Investigating the Generality of Genetic Programming based Hyper-heuristic Approach to Dynamic Job Shop Scheduling with Machine Breakdown

John Park[1], Yi Mei[1], Su Nguyen[1,2], Gang Chen[1], Mengjie Zhang[1]

[1]Evolutionary Computation Research Group, Victoria University of Wellington, PO Box 600, Wellington, New Zealand
[2]Hoa Sen University, Vietnam
{John.Park,Yi.Mei,Su.Nguyen,Aaron.Chen,Mengjie.Zhang}@ecs.vuw.ac.nz

Abstract. Dynamic job shop scheduling (DJSS) problems are combinatorial optimisation problems that have been extensively studied in the literature due to their difficulty and their applicability to real-world manufacturing systems, e.g., car manufacturing systems. In a DJSS problem instance, jobs arrive on the shop floor to be processed on specific sequences of machines on the shop floor and unforeseen events such as dynamic job arrivals and machine breakdown occur that affect the properties of the shop floor. Many researchers have proposed genetic programming based hyper-heuristic (GP-HH) approaches to evolve high quality dispatching rules for DJSS problems with dynamic job arrivals, outperforming good man-made rules for the problems. However, no GP-HH approaches have been proposed for DJSS problems with dynamic job arrivals and machine breakdowns, and it is not known how well GP generalises over both DJSS problem instances with no machine breakdown to problem instances with machine breakdown. Therefore, this paper investigates the generality of GP for DJSS problem with dynamic job arrivals and machine breakdowns. To do this, a machine breakdown specific DJSS dataset is proposed, and an analysis procedure is used to observe the differences in the structures of the GP rules when evolved under different machine breakdown scenarios. The results show that performance and the distributions of the terminals for the evolved rules is sensitive to the frequency of machine breakdowns in the training instances used to evolve the rules.

1 Introduction

In the field of operations research, job shop scheduling (JSS) and other scheduling problems have been extensively researched for the past 50 years [1]. In a JSS problem, there is a *shop floor* that usually contains a fixed number of *machines* and the machines are used to process incoming *jobs* [1]. A job needs to be processed on a sequence of specific machines and machines on the shop floor can only process one job at a time. The goal in a JSS problem is to find a *schedule*, a solution that gives the sequences of times that the jobs are processed at the machines, that is the optimal given an *objective function* [1]. For example, in a JSS problem with makespan as the objective, the goal is to find a schedule which completes all jobs as early as possible [1].

In a real-world scenario, it is likely that unforeseen events such as dynamic job arrivals and machine breakdowns affect the properties of the shop floor in a JSS problem instance [2]. A JSS problem instances with dynamic job arrivals is called a *dynamic* JSS (DJSS) problem instance [2]. In a DJSS problem instance with dynamic job arrivals, the properties of the arriving jobs are unknown (until they arrive on the shop floor) and the number of jobs that arrive on the shop floor is unknown. To handle DJSS problems with dynamic job arrivals, researchers have proposed various dispatching rule approaches. Dispatching rules [1] are iterative heuristics that determine the job that is selected to be processed by the machine when it is available. This decision process for determining the job that is selected by the available machine is called a *decision situation* [3]. Dispatching rules are effective for DJSS problems with dynamic job arrivals because they can react quickly to the arrival of new jobs and can cope with the dynamic environment [4]. In addition, because they are easy to interpret by operators on the shop floor [4], they are used extensively in real-world manufacturing environments, e.g., semi-conductor manufacturing [5]. However, a limitation of dispatching rule approaches is that they are tailored to a specific JSS problem. Although humans are very good at identifying good building blocks for heuristics [6], constructing effective heuristics from the building blocks require extensive trial-and-error testing [4, 5]. Therefore, genetic programming based hyper-heuristic (GP-HH) approaches have been proposed in the literature to automatically evolve dispatching rules for DJSS problems [7]. GP-HH is provided heuristic building blocks, DJSS problem instances for training and searches in the heuristic space to find high quality solutions for the DJSS problem [6]. It has been shown in the literature that GP evolved rules generally perform better than man-made rules for different DJSS problems [7].

