (individuals with higher crowding distances are located in less crowded areas of the objective space) and *non-dominated rank* [29] (individuals with the same *rank* are not dominated by each other and dominated by at least one individual with a smaller *rank*) to select GP individuals for genetic operations and for collaboration between the two sub-populations. *Representatives* for collaboration are selected based on a *binary tournament selection* method [29], which randomly selects two individuals and the one with a lower non-dominated rank will be chosen. In the case that two individuals have the same rank, the individual with a higher crowding distance will be selected. The binary tournament selection is employed in DMOCC because it takes into account both the quality of the non-dominated individuals and their spread/distribution.

An external archive A is employed in this method to store the non-dominated scheduling policies. After all individuals have been evaluated, a set of non-dominated scheduling policies are extracted from individuals in the two sub-populations and the current archive to form a new archive. Besides storing the non-dominated scheduling policies, the archive in

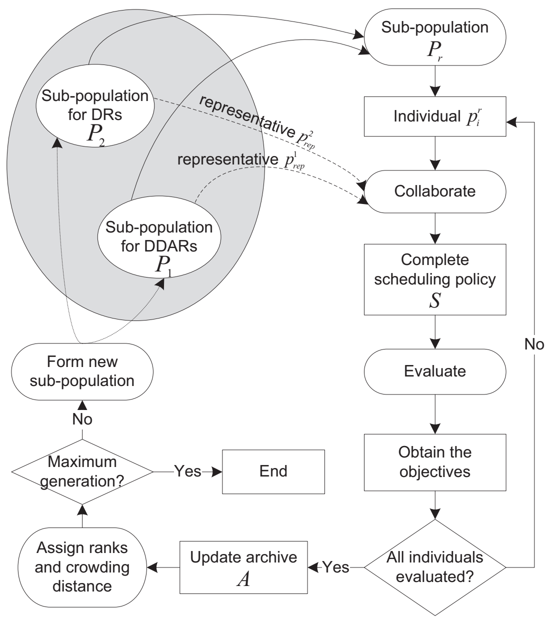

Sub-population P_r
Individual p_i^r
Collaborate
Complete scheduling policy S
Evaluate
Obtain the objectives
All individuals evaluated?
No
Yes
Update archive A
Assign ranks and crowding distance
Maximum generation?
Yes
End
No
Form new sub-population
Sub-population for DRs P_2
Sub-population for DDARs P_1
representative p_{rep}^2
representative p_{rep}^1

Fig. 4. Overview of DMOCC.

DMOCC is also used for two other purposes. First, it is used to evaluate the quality (rank and crowding distance) of the evolved scheduling policies in the two sub-populations. Instead of evaluating the quality of the evolved rules independently in each sub-population, it is better to assess their quality based on comparisons with those in the archive and the other sub-population to identify other potential non-dominated scheduling policies. Secondly, the archive can provide genetic materials which are needed for the crossover operation (more details are provided in Section III-C). Different from NSGA-II, SPEA2 and HaD-MOEA, the size of archive in DMOCC is not fixed, although the number of complete scheduling policies stored in the archive cannot exceed a predefined maximum size. When the number of non-dominated scheduling policies extracted from a generation is more than the maximum size, only individuals with the highest crowding distance will be preserved in the archive. Since new individuals will be created from parents in the archive through crossover, such a dynamic archive will help focus the search towards non-dominated scheduling policies at the early stage of the evolution. When the number of individuals in the archive increases, the shape of the Pareto front will be characterised and the method will focus on distributing the individuals uniformly.

The pseudo code of DMOCC is shown in Fig. 5. The algorithm starts by populating the two sub-populations P_1 and P_2 with randomly generated individuals. In each generation, each individual p_j^r of the two populations collaborates with the representative $p_{rep}^{r'}$ from the other sub-population to create a complete scheduling policy S. Then, the objective values of p_i^r are obtained by applying S to the simulated job shop. When all individuals have been evaluated, the archive A will be updated. Ranks and crowding distances are then assigned to individuals in A, P_1, and P_2. Here, new sub-populations are generated by genetic operations and the algorithm starts a new generation if the maximum generation is not reached.

```
initialise each sub-population $P_r$ with $r = \{1, 2\}$
  $P_r \leftarrow \{p_1^r, p_2^r, \dots, p_N^r\}$
$A \leftarrow \{\}$
**while** $maxGeneration$ is not reached **do**
    **for** $r = 1 \rightarrow 2$ **do**
        **for** $i = 1 \rightarrow N$ **do**
            $S \leftarrow collaborate(p_i^r, p_{rep}^{r'})$ where $r' \neq r$
            $p_i^r.objectives \leftarrow evaluate(S)$
        **end for**
    **end for**
    $A \leftarrow update(A \bigcup P_1 \bigcup P_2)$
    assign ranks and crowding distance
    **for** $r = 1 \rightarrow 2$ **do**
        $P_r \leftarrow genetic\_operations(P_r, A)$
    **end for**
**end while**
**return** $A$
```

Fig. 5. Pseudo code for DMOCC.

C. Genetic Operators

Traditional genetic operators are employed by the proposed MO-GPHH methods. For crossover, GP uses the subtree crossover [14], which creates new individuals for the next generation by randomly recombining subtrees from two selected parents. SPEA2 uses tournament selection to select parents in the population with the highest fitness value in the tournament. NSGA-II and HaD-MOEA use binary tournament selection based on rank and crowding distance as explained in the previous section. For DMOCC, binary tournament selection is used to select one parent from a sub-population and one parent from the archive. Since individuals in the archive have a rank of zero, the selection is made only based on the crowding distance in order to direct the search to less crowded areas. Here, mutation is performed by the subtree mutation [14], which randomly selects a node of a chosen individual in the population and replaces the subtree rooted at that node by a newly randomly-generated subtree. For NSGA-II, SPEA2 and HaD-MOEA, the genetic operations will first randomly choose which tree (either DR or DDAR) of the parents to perform the operations on since each individual includes two trees for the two scheduling rules. If the crossover is applied, only genetic materials from the selected tree of the same type will be exchanged (e.g. a tree representing DR in one parent will only crossover with a tree representing DR of the other parent).

D. Parameters

Table III shows the parameters used by the four proposed MO-GPHH methods. SPEA2 applied tournament selection with a tournament size of 5 to select individuals for the genetic operations. NSGA-II, SPEA2, and HaD-MOEA used a population size of 200 while DMOCC used a population size of 100 for each sub-population to ensure that the number of program evaluations remains the same for all methods. The archive size of SPEA2 and *maximum-size* of DMOCC are set to 200. These settings are used so that the proposed methods

TABLE III
PARAMETER SETTINGS

Initialisation	ramped-half-and-half [14]
Crossover rate	90%
Mutation rate	10%
Maximum depth	8
Number of generations	100
Population size	200 for NSGA-II, SPEA2, and HaD-MOEA, and 100 for each sub-population of DMOCC

TABLE IV
TRAINING AND TESTING SCENARIOS

Factor	Training	Testing
Number of machines	4,6	5,10,20
Utilisation	80%,90%	70%,80%,90%,95%
Distribution of processing time	Exponential	Exponential, Uniform
Distribution of # of operations	missing	missing,full

will give the same number of non-dominated scheduling policies at the end of each run.

E. Job Shop Simulation Model

In this work, we use a symmetrical (balanced) job shop simulation model in which each operation of a job has equal probability to be processed at any machine in the shop and a job visits each machine at most once. Therefore, machines in the shop are expected to have the same level of congestion in long simulation runs. This model has been used in many studies in the JSS literature [18], [37], [38], [70], [71]. Based on the factors discussed above, the scenarios for training and testing of the scheduling policies are shown in Table IV.

Without loss of generality, the mean processing time of operations is fixed to 1 in these scenarios. The arrival of jobs will follow a Poisson process [18], [37], [38] with the arrival rate adjusted based on the utilisation level. For the distribution of the number of operations, the *missing* setting is used to indicate that the number of operations will follow a discrete uniform distribution from 1 to the number of machines. Meanwhile, the *full* setting indicates that each job will have its number of operations equal to the number of machines in the shop. The sequence of machines that a job will visit is selected randomly with a job visiting a machine at most once. New jobs will be assigned random weights such that 20% of the jobs have weights of 1, 60% have weights of 2 and 20% have weights of 4. This setting was inspired by Pinedo and Singer [7], 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 replication of a simulation scenario, we start with an empty shop and the interval from the beginning of the simulation until the arrival of the 1000^{th} job is considered as the warm-up time and the statistics from the next completed 5000 jobs (set $\mathbb{C}$) are recorded to evaluate the performance measures of the scheduling policies as shown in Table V. In this table, $\mathtt{M}$ is the number of machines in the shop, $\bar{w}$ is the average weight and $\frac{1}{\mu}$ is the average processing time of an operation. The average values of these performance measures across different simulation scenarios/replications are the objectives to be minimised by the proposed MO-GPHH methods.

