A Coevolution Genetic Programming Method to Evolve Scheduling Policies for Dynamic Multi-objective Job Shop Scheduling Problems

Su Nguyen, Mengjie Zhang
School of Engineering and Computer Science
Victoria University of Wellington
PO Box 600, Wellington, New Zealand
Email: {su.nguyen, mengjie.zhang}
@ecs.vuw.ac.nz

Mark Johnston
School of Mathematics, Statistics
and Operations Research
Victoria University of Wellington
PO Box 600, Wellington, New Zealand
Email: mark.johnston@msor.vuw.ac.nz

Kay Chen Tan
Department of Electrical
and Computer Engineering
National University of Singapore
Singapore, 117576
Email: eletankc@nus.edu.sg

***Abstract*—A scheduling policy (SP) strongly influences the performance of a manufacturing system. However, the design of an effective SP is complicated and time-consuming due to the complexity of each scheduling decision as well as the interactions between these decisions. This paper proposes novel multi-objective genetic programming based hyper-heuristic methods for automatic design of SPs including dispatching rules (DRs) and due-date assignment rules (DDARs) in job shop environments. The experimental results show that the evolved Pareto front contains effective SPs that can dominate various SPs from combinations of existing DRs with dynamic and regression-based DDARs. The evolved SPs also show promising performance on unseen simulation scenarios with different shop settings. On the other hand, the proposed Diversified Multi-Objective Cooperative Coevolution (DMOCC) method can effectively evolve Pareto fronts of SPs compared to NSGA-II and SPEA2 while the uniformity of SPs obtained by DMOCC is better than those evolved by NSGA-II and SPEA2.**

I. Introduction

It has been commonly assumed that job shop scheduling (JSS) is equivalent to sequencing that determines the order of jobs waiting in the queue of a machine to be processed [1]. A large number of studies on JSS mainly focus on the sequencing part. However, sequencing is only one of several scheduling decisions. One of the other important activities in JSS is due-date assignment (DDA), sometimes referred to as estimation of job flowtimes (EJF). Due date assignment decisions are normally made whenever orders (jobs) are received from customers and good due-date assignments are needed in order to maintain high customer satisfaction.

Most studies on job shop scheduling have only considered one of the decisions and fixed the others in order to reduce the complexity of the scheduling problems, which is crucial for optimisation methods. However, these studies are only valid when there is no interaction between scheduling decisions, which is not the case in real world situations. Although JSS has been popular for decades and investigating the interactions between these decisions is essential for the development of an effective and comprehensive scheduling systems, studies on the interactions between scheduling decisions are quite limited. One of the reasons for the lack of research in this direction is that dealing with each scheduling decision has been already hard; therefore, considering multiple scheduling decisions simultaneously will be much more complicated. Since the 1990s, optimisation has been the dominant research trend in JSS. While there have been many great achievements, the focus on optimisation methods (mainly for a specific scheduling decision in a static environment) also restricts the studies of the interactions between scheduling decisions. In order to effectively tackle this problem, there is a need for a new methodology for improving scheduling decisions, which can cope with dynamic features of job scheduling problems.

Genetic Programming (GP) [2] is an evolutionary computation method which has been applied to evolve/train programs that are able to solve difficult computational problems. In this study, GP is used as a hyper-heuristic method [3] for the automatic design of scheduling policies (SPs) which include sequencing/dispatching rules (DRs) and due-date assignment rules (DDARs) for dynamic JSS problems. Compared with different machine learning methods, GP is considered as a good candidate for designing SPs because the flexibility of GP representation allows it to encode different scheduling rules into GP individuals that can be evolved through the GP evolution. Moreover, as an evolutionary approach, GP can employ available search mechanisms to handle multiple conflicting objectives of JSS problems. Another advantage of GP is that the scheduling policies can be interpreted, which will be useful for understanding how the evolved policies can solve the problems as well as how the trade-offs between different objectives in JSS can be obtained.

This paper aims to develop novel genetic programming based hyper-heuristic (GPHH) [4] methods to design SPs for dynamic multi-objective JSS problems. This is the first time GP is used to deal with DRs and DDARs simultaneously. In order to solve these problems, three key aspects will need to be considered: (i) representations of different scheduling rules, (ii) evolutionary search mechanisms to evolve the Pareto front of non-dominated SPs; and (iii) reusability of evolved SPs.