There are several factors that need to be considered to develop an effective GP-HH approach to DJSS problems. One of the factors is that evolved rules need to *generalise* well [6, 7]. In other words, the evolved rules trained over a specific problem domain needs to perform well on unseen problem instances both within and outside the problem domain. Generality has been covered in the literature for DJSS problems with dynamic job arrivals [7]. However, although generality of GP for DJSS with dynamic job arrivals have been investigated [4, 5, 7], it is not known how well GP can generalise for over DJSS problem instances with no machine breakdowns and with machine breakdowns. When a machine breakdown occurs, any job that is being processed on the machine is interrupted and the machine needs to be repaired for a specific amount of time before it is back "online" again. Machine breakdowns can severely disrupt the processing that occurs on the shop floor.

1.1 Goal

The goal of this paper is to investigate the *generality* of GP for DJSS problems with dynamic job arrivals and machine breakdowns, and to analyse the terminals that are effective for DJSS problem instances with machine breakdowns. By analysing the generalisation ability of the evolved rules, it may be possible to determine whether the standard GP-HH approach is suitable for the DJSS problem with machine breakdowns. Otherwise, if the standard GP-HH approach cannot generalise well over the DJSS problems with machine breakdowns, then the analysis of the terminals may provide insight

for developing new extensions to the standard GP-HH approach that are more effective for the DJSS problem. To achieve the goal, this paper carries out the following objectives:

(a) Develop a new DJSS dataset for generating problem instances with dynamic job arrivals and machine breakdown.
(b) Investigate the generality of an existing GP-HH [4, 8] by evolving and evaluating the rules over different combinations of machine breakdown scenarios.
(c) Analyse the structure of the GP rules to extract information on the distributions of the terminals for the evolved rules.

1.2 Organisation

First, we cover the background to DJSS in Section 2, which provides the problem definitions and outlines sample GP-HH approaches to DJSS problems. Afterwards, Section 3 describes the testing framework that is used to test the generality of GP-HH approach for the DJSS problem. Section 4 describes the benchmark GP-HH approach that is used to evolve the rules, the fitness function and the GP parameters. Finally, Section 5 gives the results and an analysis of the findings, and Section 6 gives the concluding remarks and future works.

2 Background

This section covers the problem definition, including the notations used and the description of the DJSS problem with dynamic job arrivals and machine breakdowns. It then discusses the related work to DJSS problems in the literature.

2.1 Problem Definition

We use the following notation for the DJSS problem handled by the GP-HH approach. There are M machines on the shop floor, and a job j arrives on the shop floor with the sequence of operations $\sigma_{1j}, \ldots, \sigma_{(N_j)j}$. The processing time for the operation σ_{ij} is denoted as p_{ij}, and the operation's ready time is denoted as r_{ij}. In addition, the time when a job arrives on the shop floor is abbreviated to r_j. A job j also has a due date d_j and a weight w_j. If the completion time C_j is greater than the job's due date d_j, then the job is tardy and has tardiness $T_j = C_j - d_j$. From this, the mean weighted tardiness (MWT) for a schedule is defined as $\frac{1}{N}\sum_j^N w_j T_j$ [1]. MWT and other tardiness objectives have commonly been used in the literature to evaluate the effectiveness of heuristics for DJSS problems [4, 5, 8].

The two dynamic components for the DJSS problem are dynamic job arrivals and machine breakdowns. In a DJSS problem instance with dynamic job arrival, a job j's attributes are unknown until job j arrives on the shop floor at time r_j. On the other hand, machine breakdowns are unforeseen events where a machine m is shut down at time ${b_t}^m$, and requires ${t_r}^m$ time to repair the machine. During the repair time, the machine is unable to process any new operations. If a job's operation is being processed on the

machine at the moment of the machine breakdown, then the operation is suspended and resumed after the machine is repaired and is available. In other words, if a job j's operation σ_{ij} was started at s_{ij} at machine m before the machine breaks down at time ${b_t}^m$ and requires ${t_r}^m$ time to repair. Then the job j's operation is resumed at time ${b_t}^m + {t_r}^m$, and the operation completes at $s_{ij} + p_{ij} + {t_r}^m$. This definition of machine breakdown was proposed by Holthaus [9].

2.2 Related Work

Many researchers have developed priority-based dispatching rule approaches to handling different DJSS problems. Examples of priority-based dispatching rule are cost over time (COVERT) rule and apparent tardiness cost (ATC) rule [10] for JSS problems with tardiness objectives. Additionally, other effective man-made dispatching rule approaches have been proposed and investigated in the literature for JSS problems [1]. In particular, Holthaus [9] investigates the performances of man-made dispatching rules that DJSS problems with dynamic job arrivals and machine breakdowns for different objective functions. He showed that for due date related objectives, including MWT, the rules have different performances for the different configurations associated with two attributes: the average time it takes for the machine to be repaired and the approximate amount time the machines are broken down for over the duration of the job processing.