TABLE V
PERFORMANCE MEASURES OF SCHEDULING POLICIES

Makespan [56]	$\mathtt{C_{max}} = \max_{j \in \mathbb{C}}\{f_j\}$
Normalised Total Weighted Tardiness [72]	$\mathtt{TWT} = \frac{\sum_{j \in \mathbb{C}} w_j T_j}{\lvert\mathbb{C}\rvert \times \mathtt{M} \times \frac{1}{\mu} \times \bar{w}}$
Mean Absolute Percentage Error [43]	$\mathtt{MAPE} = \frac{1}{\lvert\mathbb{C}\rvert} \sum_{j \in \mathbb{C}} \frac{\lvert e_j \rvert}{f_j}$

In the training stage, due to the heavy computation time, we only perform one replication for each scenario. As mentioned in Table IV, there are $(2 \times 2 \times 1 \times 1) = 4$ simulation scenarios used for evaluating the performance of the evolved scheduling policies. It should be noted that the performance measures are obtained for each scenario by applying the evolved scheduling policies thousands of times, since there are thousands of due-date assignment and sequencing decisions needed to be made during a simulation replication of that scenario. During the testing, each of the non-dominated scheduling policies from a GP run is applied to $(3 \times 4 \times 2 \times 2) = 48$ simulation scenarios (see Table IV) and 5 simulation replications are performed for each scenario; therefore, we perform $48 \times 5 = 240$ simulation replications for testing the performance of the obtained non-dominated scheduling policies. The use of a large number of scenarios and replications in the testing stage will help confirm the quality and reusability of the evolved scheduling policies.

F. Performance Measures for MO-GPHH Methods

Similar to other multi-objective optimisation applications, we are interested in the quality of the obtained Pareto fronts in terms of (1) *convergence* to the trade-off solutions and (2) the *spread* or *distribution* of the solutions on the obtained Pareto front. Three popular performance metrics for multi-objective optimisation are used here: hypervolume ratio (HVR) [28], [73]; SPREAD [29]; and generational distance (GD) [27].

1) Hypervolume (HV) and Hypervolume Ratio (HVR): Hypervolume is used to measure the "volume" in the objective space covered by the obtained non-dominated solutions for minimisation problems,

$$HV = volume(\bigcup_{i=1}^{n_{PF}} \nu_i) \tag{2}$$

where n_{PF} is the number of members in the obtained Pareto front PF_{known}, ν_i is the hypercube constructed with a reference point and the member i as the diagonal of the hypercube [28]. van Veldhuizen and Lamont [73] normalised HV by using the hypervolume ratio which is the ratio of the hypervolume of PF_{known} and the hypervolume of the reference Pareto front PF_{ref},

$$HVR = \frac{HV(PF_{known})}{HV(PF_{ref})} \tag{3}$$

2) SPREAD: This metric measures the non-uniformity of PF_{known} [29]. A widely and uniformly spread out set of non-dominated solutions in the PF_{known} will result in a small

$SPREAD$.

$$SPREAD = \frac{d_f + d_l + \sum_{i=1}^{n_{PF}-1} |d'_i - \bar{d}|}{d_f + d_l + (n_{PF} - 1)\bar{d}} \quad (4)$$

where d'_i is the Euclidean distance between member i and its nearest member in PF_{known}, $\bar{d}$ is the average of all d'_i, and d_f and d_l are the Euclidean distances between the extreme solutions and the boundary solutions of PF_{known}.

3) Generational Distance (GD): This metric is used to measure the distance between the obtained Pareto front (PF_{known}) and the reference Pareto front (PF_{ref}) [27],

$$GD = \left(\frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} d_i^2 \right)^{1/2} \quad (5)$$

where d_i is the Euclidean distance between the member i in PF_{known} and its nearest member in PF_{ref}.

PF_{ref} is normally the true Pareto front, which is unknown in the simulation here. Therefore, we adopt a reference Pareto front PF_{ref} in the calculation of these performance metrics. In this work, PF_{ref} includes the non-dominated scheduling policies extracted from all scheduling policies found by the four MO-GPHH methods (NSGA-II, SPEA2, HaD-MOEA, and DMOCC) in all independent runs as shown in the next section.

IV. Results

In order to evaluate the effectiveness of the proposed methods, 30 independent runs of each MO-GPHH method are performed and the evolved Pareto fronts obtained from each run are recorded. The evolved non-dominated scheduling policies (SPs) are then compared with the existing SPs based on combinations of well-known dispatching rules with dynamic and regression-based due-date assignment rules.

A. Pareto Front of the Evolved Scheduling Policies

Fig. 6(a) and Fig. 6(b) show the *aggregate Pareto fronts* extracted from all the evolved scheduling policies, which were obtained by the four proposed MO-GPHH methods in 30 independent runs for both the training and testing scenarios. It can be observed that the three objectives $\texttt{C}_{\texttt{max}}$, TWT and MAPE are conflicting objectives. When tracing along the Pareto front to find scheduling policies that are able to minimise $\texttt{C}_{\texttt{max}}$ and TWT, it can be seen that the value of MAPE tends to be increased. This suggests that scheduling policies that provide better shop performance (small $\texttt{C}_{\texttt{max}}$ and TWT) will result in flowtimes that are hard to predict accurately (large MAPE). Given a similar value of MAPE, trade-off between $\texttt{C}_{\texttt{max}}$ and TWT can also be observed, which suggests that there is no evolved dispatching rule that can simultaneously optimise these objectives. Such an observation is consistent with those discussed in the literature [55], [20].

In both the testing and training scenarios, it is observed that $\texttt{C}_{\texttt{max}}$ and TWT can be significantly reduced by using SPs with MAPE smaller than 0.5. The use of more sophisticated SPs can provide slightly better $\texttt{C}_{\texttt{max}}$ and TWT but they also make the job flowtimes much more difficult to be estimated. These results show that there are many trade-offs to be considered when selecting an appropriate SP for a scheduling system and the knowledge about these trade-offs is useful in making a better decision. For example, the obtained Pareto fronts suggest that much better delivery reliability (a small MAPE) can be achieved with a reasonable sacrifice in $\texttt{C}_{\texttt{max}}$ or TWT. However, if a single objective such as $\texttt{C}_{\texttt{max}}$ or TWT is to be minimised in this case, the evolved SPs will lead to very poor delivery reliability (a high MAPE) and thus reduce the customer satisfaction. Also, given the shape of the Pareto fronts in Fig. 6, it would be difficult to apply a traditional linear combination of objective values for fitness assessment [20] to find desirable rules due to the difficulty in identifying suitable weights for each objective. These observations show that handling multiple objectives with knowledge about their Pareto front is crucial for the design of effective scheduling policies.

B. Comparison to Existing DRs and Dynamic DDARs

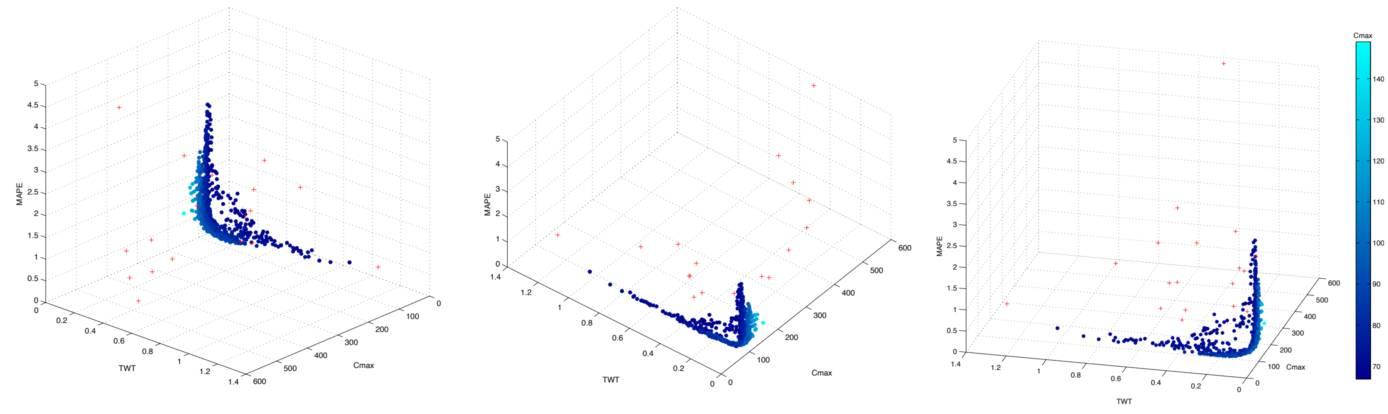
The combination of six popular DRs and three dynamic DDARs are evaluated on both the training and testing scenarios. The results are compared with the evolved non-dominated SPs as shown in Fig. 6. The six DRs used in this comparison are First-In-First-Out (FIFO), Critical Ratio (CR), Slack-per-Operation (S/OPN), Shortest Processing Time (SPT) [57], weighted Apparent Tardiness Cost (ATC) and weighted Cost Over Time (COVERT) [72]. The parameters of ATC and COVERT are the same as those used in Vepsalainen and Morton [72] ($k = 3$ for ATC, $k = 2$ for COVER, and the leadtime estimation parameter $b = 2$). The three dynamic DDARs are DTWK, DPPW [37] and ADRES [43]. These DDARs are selected for comparison because they are well-known in the scheduling literature and the application of these rules does not require predetermination of any parameter or coefficient for each simulation scenario. The objective values obtained by these 18 combinations for the training and testing scenarios are shown in Table VI and Table VII and are visualised as crosses in Fig. 6(a) and (b).