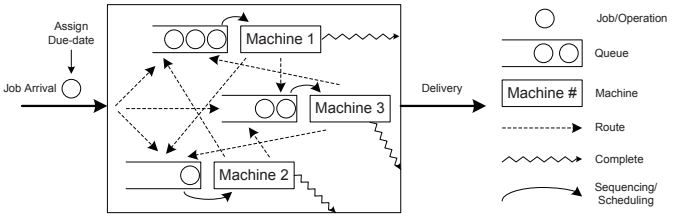


Fig. 1. Job Shop Scheduling (shop with 3 machines).

Three research objectives that we want to achieve in this study are: (1) developing multi-objective GPHH methods for automatic design of scheduling policies for dynamic JSS problems; (2) comparing the evolved scheduling policies with existing scheduling policies from existing combinations of dispatching rules and due-date assignment rules; and (3) evaluating the reusability of evolved scheduling policies.

In the next section, we provide the background of JSS problems and GPHH methods. Section III describes the proposed multi-objective GPHH methods and the job shop simulation models used for training and testing. The experimental results, the comparison and analysis of evolved scheduling policies are presented in Section IV. Section V gives some conclusions from this research and directions for future studies.

II. Background

A. Job Shop Scheduling

In the JSS problem, a number of jobs are to be processed, each including one or more operations to be performed in a specified sequence on specified machines and requiring certain amounts of times [5]. 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 needed to be made for jobs and machines in the shops such as due-date assignment, job order release and job scheduling. In this study, 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 job shop is shown in Fig. 1. In this figure, the due-date will be assigned to a newly arriving job by some due-date assignment rule (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) rules (DR) to be processed next.

Due-date assignment decisions are made whenever jobs (customer orders) are received from customers. Due-date can be set exogenously or endogenously [5]. In the former case, due-dates are decided by independent agencies (sellers, buyers). In this study, we only focus on the second case, in which the due-dates are internally set based on the characteristics of jobs and machines [5], to improve the delivery performance of job shops. Basically, the due-date of a new job is $d_j = r_j + \hat{f}_j$, where d_j is the due-date, r_j is the release time (in our study, the release time is the arrival time of the job since the job is released to the shop immediately), and $\hat{f}_j$ is the estimated (predicted) flowtime of job j. 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 $e_j = C_j - d_j = f_j - \hat{f}_j$, where f_j is the actual flowtime. Many DDARs have been proposed in the job shop literature. The traditional DDARs focus on exploiting the shop and job information to make a good flowtime estimation. Most of the early DDARs are based on linear combinations of different terms (variables) and the coefficients of the models are then determined based on simulation results. Regression (linear and non-linear) has been used very often in order to help find the best coefficients for the models employed [6], [7], [8]. Since the early 1990s, artificial intelligence methods were also applied to deal with due-date assignment problems such as neural networks [9], decision trees [10], regression trees [11], and a regression based method with case-based tuning [12]. Some dynamic DDARs that do not depend on the determination of coefficients are also proposed such as Dynamic Total Work Content (DTWK), Dynamic Processing Plus Waiting (DPPW) [6], and ADRES [13].

For job sequencing, each job in the queue of the machine will be assigned a priority based on the dispatching rules and the job with the highest priority will be considered for processing first. There have been a large number of rules proposed in the literature and they can be classified into three categories [14]: (1) simple priority rules, which are mainly based on the 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) [15] 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 [16] provided a comprehensive survey on dispatching rules used in research and real world applications using a similar classification.

The interaction between DRs and DDARs has also been studied [17], [18], [19], [20]. It has been noted that logical combinations of DRs and DDARs can enhance the performance the scheduling systems. Because the design of an effective DR or DDAR has been already time-consuming and complicated enough, developing a comprehensive scheduling policy with rational design of DRs and DDARs will be more challenging.