In addition to manually designing dispatching rules, there have been many GP-HH approaches that evolve dispatching rules to DJSS problems in the literature [3–5, 7, 8]. In addition, several of the GP-HH approaches to JSS also investigate the generality of GP evolved rules by applying them to different problem domains [4, 5]. Nguyen et al. [4] showed that GP rules evolved using static JSS problem instances do not perform as well as some man-made dispatching rules (such as the ATC rule) in a DJSS problem with dynamic job arrivals. Therefore, if problem instances are encountered outside of the problem domain that the rules were trained on, new rules may need to be evolved to be competitive with existing approaches to the problem instances. On the other hand, Burke et al. [6] proposed a method improving the generality of rules evolved by GP for a 2-D bin packing problem. They generate both "best-fit" GP rules by applying the GP over training sets consisting of problem instances with specific properties and "generalist" GP rules by evolving the rules over problem instances with different properties. They showed that GP can evolve good reusable rules, and general rules can sometimes outperform best-fit rules. Finally, Branke et al. [7] provides a survey of various evolutionary computation approaches to scheduling problems in the literature that also addresses the generality of GP evolved rules.

JSS problems with machine breakdowns have also been covered extensively in the literature. Many research have proposed predictive-reactive [2] approaches to DJSS problems with machine breakdowns. Predictive-reactive algorithms first generate an initial schedule for the DJSS problem instance, and then generates a new schedule, i.e., reschedules, when a machine breaks downs during processing. They have been effectively applied to JSS problems where the job properties are known *a priori*, but are not suitable for DJSS problems with dynamic job arrivals, as the schedule needs to constantly be updated with the arrival of new jobs. Therefore, for DJSS problems with dynamic job arrivals and machine breakdowns, completely-reactive approaches [2, 9],

where the schedule is generated during processing, have been proposed. This includes dispatching rule approaches. Ouelhadj and Petrovic [2] provides a survey for various approaches to DJSS problems with machine breakdowns.

3 Framework for Investigating the Generality of GP

This section describes the framework that is used to investigate the generality of GP-HH for DJSS problems with dynamic job arrivals and machine breakdowns. This covers the DJSS dataset containing the problem instances with machine breakdowns, how the rules are evolved from problem instances with different machine breakdown scenarios from the dataset, and the procedure for analysing the structures of the GP evolved rules.

3.1 Generating DJSS Problem Instances using Simulations

The standard approach in the literature to generate DJSS problem instances is to use discrete-event simulations [3, 5, 8, 9]. This means that the job arrivals, the machine breakdown events and the repair times for the breakdowns are generated stochastically. For this paper, the dataset Δ used to evaluate the generality of GP is modified from the dataset proposed by Holthaus [9], which has been used effectively to evaluate and analyse man-made dispatching rules in DJSS problems with dynamic job arrivals and machine breakdowns. The following parameters are kept consistent as the ones originally used by Holthaus. The problem instances have $M = 10$ machines on the shop floor. The processing times for an operation for a job is generated from a uniform distribution between $[1, 49]$. In other words, the mean processing time of the operations is $\mu = 25$. For generating an arriving job, the arrival times of the jobs are generated according to a Poisson process with mean λ. The utilisation rate is a standard parameter used in DJSS discrete-event simulations that defines the expected proportion of time that the machines are occupied processing the jobs against the total duration of simulation [5, 9]. Because of this, the mean arrival time is often defined by the utilisation rate of the problem instances, and is given in Equation (1) [5, 9]. In the equation, ρ is the utilisation rate and p_M is the expected number of operations per job divided by the number of machines. The utilisation rate is set to $\rho = 0.9$, which is consistent with Holthaus's dataset. For our paper, there is no re-entry, i.e., a job cannot have at least two separate operations on the same machine [1]. This means that a job can have at most 10 operations, and the number of operations per job is modified to be random between 2 to 10 operations, i.e., $p_M = (2 + 10)/2 = 6$.