TABLE VI
PERFORMANCE OF EXISTING SCHEDULING POLICIES FOR TRAINING SCENARIOS ($\texttt{C}_{\texttt{max}}$, TWT, MAPE)

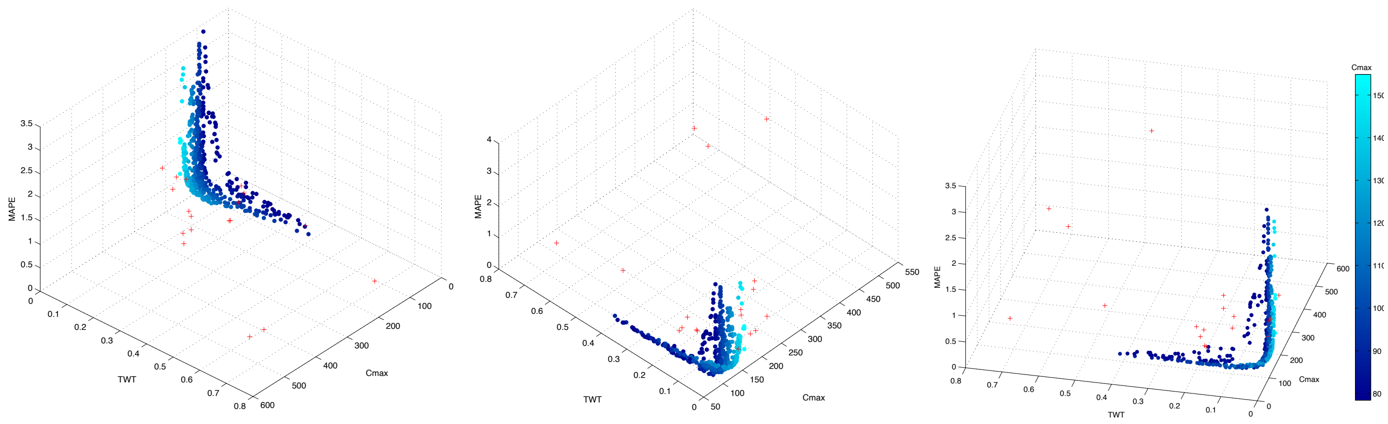
	DTWK	DPPW	ADRES
FIFO	(101.5, 1.25, 0.81)	(101.5, 0.71, 2.00)	(101.5, 0.36, 1.05)
CR	(174.6, 0.53, 0.73)	(127.4, 0.52, 2.47)	(178.0, 0.49, 1.33)
S/OPN	(156.9, 0.42, 0.57)	(114.0, 0.42, 1.63)	(123.9, 0.14, 1.68)
SPT	(575.8, 0.60, 0.67)	(575.8, 0.69, 1.45)	(575.8, 0.46, 4.96)
ATC	(476.9, 0.32, 0.58	(504.3, 0.36, 1.32)	(173.8, 0.06, 2.17)
COVERT	(301.6, 0.25, 0.40)	(362.9, 0.23, 1.00)	(145.4, 0.09, 0.96)

TABLE VII
PERFORMANCE OF EXISTING SCHEDULING POLICIES FOR TESTING SCENARIOS ($\texttt{C}_{\texttt{max}}$, TWT, MAPE)

	DTWK	DPPW	ADRES
FIFO	(149.7, 0.73, 0.35)	(149.7, 0.47, 0.80)	(149.7, 0.21, 0.60)
CR	(160.5, 0.19, 0.18)	(115.2, 0.18, 0.71)	(206.1, 0.03, 0.60)
S/OPN	(159.9, 0.20, 0.20)	(114.2, 0.19, 0.58)	(184.6, 0.02, 0.74)
SPT	(510.2, 0.68, 0.56)	(510.2, 0.74, 0.86)	(510.2, 0.45, 2.59)
ATC	(302.8, 0.19, 0.29)	(306.4, 0.19, 0.53)	(220.4, 0.01, 1.05)
COVERT	(242.8, 0.15, 0.21)	(237.9, 0.14, 0.46)	(152.2, 0.02, 0.55)



(a) Training Scenarios



(b) Testing Scenarios

Fig. 6. Pareto front of non-dominated scheduling policies. The plots show three different views of the aggreate Pareto fronts obtained for training and testing, ● and + respectively represent evolved and existing scheduling policies as shown in Section IV-C.

Among these existing scheduling policies, the ones given with FIFO provide the best $\mathtt{C_{max}}$. The scheduling policies with DTWK provide the best MAPE, and the combination of ATC and ADRES achieves the best TWT. However, these existing scheduling policies are easily dominated by the evolved scheduling policies in the *aggregate Pareto fronts* as shown in Fig. 6. Moreover, when compared with the non-dominated scheduling policies obtained by *each independent run* of NSGA-II, SPEA2, HaD-MOEA and DMOCC, it can be observed from our experiments that these scheduling policies are dominated by at least one of the evolved scheduling policies using the proposed methods in both the training and testing scenarios. These results show that the non-dominated scheduling policies evolved by the proposed MO-GPHH methods not only show good performance on the training scenarios, but can also be effectively reused for unseen scenarios.

C. Comparison to Existing DRs and Regression-based DDARs

We further examine the effectiveness of the evolved SPs by comparing them with existing DRs and regression-based DDARs. The four due-date assignment models used here are TWK, NOP, JIQ and JIS in combination with the six dispatching rules reported in the previous section. Different from the dynamic DDARs, the coefficients of the employed models have to be determined by regression methods for each job shop setting. Fig. 7 and Table VIII show the performance of these $(6 \times 4) = 24$ combinations and the *aggregate Pareto front* of the non-dominated scheduling policies for the case with utilisation of 90%, 5 machines, *full* setting and processing times follow an exponential distribution. In this case, the coefficients of the due-date assignment models TWK, NOP, JIQ and JIS were determined by using Iterative Multiple Regression (IMR) [74]. The values shown in the figure are the average values of the three objectives obtained from 30 independent simulation replications.

Since this work deals with a dynamic JSS environment with stochastic factors (such as arrival process, processing time), we also examine the Pareto dominance of SPs under uncertainty. The definition of *statistical Pareto dominance* is introduced in Appendix A to help determining the Pareto dominance relation between two scheduling policies in this case. The results show that each of the 24 existing SPs considered here is statistically dominated (with a significant level $\alpha = 0.05$ and the Bonferroni method [75] used to adjust the value of

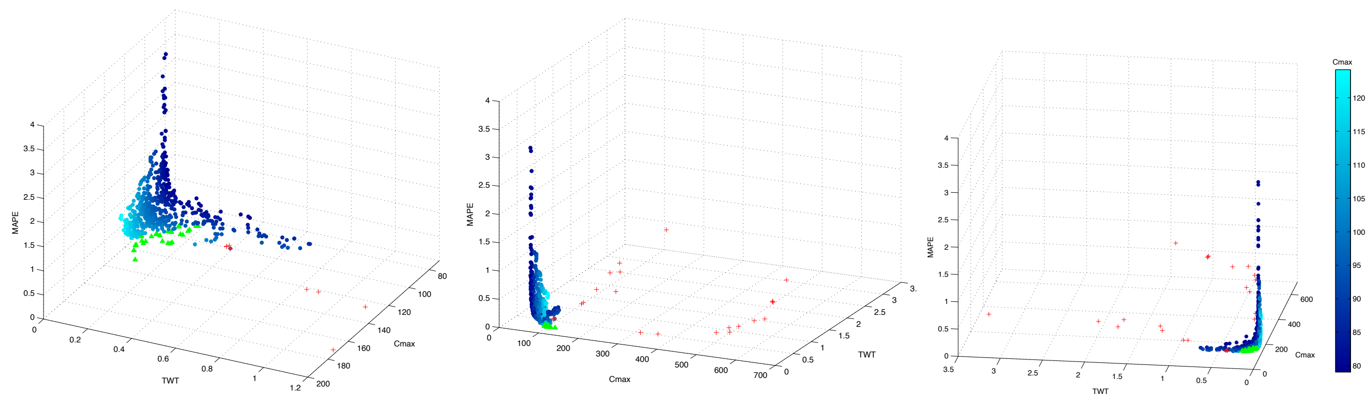


Fig. 7. DRs and Regression-based DDARs vs. evolved scheduling policies. (● and ▲ are respectively the non-dominated and dominating evolved scheduling policies; and + represents existing scheduling policies)