B. GPHH for Scheduling Problems

Recently, GP has become popular in the field of hyper-heuristics and it is known as genetic programming based hyper-heuristics (GPHH) [4]. Because GP is able to represent and evolve complex programs or rules, it naturally becomes an excellent candidate for heuristic generation. GPHH has

also been applied to evolve dispatching rules for scheduling problems. Dimopoulos and Zalzala [21] used GP to evolve dispatching rules for the one-machine scheduling problem 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 large and unseen instances. Also trying to learn new dispatching rules for the single machine environment, Geiger et al. [22] presented a learning system that combines GP with a simulation model of an industrial facility to learn dispatching rules for single and multiple machine scheduling problems. Comparison with the optimal rule in a simple two machine environment showed that the evolved rules are quite competitive.

Jakobovic and Budin [23] applied GP to evolve dispatching rules for both single machine and job shop environments. This study proposed a GP-3 system that evolves three components, a discriminant function and two dispatching rules. This function serves as the classifier in binary classification problems. Based on the classification decision obtained from the discriminant function, one of two dispatching rules will be selected to sequence jobs in the queue of that machine. The results show that GP-3 system performed better than traditional GP with a single tree. Unfortunately, no analysis of the evolved rules or demonstrations of the evolved rules are given. Tay and Ho [24] proposed a GP system to evolve dispatching rules for a multi-objective job shop environment. The multi-objective problem was converted into the single objective problem by linearly combining all 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. The set of instances was randomly generated and it was shown that the evolved dispatching rules can outperform other simple dispatching rules. However, they did not consider the use of machine attributes in the priority function. Hildebrandt et al. [25] re-examined this system in different dynamic job shop scenarios and showed that rules evolved by Tay and Ho [24] are only slightly better than the earliest release date (ERD) rule and quite far away from the performance of the SPT rule. They explained that the poor performance of these rules is caused by the use of the linear combination of different objectives and the fact that the randomly generated instances cannot effectively represent the situations that happened in a long term simulation. For that reason, Hildebrandt et al. [25] evolved dispatching rules by training them on four simulation scenarios and only aimed at minimising mean flow time. Some aspects of the simulation models were also discussed in their study. The experimental results indicated that the evolved rules were quite complicated but effective when compared to other existing rules. Moreover, these evolved rules are also robust when tested with another environment. From an interesting viewpoint, Vazquez-Rodriguez and Ochoa [26] proposed a GP system to learn variants of the Nawaz, En-score and Ham (NEH) procedure for flow shop scheduling. The results showed that the evolved heuristics can outperform the original and stochastic variants of NEH.

III. Multi-objective GP-HH methods for dynamic JSS problems

This section will describe the GPHH methods to evolve scheduling policies (SPs). We first show how due-date assignment rules (DDARs) and dispatching rules (DRs) are represented by GP individuals. Then, multi-objective GPHH methods to deal with dynamic JSS problems are proposed. Finally, the job shop simulation model used for training and testing is described.

A. Representations

1) Due-date Assignment Rules: The task of a DDAR is to determine flowtimes/due-dates by employing information from jobs and the shop. In this study, we want to evolve operation-based DDARs which will indirectly calculate job flowtimes through the accumulation of operation flowtimes in order to enhance the accuracy of the estimation by employing more detailed operation information. Here, we use the GP tree [2] to represent mathematical combinations of these pieces of information. For this reason, the function set will include four standard mathematical operators $+,-,\times$, and protected division $\%$, along with a conditional function `If` to allow GP to evolved sophisticated DDARs. Function `If` includes 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 sample of the 20 previous jobs processed at machine δ. `SAR`, on the other hand, is calculated based on the arrivals of the last 100 jobs.