$$\lambda = \frac{\rho \times p_M}{(1/\mu)} \tag{1}$$

The due date of arriving job j $d_j = r_j + h \sum_{i=1}^{N_j} p_{ij}$, where h multiplied to the sum processing times of the job is the due date tightness factor. Tightness of $h = 3$ and $h = 5$ are used, where $h = 3$ represents tight due dates and $h = 5$ loose due dates. This is adjusted from the original tightness values of $h = 4, 8$ used by Holthaus [9], as preliminary experiments found that due date tightness $h = 8$ resulted in GP evolved rules generating schedules for problem instances where the MWT values are zero. The

weight of a job is either 1, 2, or 4 with probabilities 0.2, 0.6 and 0.2 respectively, which is a standard method of generating weights for jobs in due-date related DJSS problems [4, 8]. For each problem instance, there is a "warm-up" period of 500 jobs which do not contribute towards the objective value, and jobs continue arriving until the 2500th job has been completed. However, all jobs that have arrived on the shop floor need to be completed before the problem instance is completed. From Holthaus's dataset [9], the machine repair times and the times between machine breakdowns (excluding the repair times) are exponentially distributed. The mean repair time (RTM) and the mean time between machine breakdowns (BTM) are the same for all machines on the floor. In addition, for the configuration used to generate a problem instance, RTM depends on the mean processing times of the operations μ and the machine breakdown level parameter (BL). The machine breakdown level can be considered as the proportion of time the machine is being repaired during processing, e.g., if BL $= 0.025$ and the all jobs took 2500 time units to process, then the total repair time for all machines is approximately $0.025 \times 2500 = 62.5$ time units. In other words, the machine breakdown level is given by BL $=$ RTM$/($BTM $+$ RTM$)$, which means that BTM $=$ RTM$/$BL $-$ RTM [9]. The dataset has variable configurations for the following parameters: due date tightness (h), mean repair times of machines (RTM) and breakdown level (BL). The configurations can have RTM $\in \{\mu, 5\mu, 10\mu\}$ and BL $\in \{0, 0.025, 0.05\}$. Overall, the two due date tightness configurations and the configurations for the machine breakdowns results in a total of 18 different configurations.

3.2 GP-HH Training Procedure

To evolve and evaluate the GP rules, different subsets of DJSS problem instances in the dataset are used to evolve different sets of GP rules. Figure 1 shows an overview of how the dataset Δ is used to evolve different sets of GP rules that are either "generalists" or "best-fit" over the machine breakdown level (BL). The generalist rules are designed to be effective for the different machine breakdown scenarios, whereas the best-fit rules [6] are designed to be effective for specific machine breakdown scenarios. For the scope of this paper, the machine breakdown level parameter allows us to analyse the generality of GP rules in DJSS problems with dynamic job arrivals and machine breakdowns. First, the dataset Δ containing 18 different configurations for generating problem instances is partitioned into three subsets based on the machine breakdown level. In the subsets, machine breakdown level BL $= 0$ means that the generated problem instances do not have machine breakdowns, BL $= 0.025$ and BL $= 0.05$ means that the generated problem instances have "medium" and "high" levels of machine breakdowns respectively. The subsets are denoted as Δ_N, Δ_M and Δ_H respectively and contain six different configurations. The best-fit rules are evolved from Δ_N, Δ_M and Δ_H and are designed to cope with the specific level of machine breakdown. Additionally, $\Delta_{N/M}$ and $\Delta_{M/H}$ combine two smaller subsets together (e.g. Δ_N and Δ_M for $\Delta_{N/M}$, and are used to evolve "intermediate" sets of rules. If the intermediate rules are competitive by the best-fit rules, e.g., rule evolved from Δ_H does not perform significantly worse than $\Delta_{M/H}$ for problem instances with BL $= 0.05$, then it is likely that GP can generalise well over different machine breakdown scenarios even without incorporating information about machine breakdowns. Finally, all possible configurations in the dataset Δ,

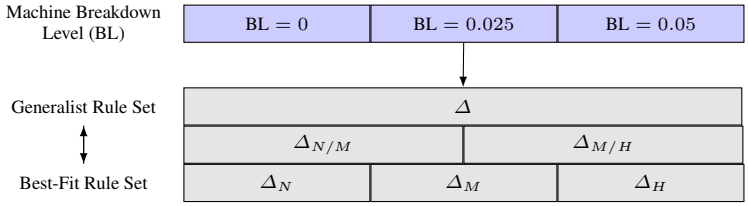


Fig. 1: Overview of how the dataset used for the DJSS problem is partitioned to train GP rules specialised for different machine breakdown configurations.

i.e., configurations from Δ_N, Δ_M and Δ_H combined, are used to evolve the final set of general rules. Overall, this results in a total of 6 sets of GP rules that range from generalists to best-fit over the DJSS problem. This procedure was first covered by Burke et al. [6] for improving the generality of the GP-HH approach for a bin packing problem.