TABLE VIII
PERFORMANCE OF DRS AND REGRESSION-BASED DDARS

	C_{max}	TWT	MAPE
FIFO + TWK	129.381 $\pm$ 27.26	3.231 $\pm$ 1.38	0.495 $\pm$ 0.05
FIFO + NOP	129.381 $\pm$ 27.26	1.933 $\pm$ 1.31	0.451 $\pm$ 0.09
FIFO + JIQ	129.381 $\pm$ 27.26	0.924 $\pm$ 0.09	0.178 $\pm$ 0.01
FIFO + JIS	129.381 $\pm$ 27.26	0.871 $\pm$ 0.13	0.178 $\pm$ 0.01
CR + TWK	183.169 $\pm$ 44.14	1.239 $\pm$ 1.06	0.291 $\pm$ 0.07
CR + NOP	139.275 $\pm$ 40.01	1.183 $\pm$ 0.94	0.310 $\pm$ 0.09
CR + JIQ	103.784 $\pm$ 15.15	0.406 $\pm$ 0.04	0.084 $\pm$ 0.01
CR + JIS	102.935 $\pm$ 16.73	0.384 $\pm$ 0.04	0.085 $\pm$ 0.01
S/OPN + TWK	156.279 $\pm$ 27.73	1.650 $\pm$ 1.36	0.440 $\pm$ 0.10
S/OPN + NOP	126.500 $\pm$ 29.73	1.698 $\pm$ 1.44	0.375 $\pm$ 0.10
S/OPN + JIQ	102.718 $\pm$ 15.44	0.383 $\pm$ 0.05	0.088 $\pm$ 0.08
S/OPN + JIS	101.404 $\pm$ 15.74	0.390 $\pm$ 0.07	0.089 $\pm$ 0.09
SPT + TWK	603.661 $\pm$ 235.69	0.996 $\pm$ 0.30	0.608 $\pm$ 0.03
SPT + NOP	603.661 $\pm$ 235.69	1.370 $\pm$ 0.33	0.844 $\pm$ 0.03
SPT + JIQ	603.661 $\pm$ 235.69	0.979 $\pm$ 0.29	0.632 $\pm$ 0.02
SPT + JIS	603.661 $\pm$ 235.69	0.990 $\pm$ 0.30	0.616 $\pm$ 0.02
ATC + TWK	600.539 $\pm$ 213.47	0.507 $\pm$ 0.19	0.481 $\pm$ 0.04
ATC + NOP	609.517 $\pm$ 200.21	0.702 $\pm$ 0.21	0.441 $\pm$ 0.03
ATC + JIQ	570.168 $\pm$ 201.18	0.409 $\pm$ 0.10	0.395 $\pm$ 0.02
ATC + JIS	548.237 $\pm$ 190.89	0.381 $\pm$ 0.08	0.362 $\pm$ 0.02
COVERT + TWK	511.948 $\pm$ 204.55	0.424 $\pm$ 0.18	0.225 $\pm$ 0.03
COVERT + NOP	540.604 $\pm$ 210.92	0.483 $\pm$ 0.19	0.233 $\pm$ 0.04
COVERT + JIQ	379.399 $\pm$ 141.83	0.258 $\pm$ 0.03	0.141 $\pm$ 0.02
COVERT + JIS	336.875 $\pm$ 116.26	0.237 $\pm$ 0.02	0.135 $\pm$ 0.02

each individual statistical test) by at least one evolved SP in the aggregate Pareto front, which is indicated as a *dominating evolved scheduling policy* in Fig. 7 and Table IX. This further shows the high-quality of the evolved SPs even when they are compared with customised SPs. Fig. 7 also reveals that the combinations of existing DRs and DDARs do not cover all promising regions in the objective space. This observation suggests that automatic design methods like the proposed MO-GPHH methods are essential in order to provide informed knowledge about any potential SPs. Moreover, these results suggest that the evolved SPs are robust to uncertain JSS environments even though they are trained/evolved based on the mean values of the objectives across different simulation scenarios.

V. FURTHER ANALYSIS

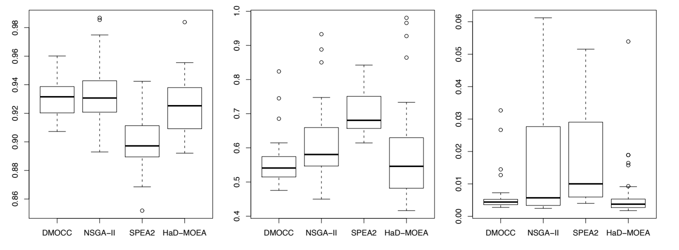
The comparison results in the previous section have shown the effectiveness of the proposed MO-GPHH methods for evolving efficient SPs. In this section, we will compare the ability of the proposed MO-GPHH methods in exploring the Pareto front of non-dominated SPs.

TABLE IX
PERFORMANCE OF DOMINATING EVOLVED SCHEDULING POLICIES

SP/Objective	C_{max}	TWT	MAPE
#1	91.507 $\pm$ 15.32	0.204 $\pm$ 0.04	0.114 $\pm$ 0.01
#2	91.824 $\pm$ 15.47	0.194 $\pm$ 0.04	0.114 $\pm$ 0.01
#3	95.497 $\pm$ 16.46	0.135 $\pm$ 0.02	0.116 $\pm$ 0.01
#4	95.590 $\pm$ 16.61	0.182 $\pm$ 0.03	0.093 $\pm$ 0.01
#5	96.075 $\pm$ 16.85	0.134 $\pm$ 0.02	0.117 $\pm$ 0.01
#6	96.996 $\pm$ 16.67	0.182 $\pm$ 0.03	0.093 $\pm$ 0.01
#7	101.597 $\pm$ 15.45	0.180 $\pm$ 0.02	0.097 $\pm$ 0.01
#8	102.104 $\pm$ 17.60	0.171 $\pm$ 0.02	0.111 $\pm$ 0.01
#9	102.430 $\pm$ 19.77	0.133 $\pm$ 0.01	0.127 $\pm$ 0.01
#10	102.562 $\pm$ 18.31	0.132 $\pm$ 0.01	0.122 $\pm$ 0.01
#11	102.567 $\pm$ 19.25	0.122 $\pm$ 0.01	0.125 $\pm$ 0.01
#12	102.885 $\pm$ 18.65	0.118 $\pm$ 0.01	0.135 $\pm$ 0.02
#13	103.970 $\pm$ 18.12	0.106 $\pm$ 0.02	0.138 $\pm$ 0.02
#14	106.151 $\pm$ 17.82	0.096 $\pm$ 0.01	0.084 $\pm$ 0.01
#15	106.892 $\pm$ 17.40	0.085 $\pm$ 0.01	0.136 $\pm$ 0.02
#16	107.526 $\pm$ 21.07	0.086 $\pm$ 0.01	0.122 $\pm$ 0.01
#17	109.083 $\pm$ 18.78	0.165 $\pm$ 0.01	0.066 $\pm$ 0.01
#18	110.068 $\pm$ 22.30	0.126 $\pm$ 0.01	0.071 $\pm$ 0.01
#19	110.518 $\pm$ 19.41	0.163 $\pm$ 0.01	0.069 $\pm$ 0.01
#20	110.667 $\pm$ 19.84	0.154 $\pm$ 0.01	0.068 $\pm$ 0.01
#21	110.754 $\pm$ 19.52	0.080 $\pm$ 0.01	0.136 $\pm$ 0.02
#22	111.922 $\pm$ 19.59	0.040 $\pm$ 0.01	0.134 $\pm$ 0.01
#23	112.373 $\pm$ 17.29	0.040 $\pm$ 0.01	0.122 $\pm$ 0.01
#24	114.717 $\pm$ 23.73	0.072 $\pm$ 0.01	0.097 $\pm$ 0.01
#25	115.209 $\pm$ 24.68	0.057 $\pm$ 0.01	0.101 $\pm$ 0.01
#26	116.613 $\pm$ 22.75	0.069 $\pm$ 0.01	0.091 $\pm$ 0.01
#27	120.045 $\pm$ 27.26	0.053 $\pm$ 0.01	0.107 $\pm$ 0.01
#28	120.625 $\pm$ 25.72	0.052 $\pm$ 0.01	0.107 $\pm$ 0.01
#29	124.931 $\pm$ 24.49	0.065 $\pm$ 0.01	0.095 $\pm$ 0.01
#30	125.122 $\pm$ 26.18	0.065 $\pm$ 0.01	0.096 $\pm$ 0.01
#31	132.354 $\pm$ 26.58	0.105 $\pm$ 0.01	0.080 $\pm$ 0.01

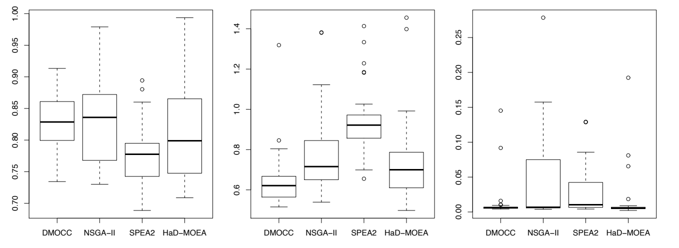
A. *Performance of MO-GPHH Methods*

The performance indicators of the four MO-GPHH methods are shown in Fig. 8 and Fig. 9 (better methods have higher HVR and smaller SPREAD and GD). With the training scenarios, *Wilcoxon signed-rank tests* (with significance level of 0.05) show that the HVRs obtained by DMOCC, NSGA-II, and HaD-MOEA are significantly better (higher) than that of SPEA2. This means that the SPs obtained by these methods can significantly dominate those obtained by SPEA2. In terms of HVR, there is no significant difference between DMOCC, NSGA-II, and HaD-MOEA but the standard deviations of



(a) HVR (b) SPREAD (c) GD

Fig. 8. Performance of MO-GPHH methods on training scenarios. (HVR to be maximised; and SPREAD and GD to be minimised)



(a) HVR (b) SPREAD (c) GD

Fig. 9. Performance of MO-GPHH methods on testing scenarios.