Also, 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 of the new job, starting from the first operation. When $\hat{f}_o$ is obtained, a condition is checked to see whether the operation being considered 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 GP tree is applied to

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

`N`	number of jobs in the shop
`SAR`	sampled arrival rate
`APR`	average processing times of jobs in queue of the machine that processes ϕ
`OT`	processing time of ϕ
`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

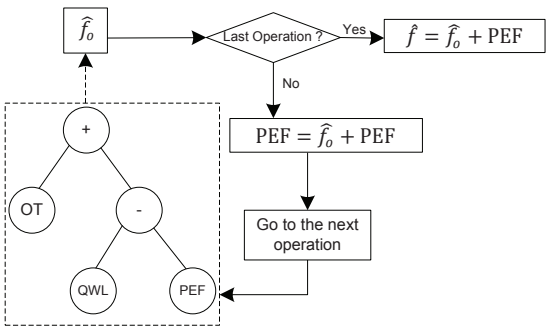


Fig. 2. Operation-based DDAR.

estimate the flowtime for the next operation. 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 flowtime $\hat{f}$. The evaluation scheme for DDAR is shown in Fig. 2. The use of PEF (initially zero for the first operation) in the terminal set of DDARs also provides them a chance to predict the changes of the system, given that PEF is well predicted.

2) Dispatching Rules: The composite dispatching rules (CDR) [15], [27] will be represented by GP tree and evolved in this study. 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 will be scheduled to be processed next on the machine. A function set similar to the one used for DDARs is also applied here along with `min`, `max` and `abs`, which commonly appear in existing CDRs. The terminal set for CDRs are 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 machine and shop status, where $p(\sigma)$ is the processing time of operation/job σ; Λ be the set of operations/jobs that are planned to visit the considered machine m; and K and I 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 $\sum_{\sigma \in \Omega} p(\sigma)$ with Ω is the set of jobs at the considered machine. An example of a CDR is shown in Figure 3.

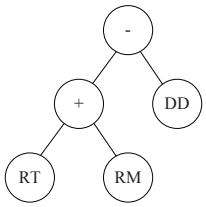


Fig. 3. Example of GP tree as a CDR.

B. Multi-objective GPHH Methods

As discussed, the aim of this study is to evolve scheduling policies which include two key components, due-date assignment rules and dispatching rules. The representation of these rules have been discussed previously; however, we need to figure out how to evolve these rules in our GPHH methods.

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
`#`	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 $= \frac{\sum_{\sigma \in \Omega^b} p(\sigma)}{\sum_{\sigma \in \Omega} p(\sigma)}$

In this study, we investigate two possibilities to deal with this issue. First, a GP individual will contain 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 the machine becomes idle and there are jobs in the queue of that machine to find the next job to be processed. Another way to evolve scheduling policies is to employ cooperative coevolution approach to evolve two decision rules in the two sub-populations. The scheduling policy is the combination of the individual in a sub-population with the representative individual in the another sub-population.

For the first option, we will apply NSGA-II [28] and SPEA2 [29] to explore the Pareto front of non-dominated scheduling policies similar to common applications of these two algorithms. For the second option, we propose a diversified multi-objective cooperative coevolution (DMOCC) method. With this approach, we employ an external archive to store the non-dominated scheduling policies and the *crowding distance* (individuals with higher *crowding distances* are located in less crowded areas of the objective space) as well as *non-dominated rank* [28] (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 operators and for cooperation between the two sub-populations. The overview of how DMOCC method works is shown in Fig. 4. In each generation, the objective values for each individual in the two sub-populations are obtained by joining that individual with a representative individual in the another population to form a complete scheduling policy for evaluation. *Representative individuals* for cooperations are selected based on *binary tournament selection* method [28], which randomly selects two individuals and the one with lower *non-dominated rank* will be chosen. In the case that two individuals have the same rank, the individuals with higher crowding distance will be selected. *Binary tournament selec-*

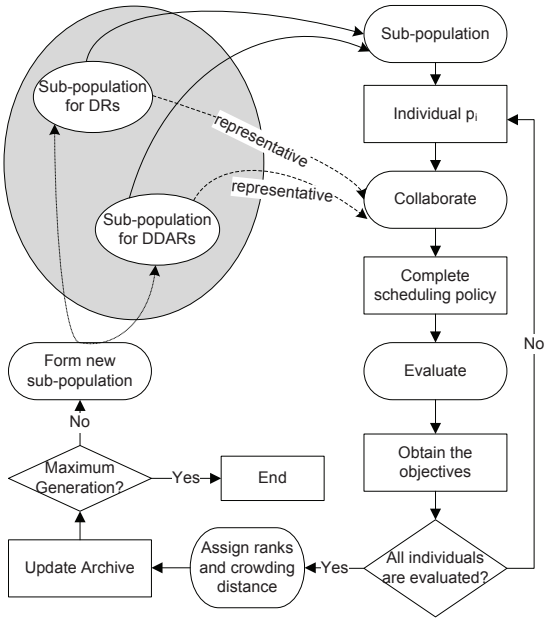


Fig. 4. Overview of DMOCC.