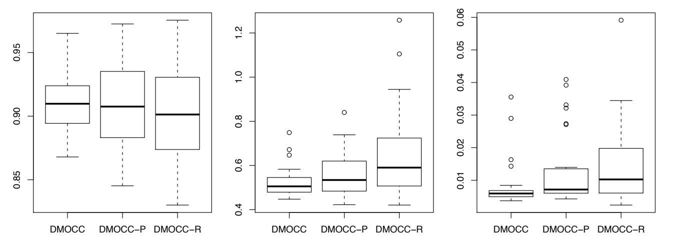
HVRs obtained by DMOCC and HaD-MOEA are slightly smaller than those obtained by NSGA-II. For the distribution of the obtained SPs on the Pareto fronts, the SPREAD values obtained by DMOCC and HaD-MOEA are significantly better than those obtained by NSGA-II and SPEA2. Although DMOCC uses *crowding distance* (like NSGA-II) as the indicator for individuals in less crowded areas, the selection method for choosing representative individuals as well as individuals for crossover has significantly improved the uniformity of the Pareto fronts obtained by DMOCC. Given a better distribution of scheduling policies, GD of DMOCC is significantly smaller than NSGA-II although there is no significant difference in HVR. Overall, DMOCC and HaD-MOEA are the two most competitive methods for the problems studied in this paper. It should be noted that performances of the obtained non-dominated SPs on the testing scenarios are rather consistent with those obtained in the training scenarios. However, the SPREAD of DMOCC is significantly better than all the other methods. These experimental results show that the proposed DMOCC is a very promising approach for evolving highly efficient SPs.

B. Complexity of DMOCC

The complexity of DMOCC depends on the operations performed at each generation. Similar to NSGA-II, the three basic operations of DMOCC are (1) non-dominated sorting, (2) crowding-distance assignment, and (3) sorting for genetic and representative selection. For non-dominated sorting, we adopt the procedure proposed by Deb et al. [29], which results in the worst-case complexity of $O(MR^2)$ where M is the number of objective functions to be minimised and R is the size of the joined population. Assuming that the size of each sub-population N is the same and the maximum size of the archive is A, the size of the joined population is $R = 2N + A$. The worst-case complexity of the crowding-distance assignment and sorting for genetic operators and representative selection are $O(MR\log(R))$ and $O(R\log(R))$, respectively. It is obvious that the complexity of the algorithm is $O(MR^2)$, governed by the non-dominated sorting procedure. Therefore, both the sub-population size N and archive size A will influence the complexity of DMOCC. For complex problems where GP needs a large population size in order to maintain the diversity of the population, the complexity of NSGA-II ($O(MN^2)$) will increase since its complexity depends mainly on the population size. Because the number of final non-dominated solutions is not necessarily as large as the population, the complexity of DMOCC can be smaller than that of NSGA-II by maintaining a small archive.

C. Representative Selection

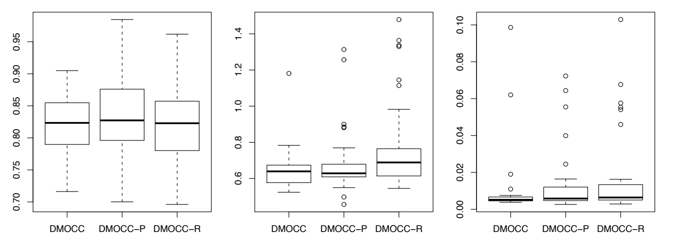
As mentioned earlier, representative selection is an important factor in the proposed cooperative coevolution method. Here we will examine the influence of representative selection methods on the performance of the proposed DMOCC. Apart from the representative selection method discussed above, two other methods are also examined here. The first is a problem-based method which applies two different representative selection strategies for each sub-population. In this method, the representatives of the sub-population of DRs are selected by using a similar method as in DMOCC (based on the non-dominated rank and crowding distance). On the other hand, the representatives of DDARs are selected based on the values of `MAPE`. This method assumes that good DDARs (with small values of `MAPE`) are able to cope with a wide range of DRs, and thus it will only focus on `MAPE` when selecting representatives to form complete SPs with the evolved DRs. The second method simply selects random representatives from each sub-population. The performances of the three representative selection methods are shown in Fig. 10 and Fig. 11. The DMOCC-P and DMOCC-R are similar to DMOCC, except that they employ problem-based and random representative selection methods, respectively.



(a) HVR (b) SPREAD (c) GD

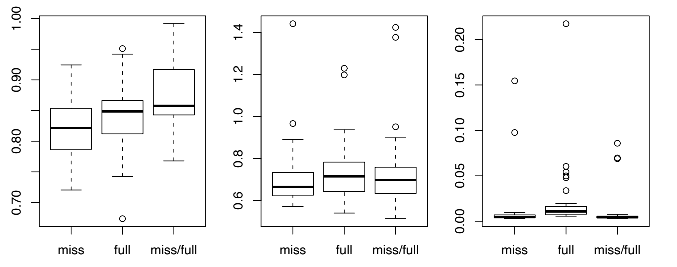
Fig. 10. Influence of representative selection methods on training scenarios.

The results from these figures show that the HVR values of the three selection methods are not significantly different. However, DMOCC gives significantly better SPREAD and GD performances as compared to DMOCC-P and DMOCC-R. Also, the DMOCC-P is better than DMOCC-R according to these two performance metrics. The results show that it is important to include the representative selection method based on the non-dominated rank and crowding distance. Al-



(a) HVR (b) SPREAD (c) GD

Fig. 11. Influence of representative selection methods on testing scenarios.



(a) HVR (b) SPREAD (c) GD

Fig. 12. Influence of training scenarios on testing performance of DMOCC.

though individuals selected for genetic operations for each subpopulation also employ the non-dominated rank and crowding distance, the features of non-dominated rank and crowding distance still have a strong impact on the performance of the representative selection method. Also, the evolved DDARs with good MAPE are not necessarily suitable for a wide range of DRs, since DMOCC-P does not produce Pareto fronts with the performance of SPREAD as good as DMOCC.

D. Choice of Training Scenarios

Like other machine learning methods, it would be interesting to examine how the choices of training sets or training scenarios may influence the ability of the proposed MO-GPHH in exploring effective scheduling policies. Previously, we have trained the proposed MO-GPHH on scenarios with the *missing* setting of arriving jobs. This section will further examine the cases where *full* and *missing/full* settings are used. The first case used 4 simulation scenarios and the second case used 8 simulation scenarios for training. Fig. 12 shows the performance of DMOCC on the testing scenarios when different training scenarios have been used.

The results show that there is no significant difference between the cases where *missing* and *full* settings are used. When both *missing* and *full* settings are used for training, the obtained HVRs are significantly better than those obtained in the cases where either *missing* or *full* setting is used and there is no significant differences in SPREAD and GDs. This indicates that more general training scenarios are necessary in order to improve the quality of the evolved scheduling policies. Although the simulation scenarios with jobs following the *missing* setting also include jobs following the *full* setting, it is still unable cover all situations that happened in the simulation scenarios with jobs following the *full* setting. The major problem is that the use of a large number of simulation scenarios will increase the computation cost of the proposed methods. Thus, there is a trade-off between the computational effort and the reusability of the evolved scheduling policies. Depending on the available computational resources and the environments where the evolved scheduling policies will be applied, the training simulation scenarios should be logically selected.

E. The Evolved Scheduling Policies

This section investigates how the evolved scheduling policies can effectively solve the problem and how trade-offs can be made among different objectives. Since many SPs have been obtained from our experiments, we will present some examples of the SPs. Fig. 13 is the same as Fig. 6(a) with a different view and the points surrounded by rectangles are the example SPs, which are also presented in Table X.

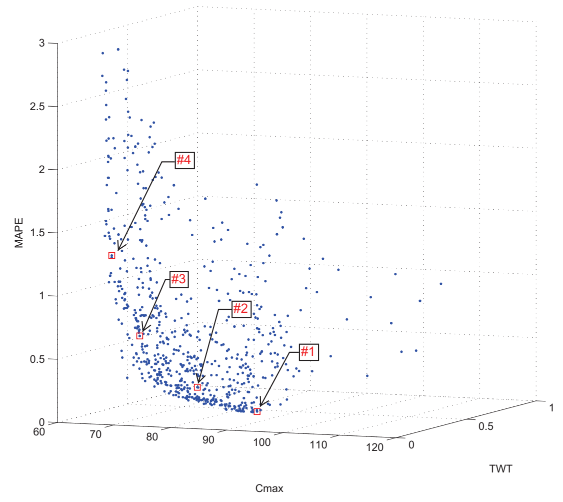


Fig. 13. Pareto front and selected evolved scheduling policies.

TABLE X
EXAMPLES OF THE EVOLVED SCHEDULING POLICIES

Scheduling Policy #1 ($C_{max} = 90.774, TWT = 0.170, MAPE = 0.098$)	
DR :	$\frac{2RO^2MP}{(0.9087407-RJ)} + IF(\max(BWR, WR) - 2SJ, -PR, -RJ)$
DDAR:	$OT + LOT + QWL$
Scheduling Policy #2 ($C_{max} = 83.597, TWT = 0.048, MAPE = 0.312$)	
DR :	$\max(-0.043989424, \frac{1}{PR}(\frac{2PR+RT}{PR} + \max(RT, PR) - SJ))$
DDAR:	$OT + LOT + QWL$
Scheduling Policy #3 ($C_{max} = 66.844, TWT = 0.311, MAPE = 0.608$)	
DR :	$RT\frac{RO}{PR} + IF(-CWR\frac{SJ}{PR}, DJ, RT(\frac{RO}{PR})^2) - 2rJ + W$
DDAR:	$OT + 2LOT + QWL$
Scheduling Policy #4 ($C_{max} = 68.162, TWT = 0.059, MAPE = 1.321$)	
DR :	$RT(2\frac{RO}{PR} - CWR) - 3rJ + W$
DDAR:	$OT + 2LOT + 2QWL + 2SOTR$

*These rules have been simplified for better presentation but still ensure to achieve the same objective values obtained by the original evolved rules.
*$IF(a, b, c)$ will return b if $a \geq 0$; otherwise it will return c.

In general, the evolved rules are not very complicated and are in forms that are explainable, especially for the DDARs, which are simply linear combinations of different terms. Scheduling Policy #1 is the one that achieves the best MAPE among the four SPs. Since Scheduling Policy #2 also employs the same DDAR, the better $\mathtt{MAPE}$ obtained by the first SP is strongly influenced by its DR. The first component of the DR in Scheduling Policy #1 will be negligible at the latter stage of the simulation since it has $\mathtt{RJ}$ in its denominator, which increases with the time. Therefore, this component has little impact on the performance of the scheduling policy and the performance of the rules will be governed by the second component. At the first glance, the second component is a combination of both SPT and FIFO because the priorities of jobs are either $-\mathtt{PR}$ (higher priority for jobs with smaller processing time) or $-\mathtt{RJ}$ (higher priority for jobs arriving at the machine earlier). The switch between FIFO and SPT is controlled by $\mathtt{max}(\mathtt{BWR},\mathtt{WR})-2\mathtt{SJ}$. In the case that the slack of jobs $\mathtt{SJ}$ is positive (not late) and larger than $\frac{1}{2}\mathtt{max}(\mathtt{BWR},\mathtt{WR})$, FIFO will be applied; otherwise SPT will be used. The purpose of this rule is to maintain a more predictable flow (by FIFO) of jobs when the jobs are not late, and to finish jobs with a smaller processing time first so as to reduce the number of jobs in the shop. These features make the flowtime prediction by $\mathtt{OT}+\mathtt{LOT}+\mathtt{QWL}$ more accurate because it is important for a shop to have a smooth flow of jobs.

Scheduling Policy #2 provides a better $\mathtt{C}_{\mathtt{max}}$ and $\mathtt{TWT}$ as compared to the first SP. Different from DR in the first SP, the DR in this SP emphasises more on reducing the time jobs stay in the shop as well as on reducing the lateness of jobs. In this case, if the remaining processing time $\mathtt{RT}$ of a job is large, the rule will give a higher priority to jobs with a higher $\mathtt{RT}$ in order to reduce its flowtime (and the makespan in general). In the case that jobs have large negative slacks, the rule will give a higher priority to jobs with larger negative slacks and reduce the lateness of jobs. However, because this rule will disturb the flow of jobs, the prediction will become less reliable.

Scheduling Policies #3 and #4 are the two SPs that provide the smallest makespans among the four. In general, the DRs in these two SPs give a higher priority for jobs with a larger $\mathtt{RT}$, $\mathtt{RO}$ and a smaller release time $\mathtt{rJ}$ in order to reduce the makespan. However, when these values are similar for the considered jobs, $\mathtt{W}$ is used to break the ties and gives a higher priority to jobs with higher weights. The focus on makespan has made the delivery performance of Scheduling Policy #3 worse as compared to the first and the second SPs. Scheduling Policy #4 tries to minimise $\mathtt{TWT}$ by overestimating the flowtimes for reducing the tardiness of jobs. The consequence is that the reliability of the due-date is deteriorated significantly.

From Scheduling Policies #1 to #4, $\mathtt{MAPE}$ tends to be increased and either $\mathtt{C}_{\mathtt{max}}$ or $\mathtt{TWT}$ is reduced, especially for $\mathtt{C}_{\mathtt{max}}$. Tracking the evolved SPs along this direction helps explain how the trade-offs can be achieved, mainly between $\mathtt{C}_{\mathtt{max}}$ and $\mathtt{MAPE}$. When observing the evolved SPs on the Pareto fronts along other directions, we are also able to explore other types of trade-offs, e.g., accepting a higher makespan for a better $\mathtt{TWT}$. Since the evolved SPs are constructed based on the genetic materials of other SPs or individuals, we can easily examine the connection among these SPs and understand what creates the trade-offs. In other words, the evolved DRs and DDARs are interpretable in this case.

VI. Conclusions

Designing an effective scheduling policy is challenging and time-consuming because it needs to take into account multiple scheduling decisions and conflicting objectives in a manufacturing system. The original contributions of our paper can be summarised as follows. First, the paper developed genetic programming based hyper-heuristics for automatic design of scheduling policies. The novelty here is on the representations and evolutionary search approaches to handling multiple scheduling rules and conflicting objectives in the evolution of scheduling policies. Four genetic programming based hyper-heuristics have been proposed in this paper. The performances of the proposed methods are examined by training and testing the evolved scheduling policies on various simulation scenarios. The results show that the evolved scheduling policies outperform the existing scheduling policies created from combinations of popular dispatching and dynamic or regression-based due-date assignment rules on both the training and testing scenarios. Moreover, the obtained Pareto fronts also provide much better knowledge about the space of potential scheduling rules, which cannot be achieved by simple combinations of existing scheduling rules or by methods using an aggregate objective function to handle multiple objectives. Another advantage of the proposed methods is that they perform well on unseen situations, which makes the evolved scheduling policies more robust when they are employed in stochastic and dynamic job shops. Analysis of the evolved scheduling policies also shows that the proposed methods can evolve not only effective but also very meaningful scheduling policies. In addition, it is easy to apply the proposed method to track the evolved scheduling policies along the Pareto front for better understanding of the trade-off among different objectives.

The second contribution is the development of a diversified multi-objective cooperative coevolution method, which shows favourable performance as compared to other popular evolutionary multi-objective search strategies. The experimental results show that the proposed DMOCC method can evolve Pareto fronts that are better than NSGA-II and SPEA2. It is also very competitive on all performance metrics for the training scenarios and provides a better spread of the evolved scheduling policies for the testing scenarios. Further analysis also shows that the representative selection strategies based on non-dominated rank and crowding distance play a very important role in the proposed cooperative coevolution for evolving well-distributed Pareto fronts. Another advantage of the coevolution approach is that multiple scheduling decisions can be evolved in different sub-populations in order to reduce the complexity of evolving the sophisticated scheduling policies. This method can be modified to take advantage of parallelisation techniques so as to reduce the computational time.

For future studies, we will enhance the performance of the proposed cooperative coevolution method by improving representation of the scheduling rules, genetic operations and strategies to explore optimal Pareto fronts. When examining the impact of the training scenarios, it has been observed that general training scenarios helped improve the quality of the evolved scheduling policies. However, this also increased the computational time, and thus we will also study decomposition approaches for better learning/evolving in the algorithm. The incorporation of other scheduling rules such as order review/release and order acceptance/rejection will also be considered. In addition, we want to extend the proposed methods to evolve heterogeneous scheduling rules for shops with specialised machines or groups of machines (batch processing, machines with sequence-dependent setup, etc.). Besides, we will further examine the performance of DMOCC on other multi-objective optimisation problems.