tion is employed in DMOCC because it is easy to implement and takes into account both the quality of the non-dominated individuals and their spread. 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. Different from NSGA-II and SPEA2, the size of the archive is not fixed but 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 are more than *maximum-size*, only individuals with the highest *crowding distance* are preserved in the archive.

When a 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 (C_{max}) [27]; (2) total weighted tardiness (TWT) [27] ; and (3) mean absolute percentage error (MPEA) [13]. C_{max} and TWT are two popular performance measures for evaluating dispatching rules or scheduling methods. On the other hands, MPEA is used to indicate the accuracy of the due-date assignment rules. In this study, we will try to evolve scheduling policies that minimise these three performance measures.

C. Genetic Operators

Traditional genetic operators are employed by the proposed GPHH methods. For crossover, GP uses the subtree crossover [2], 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 in the tournament. NSGA-II uses *binary tournament selection* method based on *rank* and *crowding distance* as explained in the previous section. For DMOCC, *binary tournament selection* is also used to select one parent from a sub-population and one parent from the archive. Since individuals in the archive are ones with *rank* zero, this selection is made only based on the *crowding distance* to turn the focus of the search to less crowded areas. Meanwhile, mutation is performed by subtree mutation [2], 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 and SPEA2, the genetic operators will first need to randomly choose which tree (either DR or DDAR) of the parents to perform the operators since each individual includes two trees for the two scheduling rules. If crossover is applied, only the genetic materials from the selected tree will be exchanged (e.g. the tree representing DR in a parent will only crossovered with the tree representing DR of the another parent).

D. Parameters for Proposed Multi-objective GPHH Methods

The initial population is randomly generated by ramped-half-and-half [2]. The crossover rate and mutation rate used in the three methods are 90% and 10%, respectively. The maximum depth of GP trees is 8. SPEA2 will use tournament selection of size 5 to select individuals for genetic operators. NSGA-II and SPEA2 use a population size of 200 while DMOCC use a population size of 100 for each sub-population to ensure that the number of program evaluations are the same for all methods. The archive size of SPEA2 and *maximum-size* for DMOCC are 200 individuals. The results will be obtained after the proposed methods runs for 100 generations.

E. Job Shop Simulation Model

In this study, we use a symmetrical (balanced) job shop simulation model in which each operation of a job has equal probabilities to be processed at any machine in the shop (a job visits each machine at most once). Therefore, machines in the shop have the same level of congestion in long simulation runs. This model has also been used very often in many studies in the JSS literature [6], [7], [25]. Based on the factors discussed above, the scenarios for training and testing of scheduling policies are shown in Table III.

Without loss of generality, the mean processing time of operations is fixed to 1 in these scenarios and the arrival of jobs will follow a Poisson process with the arrival rates adjusted based on the utilisation level. For the distribution of 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 the case that each job will have its number of operations equal to the number of machines in the shop. New jobs will be assigned random weights such that 20% of the jobs have weights of 1, 60% have weights of 2 and 20%

TABLE III
TRAINING AND TESTING SCENARIOS

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

TABLE IV
PERFORMANCE MEASURES OF SCHEDULING POLICIES

Makespan [27]	$\mathtt{C_{max}} = max_{j\in\mathbb{C}}\{f_j\}$
Normalised Total Weighted Tardiness [31]	$\mathtt{TWT} = \frac{\sum_{j\in\mathbb{C}} w_j T_j}{\lvert\mathbb{C}\rvert\times \mathtt{M}\times \frac{1}{\mu}\times \bar{w}}$
Mean Absolute Percentage Error [13]	$\mathtt{MAPE} = \frac{1}{\lvert\mathbb{C}\rvert}\sum_{j\in\mathbb{C}} \frac{\lvert e_j\rvert}{f_j}$

with weights of 4 [30]. 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 information of the next completed 5000 jobs is recored in set $\mathbb{C}$ to evaluate the performance measures of scheduling policies as shown in Table IV. 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 expected values of these performance accross simulation scenarios/replications are the objectives to be minimised by the proposed GP-HH methods.

In the training stage, since the simulation is very time-consuming, we only perform one replication for each scenario. There are $(2\times2\times1\times1) = 4$ simulation scenarios/replications used to evaluate the performance of the evolved scheduling policies. However, it should be noted that the performance measures are obtained for each scenario by applying 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. For testing, the obtained non-dominated scheduling policies from a run of GP is applied to $(3 \times 2 \times 2 \times 2) = 24$ simulation scenarios and 5 simulation replications are performed for each scenario; therefore, we perform $24 \times 5 = 120$ simulation replications to test the performance of obtained non-dominated scheduling policies. The use of a large number of scenarios and replications in the testing stage will help us confirm the quality and reusability of the evolved scheduling policies.

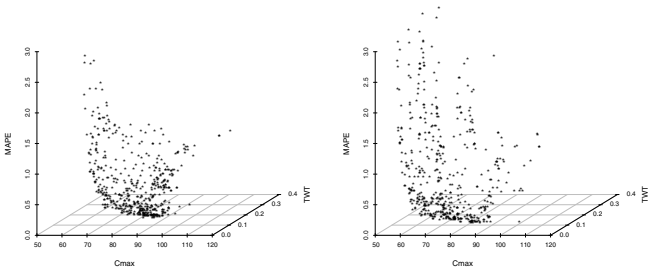
IV. RESULTS

A. Pareto Front of Evolved Scheduling Policies

Fig. 5(a) and Fig. 5(b) show the Pareto front extracted from evolved SPs obtained by all proposed GPHH methods in 30 independent runs. It is noted that the three objectives $\mathtt{C_{max}}$, $\mathtt{TWT}$ and $\mathtt{MAPE}$ are conflicting. The SPs that provide small $\mathtt{C_{max}}$ or $\mathtt{TWT}$ will result in flowtimes which cannot predicted accurately. In both testing and trainting scenarios, it is also observed that $\mathtt{C_{max}}$ and $\mathtt{TWT}$ can be significantly reduced by using SPs with $\mathtt{MAPE}$ smaller than 0.5. The use of more sophisticated SPs can provide slightly better $\mathtt{C_{max}}$ and $\mathtt{TWT}$ but they also make the job flowtimes much more difficult to be estimated.

B. Comparison to Existing DRs and Dynamic DDARs

The combinations of six popular DRs and three dynamic DDARs are evaluated on both training and testing scenarios and compared with the evolved non-dominated SPs shown in Fig. 5. 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) [16], weighted



(a) Training Scenarios (b) Testing Scenarios

Fig. 5. Pareto front of non-dominated scheduling policies.

Apparent Tardiness Cost (ATC) and weighted Cost Over Time (COVERT) [31]. The three dynamic DDARs are DTWK, DPPW [6] and ADRES [13]. These DDARs are selected for this comparison because they are well-known in the scheduling literature and the applications of these rules do not required predetermination of any parameter or coefficient for each simulation scenarios. The objective values obtained by these 18 combinations for training and testing scenarios are shown in Table V and Table VI.

TABLE V
PERFORMANCE OF EXISTING SCHEDULING POLICIES FOR TRAINING SCENARIOS ($\mathtt{C_{max}}, \mathtt{TWT}, \mathtt{MAPE}$)

	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 VI
PERFORMANCE OF EXISTING SCHEDULING POLICIES FOR TESTING SCENARIOS ($\mathtt{C_{max}}, \mathtt{TWT}, \mathtt{MAPE}$)

	DTWK	DPPW	ADRES
FIFO	(103.8, 0.53, 0.33)	(103.8, 0.33, 1.17)	(103.8, 0.13, 0.87)
CR	(152.7, 0.33, 0.30)	(110.6, 0.27, 1.05)	(165.0, 0.25, 1.19)
S/OPN	(117.6, 0.20, 0.21)	(84.5, 0.15, 0.51)	(123.5, 0.02, 0.95)
SPT	(264.6, 0.39, 0.37)	(264.6, 0.40, 0.61)	(264.6, 0.18, 2.01)
ATC	(211.1, 0.19, 0.28)	(201.1, 0.16, 0.49)	(137.5, 0.01, 1.07)
COVERT	(167.5, 0.16, 0.23)	(151.0, 0.12, 0.44)	(106.4, 0.02, 0.67)