Appendix A
Statistical Pareto dominance

As compared to the literature related to probabilistic formulation of the Pareto dominance, the problem in our work has some special features. First, our problem needs to be handled with a probabilistic formulation rather than a robustness formulation (uncertainties defined by intervals) since the uncertainties in our problem come from a probability distributions [76]. Second, it is computationally expensive to evaluate the rules; and therefore it is very time consuming to perform a large number of simulation replications. Finally, our methods try to evolve rules in tree form which are very different from vectors of design variables in most studies in the literature. Because of these features, analytical or reliability approaches which depend on some assumptions on the uncertainties [76], [77] cannot be used, and the Pareto dominance under uncertainty in our work can only be determined via the pure sampling technique [76]. In order to examine the Pareto dominance of our evolved scheduling policies against the existing ones, we need to use statistical tests and introduce the notion of statistical Pareto dominance as opposed to probabilistic Pareto dominance [78], [79].

Traditionally, solution a is said to *Pareto-dominate* solution b if and only if $\forall i \in \{1, 2, \ldots n\} : f_i(a) \leq f_i(b) \wedge \exists j \in \{1, 2, \ldots n\} : f_j(a) < f_j(b)$ where n is the number of objective functions to be minimised. In this regard, scheduling policy a *statistically Pareto-dominates* scheduling policy b if and only if $\forall i \in \{1, 2, \ldots n\} : f_i(a) \leq_{\mathcal{T}} f_i(b) \wedge \exists j \in \{1, 2, \ldots n\} : f_j(a) <_{\mathcal{T}} f_j(b)$, where $f_i(a) \leq_{\mathcal{T}} f_i(b)$ means that a is significantly smaller than or not significantly different from b based on the statistical test $\mathcal{T}$ (*Wilcoxon signed-rank test* in this work); similarly, $f_j(a) <_{\mathcal{T}} f_j(b)$ means that a is significantly smaller than b based on $\mathcal{T}$.

Acknowledgment

This work is supported in part by the Marsden Fund of New Zealand Government (VUW0806 and 12-VUW-134), administrated by the Royal Society of New Zealand, and the University Research Fund (200457/3230) at Victoria University of Wellington.

References

[1] I. Ahmed and W. W. Fisher, "Due date assignment, job order release, and sequencing interaction in job shop scheduling," *Decision Sciences*, vol. 23, no. 3, pp. 633–647, 1992.

[2] P. M. Pardalos, O. V. Shylo, and A. Vazacopoulos, "Solving job shop scheduling problems utilizing the properties of backbone and "big valley"," *Computational Optimization and Applications*, vol. 47, pp. 61–76, 2010.

[3] R. M. Aiex, S. Binato, and M. G. C. Resende, "Parallel GRASP with path-relinking for job shop scheduling," *Journal of Parallel Computing*, vol. 29, pp. 393–430, April 2003.

[4] H. R. Lourenco, "Job-shop scheduling: Computational study of local search and large-step optimization methods," *European Journal of Operational Research*, vol. 83, no. 2, pp. 347–364, 1995.

[5] E. Balas, N. Simonetti, and A. Vazacopoulos, "Job shop scheduling with setup times, deadlines and precedence constraints," *Journal of Scheduling*, vol. 11, pp. 253–262, 2008.

[6] E. Balas and A. Vazacopoulos, "Guided local search with shifting bottleneck for job shop scheduling," *Management Science*, vol. 44, pp. 262–275, 1998.

[7] 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.

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

[9] T. C. E. Cheng and M. C. Gupta, "Survey of scheduling research involving due date determination decisions," *European Journal of Operational Research*, vol. 38, no. 2, pp. 156–166, 1989.

[10] G. L. Ragatz and V. A. Mabert, "A simulation analysis of due date assignment rules," *Journal of Operations Management*, vol. 5, no. 1, pp. 27–39, 1984.

[11] K. R. Baker, "Sequencing rules and due-date assignments in a job shop," *Management Science*, vol. 30, pp. 1093–1104, 1984.

[12] S. Miyazaki, "Combined scheduling system for reducing job tardiness in a job shop," *International Journal of Production Research*, vol. 19, no. 2, pp. 201–211, 1981.

[13] T. C. E. Cheng, "Integration of priority dispatching and due-date assignment in a job shop," *International Journal of Systems Science*, vol. 19, no. 9, pp. 1813–1825, 1988.

[14] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, 1992.

[15] W. Banzhaf, P. Nordin, R. Keller, and F. Francone, *Genetic Programming: An Introduction*. Morgan Kaufmann, San Francisco, 1998.

[16] S. Smith, "Cartesian genetic programming and its application to medical diagnosis," *IEEE Computational Intelligence Magazine*, vol. 6, no. 4, pp. 56–67, 2011.

[17] A. Song, Q. Shi, and W. Yin, "Understanding of GP-evolved motion detectors," *IEEE Computational Intelligence Magazine*, vol. 8, no. 1, pp. 46–55, 2011.

[18] T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach," in *GECCO '10: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation*, 2010, pp. 257–264.

[19] D. Jakobovic, L. Jelenkovic, and L. Budin, "Genetic programming heuristics for multiple machine scheduling," in *EuroGP'07: Proceedings of the 10th European Conference on Genetic Programming*, 2007, pp. 321–330.

[20] 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, pp. 453–473, 2008.

[21] C. D. Geiger and R. Uzsoy, "Learning effective dispatching rules for batch processor scheduling," *International Journal of Production Research*, vol. 46, pp. 1431–1454, 2008.

[22] C. Pickardt, J. Branke, T. Hildebrandt, J. Heger, and B. Scholz-Reiter, "Generating dispatching rules for semiconductor manufacturing to minimize weighted tardiness," in *Proceedings of the 2010 Winter Simulation Conference (WSC)*, 2010, pp. 2504–2515.

[23] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Learning iterative dispatching rules for job shop scheduling with genetic programming," *The International Journal of Advanced Manufacturing Technology*, 2013, DOI:10.1007/s00170-013-4756-9.

[24] ——, "A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem," *IEEE Transactions on Evolutionary Computation*, 2012, DOI:10.1109/TEVC.2012.2227326.

[25] E. K. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and R. Qu, "Hyper-heuristics: A survey of the state of the art," School of Computer Science and Information Technology, University of Nottingham, Tech. Rep. Computer Science Technical Report No. NOTTCS-TR-SUB-0906241418-2747, 2010.

[26] E. Burke, M. Hyde, G. Kendall, G. Ochoa, E. Ozcan, and J. Woodward, "Exploring hyper-heuristic methodologies with genetic programming," in *Computational Intelligence*, ser. Intelligent Systems Reference Library, C. Mumford and L. Jain, Eds. Springer Berlin Heidelberg, 2009, vol. 1, pp. 177–201.

[27] D. A.van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm research: a history and analysis," Department of Electrical and Computer Engineering, Air Force Institute of Technology, Ohio, Technical Report TR-98-03, 1998.

[28] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[29] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182 –197, 2002.

[30] D. Wojtaszek and S. Wesolkowski, "Military fleet mix computation and analysis," *IEEE Computational Intelligence Magazine*, vol. 7, no. 3, pp. 53–61, 2012.

[31] C. H. Chen, T. K. Liu, I. M. Huang, and J. H. Chou, "Multiobjective synthesis of six-bar mechanisms under manufacturing and collision-free constraints," *IEEE Computational Intelligence Magazine*, vol. 7, no. 1, pp. 36–48, 2012.

[32] R. Ramasesh, "Dynamic job shop scheduling: A survey of simulation research," *Omega*, vol. 18, no. 1, pp. 43–57, 1990.

[33] S. Nguyen, M. Zhang, M. Johnston, and K. C. Tan, "Evolving reusable operation-based due-date assignment models for job shop scheduling with genetic programming," in *EuroGP'12: Proceedings of the 15th European Conference on Genetic Programming*, 2012, pp. 121–133.

[34] T. D. Fry, P. R. Philipoom, and R. E. Markland, "Due date assignment in a multistage job shop," *IIE Transactions*, vol. 21, no. 2, pp. 153–161, 1989.

[35] P. R. Philipoom, L. P. Rees, and L. Wiegmann, "Using neural networks to determine internally-set due-date assignments for shop scheduling," *Decision Sciences*, vol. 25, no. 5-6, pp. 825–851, 1994.

[36] F.-C. R. Chang, "A study of due-date assignment rules with constrained tightness in a dynamic job shop," *Computers & Industrial Engineering*, vol. 31, pp. 205–208, 1996.

[37] T. C. E. Cheng and J. Jiang, "Job shop scheduling for missed due-date performance," *Computers & Industrial Engineering*, vol. 34, pp. 297–307, 1998.

[38] I. Sabuncuoglu and A. Comlekci, "Operation-based flowtime estimation in a dynamic job shop," *Omega*, vol. 30, no. 6, pp. 423–442, 2002.

[39] D. Sha and S. Hsu, "Due-date assignment in wafer fabrication using artificial neural networks," *The International Journal of Advanced Manufacturing Technology*, vol. 23, pp. 768–775, 2004.

[40] A. Ozturk, S. Kayaligil, and N. E. Ozdemirel, "Manufacturing lead time estimation using data mining," *European Journal of Operational Research*, vol. 173, no. 2, pp. 683–700, 2006.

[41] D. Sha and C.-H. Liu, "Using data mining for due date assignment in a dynamic job shop environment," *The International Journal of Advanced Manufacturing Technology*, vol. 25, pp. 1164–1174, 2005.

[42] D. Y. Sha, R. L. Storch, and C. H. Liu, "Development of a regression-based method with case-based tuning to solve the due date assignment problem," *International Journal of Production Research*, vol. 45, no. 1, pp. 65–82, 2007.

[43] A. Baykasoglu, M. Gocken, and Z. D. Unutmaz, "New approaches to due date assignment in job shops," *European Journal of Operational Research*, vol. 187, pp. 31–45, 2008.

[44] A. Jones and L. C. Rabelo, "Survey of job shop scheduling techniques," NISTIR, National Institute of Standards and Technology, Gaithersburg, USA, Tech. Rep., 1998.

[45] M. R. Garey, D. S. Johnson, and R. Sethi, "The complexity of flowshop and jobshop scheduling," *Mathematics of Operations Research*, vol. 1, no. 2, pp. pp. 117–129, 1976.

[46] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Computers & Operations Research*, vol. 13, pp. 533–549, 1986.

[47] S. Luke, *Essentials of Metaheuristics*. Lulu, 2009. [Online]. Available: http://cs.gmu.edu/∼sean/book/metaheuristics/

[48] E. Nowicki and C. Smutnicki, "A fast taboo search algorithm for the job shop problem," *Management Science*, vol. 42, pp. 797–813, 1996.

[49] R. Cheng, M. Gen, and Y. Tsujimura, "A tutorial survey of job-shop scheduling problems using genetic algorithms, part II: hybrid genetic search strategies," *Computers & Industrial Engineering*, vol. 36, no. 2, pp. 343–364, 1999.

[50] J. F. Goncalves, J. J. de Magalhaes Mendes, and M. G. C. Resende, "A hybrid genetic algorithm for the job shop scheduling problem," *European Journal of Operational Research*, vol. 167, no. 1, pp. 77–95, 2005.

[51] T. Yamada and R. Nakano, "A genetic algorithm with multi-step crossover for job-shop scheduling problems," in *GALESIA: First International Conference on Genetic Algorithms in Engineering Systems: Innovations and Applications*, 1995, pp. 146–151.

[52] D. Ouelhadj and S. Petrovic, "A survey of dynamic scheduling in manufacturing systems," *Journal of Scheduling*, vol. 12, no. 4, pp. 417–431, 2009.

[53] C. N. Potts and V. A. Strusevich, "Fifty years of scheduling: a survey of milestones," *Journal of the Operational Research Society*, vol. 60, pp. 41–68, 2009.

[54] K. N. McKay, F. R. Safayeni, and J. A. Buzacott, "Job-shop scheduling theory: What is relevant?" *Interfaces*, vol. 18, pp. 84–90, 1988.

[55] 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.

[56] M. L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, 3rd ed. Springer, 2008.

[57] S. S. Panwalkar and W. Iskander, "A survey of scheduling rules," *Operations Research*, vol. 25, pp. 45–61, 1977.

[58] C. Dimopoulos and A. M. S. Zalzala, "Investigating the use of genetic programming for a classic one-machine scheduling problem," *Advances in Engineering Software*, vol. 32, no. 6, pp. 489–498, 2001.

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

[60] W. J. Yin, M. Liu, and C. Wu, "Learning single-machine scheduling heuristics subject to machine breakdowns with genetic programming," in *CEC '03: IEEE Congress on Evolutionary Computation*, 2003, pp. 1050–1055.

[61] K. Miyashita, "Job-shop scheduling with GP," in *GECCO'00: Proceedings of the Genetic and Evolutionary Computation Conference*, 2000, pp. 505–512.

[62] L. Atlan, J. Bonnet, and M. Naillon, "Learning distributed reactive strategies by genetic programming for the general job shop problem," in *Proceedings of the 7th Annual Florida Artificial Intelligence Research Symposium*, 1994.

[63] D. Jakobovic and L. Budin, "Dynamic scheduling with genetic programming," in *EuroGP'06: Proceedings of the 9th European Conference on Genetic Programming*, 2006, pp. 73–84.

[64] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength pareto evolutionary algorithm for multiobjective optimization," in *Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN 2001)*, 2002, pp. 95–100.

[65] Z. Wang, K. Tang, and X. Yao, "Multi-objective approaches to optimal testing resource allocation in modular software systems," *IEEE Transactions on Reliability*, vol. 59, no. 3, pp. 563–575, 2010.

[66] C. K. Goh and K. C. Tan, "An investigation on noisy environments in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 3, pp. 354–381, 2007.

[67] ——, "A competitive-cooperative coevolutionary paradigm for dynamic multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 103–127, 2009.

[68] K. C. Tan, Y. J. Yang, and C. K. Goh, "A distributed cooperative coevolutionary algorithm for multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 527–549, 2006.

[69] M. A. Potter and K. A. de Jong, "Cooperative coevolution: An architecture for evolving coadapted subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.

[70] M. Thurer, M. Stevenson, C. Silva, M. Land, and M. G. Filho, "Workload control and order release in two-level multi-stage job shops: an assessment by simulation," *International Journal of Production Research*, 2012, (appear online).

[71] M. Land, "Parameters and sensitivity in workload control," *International Journal of Production Economics*, vol. 104, no. 2, pp. 625–638, 2006.

[72] A. P. J. Vepsalainen and T. E. Morton, "Priority rules for job shops with weighted tardiness costs," *Management Science*, vol. 33, pp. 1035–1047, 1987.

[73] D. A. van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm test suites," in *Proceedings of the 1999 ACM Symposium on Applied Computing (SAC'99)*, 1999, pp. 351–357.

[74] E. S. Gee and C. H. Smith, "Selecting allowance policies for improved job shop performance," *International Journal of Production Research*, vol. 31, no. 8, pp. 1839–1852, 1993.

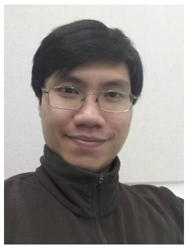
[75] D. C. Montgomery, *Design and Analysis of Experiments*. John Wiley & Sons, 2001.

[76] R. F. Coelho and P. Bouillard, "Multi-objective reliability-based optimization with stochastic metamodels," *Evolutionary Computation*, vol. 19, pp. 525–560, 2011.

[77] K. Deb, S. Gupta, D. Daum, J. Branke, A. K. Mall, and D. Padmanabhan, "Reliability-based optimization using evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1054–1074, 2009.

[78] E. Hughes, "Evolutionary multi-objective ranking with uncertainty and noise," in *EMO '01: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, 2001, pp. 329–343.

[79] J. Teich, "Pareto-front exploration with uncertain objectives," in *EMO '01: Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization*, 2001, pp. 314–328.



Su Nguyen received the B.E. degree in industrial and systems engineering from the Ho Chi Minh City University of Technology, Hochiminh City, Vietnam, in 2006, and the M.E. degree in industrial engineering and management from the Asian Institute of Technology (AIT), Bangkok, Thailand. He is currently pursuing the Ph.D. degree with the Evolutionary Computation Research Group, Victoria University of Wellington (VUW), Wellington, New Zealand.

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 (SM'10) received the B.E. and M.E. degrees from the Artificial Intelligence Research Center, Agricultural University of Hebei, Baoding, China, in 1989 and 1992, respectively, and the Ph.D. degree in computer science from RMIT University, Melbourne, Australia, in 2000.

Since 2000, he has been with the Victoria University of Wellington, Wellington, New Zealand. He is currently a Professor of computer science and heads the Evolutionary Computation Research Group. His current research interests include evolutionary computation, genetic programming, and particle swarm optimization with application areas of image analysis, multiobjective optimization, classification with unbalanced data, and feature selection and dimension reduction for classification with high dimensions.

Dr. Zhang is a member of the IEEE Computer Society, the IEEE CI Society, and the IEEE SMC Society. He is also a member of the IEEE CIS Evolutionary Computation Technical Committee, a member of the IEEE CIS Intelligent Systems Applications Technical Committee, a Vice Chair of the IEEE CIS Task Force on Evolutionary Computer Vision and Image Processing, and a Committee Member of the IEEE New Zealand Central Section. He is a member of the ACM and the ACM SIGEVO Group. He has been serving as an Associate Editor or Editorial Board Member for five international journals.

Mark Johnston (M'10) received the B.Sc. degree in mathematics and computer science and the Ph.D. degree in operations research from Massey University, Palmerston North, New Zealand.

He is currently a Senior Lecturer with the Victoria University of Wellington, Wellington, New Zealand, where he teaches various operation research courses. His current research interests include primarily combinatorial optimization and evolutionary computation, with a particular interest in scheduling models, genetic programming, and multiple-objective optimization.

Kay Chen Tan (SM'08) 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.

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.