When compared with non-dominated SPs obtained by each independent run of NSGA-II, SPEA2 and DMOCC, it is noted that these SPs are dominated by at least one of the evolved SPs in both training and testing scenarios. These results show that the non-dominated SPs evolved by the proposed multi-objective GPHH methods not only show good performance on training scenarios but also can 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. Four due-date assignment models used here are TWK, NOP, JIQ and JIS in combination with six dispatching rules in the previous section. Different from dynamic DDARs, the coefficients of the employed models have to be determined

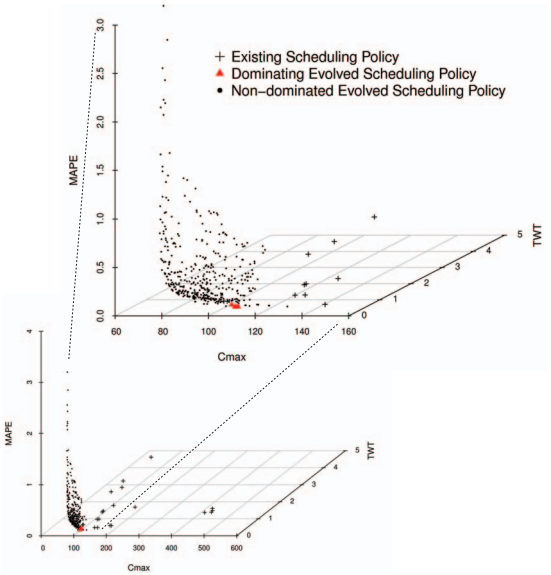
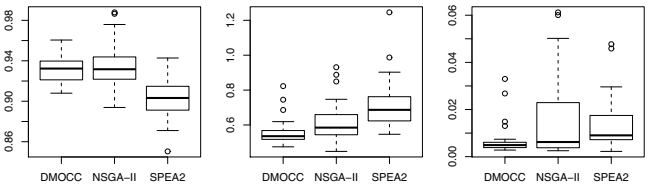
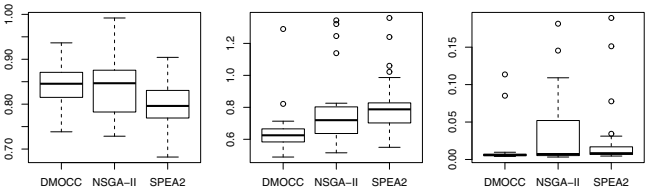


Fig. 6. DRs and Regression-based DDARs vs. evolved scheduling policies.



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

Fig. 7. Performance of multiobjective GP-HH methods on training scenarios.



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

Fig. 8. Performance of multiobjective GP-HH methods on testing scenarios.

by regression methods for each job shop setting. Fig. 6 shows performance of these $(6 \times 4) = 24$ combinations and the aggregate Pareto front of the non-dominated scheduling polices (by joining all evolved SPs from previous experiments) for the case with utilisation of 90%, 5 machines, *full* setting and processing times follow Exponential distribution. In this case, the coefficients of the due-date assignment models are determined by using Iterative Multiple Regression (IMR) [32]. The values in this figure are the average values of three objectives obtained from 30 independent simulation replications. The results show that 24 existing SPs considered here are significantly dominated (with significant level of 0.05 for *t-tests*) by at least one evolved SPs in the aggregate Pareto front. The dominating evolved SPs in Fig. 6 are evolved SPs that significantly better than all 24 existing SPs. This again shows the quality of the evolved SPs even when compared with customised SPs. Fig. 6 also reveals that the combinations of existing DRs and DDARs cannot cover all promising regions in the objective space. This observation suggests that automatic design methods like the proposed multi-objective GPHH methods are essential in order to provide informed knowledge about potential SPs.

D. Performance of Multi-objective GPHH Methods

Previous comparisons have confirmed the effectiveness of using multi-objective GPHH methods for evolving SPs. In this part, we will compare the ability of the proposed multi-objective GPHH methods to explore the Pareto front of non-dominated SPs. Similar to other multi-objective applications, we are also interested in the quality of the obtained Pareto front in term of (1) *convergence* to the good non-dominated solutions and (2) the *spread* or *distribution* of the solutions on the obtained Pareto front. Three well-known performance metrics for multi-objective problems used for this comparison are hypervolume ratio (HVR) [33], [34], SPREAD [28], and generational distance (GD) [35]. Since the true Pareto front is unknown for this application, we will use a reference Pareto front in the calculation of these performance metrics instead. In this study, the reference Pareto front includes non-dominated SPs extracted from all SPs found by the three multi-objective GPHH methods in 30 independent runs as shown in Fig. 5.

The performance indicators of the three multi-objective GPHH methods are shown in Fig. 7 and Fig. 8. With the training scenarios, *t-tests* (with significant level of 0.05) show that the HVRs obtained by DMOCC and NSGA-II are significantly better (higher) than that of SPEA2. It means that the SPs obtained by DMOCC and NSGA-II can significantly dominate those obtained by SPEA2. Regarding HVR, there is no significant different between DMOCC and NSGA-II but the standard deviation of HVR otbained by DMOCC is slightly smaller than that obtained by NSGA-II. For the distribution of the obtained SPs on the Pareto fronts, SPREAD obtained by DMOCC is significantly better than those obtained by NSGA-II and SPEA2. Even though, DMOCC still uses *crowding distance* like NSGA-II as the indicator for individuals in less crowded areas, the selection methods for choosing representative individuals and individuals for crossover has significantly improve the uniformity of the obtained Pareto fronts for 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. The performance of obtained non-dominated SPs on the testing scenarios is also very consistent with that in the training scenarios. These experimental results show that DMOCC is a very promising approach to SPs.

V. Conclusions

Three multi-objective genetic programming based hyper-heuristic methods are proposed in this paper for evolving Pareto front of scheduling policies including dispatching and due-date assignment rules for job shops. The experimental results show that the evolved scheduling policies can outperform various combinations of existing scheduling rules in

the literature. It is worth noticing that the evolved Pareto fronts show a much better knowledge about the space of potential scheduling rules compared to simple combinations of existing rules, which makes the proposed methods much more attractive to the decision makers. The performance on the unseen scenarios also shows the reusability of the evolved scheduling policies. This is a good feature that makes the scheduling policies more robust when employed in stochastic and dynamic job shops.

The diversified multi-objective cooperative coevolution methods proposed in this paper also show a favourable performance when compared with NSGA-II and SPEA2. The experimental results show that the proposed method can evolve scheduling policies that spread much better on the Pareto front as compared to NSGA-II and SPEA2. Another advantage of coevolution approach is that multiple scheduling decisions can be evolved by different sub-populations in order to reduce the complexity of evolving sophisticated scheduling policies. Also this methods can be modified to take advantage of the parallelisation techniques to reduce the computational times.

For future studies, we would like to perform extensive analysis on the evolved scheduling policies in order to understand how they can solve JSS problems and how the trade-offs between different objectives can be made. The focus will be on the choice of training scenarios and new reprentations of scheduling policies. It is also important to perform a sensitivity analysis on the parameter settings of the proposed multi-objective methods in order to enhance their performance. Moreover, it is interesting to extend the approach for other applications.

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

[3] 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.

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

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

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

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

[8] O. Joseph and R. Sridharan, "Analysis of dynamic due-date assignment models in a flexible manufacturing system," *Journal of Manufacturing Systems*, vol. 30, no. 1, pp. 28–40, 2011.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

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

[15] 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.

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

[17] 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.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] 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.

[24] 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.

[25] 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.

[26] J. A. Vazquez-Rodriguez and G. Ochoa, "On the automatic discovery of variants of the neh procedure for flow shop scheduling using genetic programming," *Journal of the Operational Research Society*, vol. 62, pp. 381–396, 2011.

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

[28] 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.

[29] 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.

[30] M. 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.