The set of rules evolved from a specific training set is denoted as 'DR-' with the suffix as the training set. For example, DR-N denotes the set of GP evolved rules which have been evolved from subset Δ_N, i.e., problem instances with no machine breakdowns. In addition, at each generation, the seeds used to stochastically generate the jobs and the machine breakdowns are rotated for the training procedure of the GP-HH. This means that the problem instances used in one generation will be different to the problem instances used for the next generation. This has shown to improve the generalisation ability of the evolved rules for DJSS problems [5].

3.3 Rule Terminal Analysis Procedure

For analysing the effectiveness of GP rules, existing literature have proposed methods where small numbers of rules are sampled from the sets of evolved rules and the tree structures of the rules are analysed [5, 4, 8]. However, for investigating the generality of GP over different machine breakdown scenarios, it may be more effective to analyse the structures of entire sets of rules instead of sampling specific rules from the sets, as GP needs to be able to evolve good rules consistently. However, it is too cumbersome to directly analyse the tree structures of the sets of rules directly. Therefore, the distributions of the terminals that make up the GP rule is analysed. For example, if the due date terminal occurs more frequently for the rules in DR-H than the rules in DR-N, then it means that processing urgent job is more important for problem instances with high level of machine breakdowns than problem instances with no machine breakdowns. To calculate the distribution, the proportion of the terminals that make up an evolved rule is first calculated. For example, suppose that an evolved rule has 23 PT terminals out of 150 terminals that make up the rule. Then the $23/150$ is the proportion of PT terminals that make up the rule. Afterwards, the proportions are normalised over the set of evolved rules.

4 Experimental Design

This section covers the benchmark GP-HH approach that is investigated for this paper, which is based off existing GP-HH approaches. This includes the GP representation,

Table 1: Terminal set for GP

Standard	RJ	operation ready time of job j
	PT	operation processing time of job j
	RO	remaining number of operations of job j
	RT	remaining total processing times of job j
	RM	machine m^*'s ready time
	DD	due date d_j
	W	job's weight w_j
	#	random number from 0 to 1
Look-ahead	NPT	next operation processing time of job j
	NNQ	number of idle jobs waiting at the next machine
	NQW	average waiting time of last 5 jobs at the next machine
	AQW	average waiting time of last 5 jobs at all machines

terminal set, function set and fitness measure used for the individuals. Afterwards, the parameters used for the GP-HH is detailed.

4.1 GP Representation, Terminals and Function Sets.

The most prominent method of evolving priority-based dispatching rule using GP is to use a tree-based GP, where the individuals represent arithmetic function trees [7]. For this paper, we modify the arithmetic representation proposed by Nguyen et al. [4] to evolve priority-based dispatching rules. In addition, a look-ahead terminal set proposed by Hunt et al. [8] that have effectively been applied to a DJSS problem with unforeseen job arrivals will be incorporated into the terminal set. The list of terminals used for the GP-HH process are shown in Table 1 for a job j waiting at machine m^*.

The function set includes of the arithmetic operators $+$, $-$, $\times$, and protected $/$, where protected $/$ returns one if the denominator is zero. The rest of the operators for the function set are `if`, which returns the value of the second branch if the value of the first branch is greater than or equal to zero or returns the value of the third branch otherwise, $\max$ and $\min$ operators.

4.2 Calculating a GP Individual's Fitness

For the evaluation procedure, the individuals are applied to the DJSS training instances as *non-delay* dispatching rules [1]. A non-delay dispatching rule greedily attempts to minimise the idle time from when a machine is available to when it starts processing the next job [1]. From this, the MWT over the training instances is normalised using the ATC rule, where a standard $k = 3$ value is used for the ATC parameter [10]. The normalisation procedure have been used in the literatures [5] to reduce the bias towards specific problem instances that are more likely to have a higher optimal MWT values than other problem instances in the training set. Given that $\text{MWT}_{\omega,\gamma}$ and $\text{MWT}_{ref,\gamma}$ are the MWT of the schedule generated by individual ω and the reference rule for training instance γ respectively, the fitness f_ω of an individual ω is given in Equation (2).

$$f_\omega = \frac{1}{|\Delta_{train}|} \sum_{\gamma \in \Delta_{train}}^{T_{train}} \frac{\text{MWT}_{\omega,\gamma}}{\text{MWT}_{ref,\gamma}} \tag{2}$$

4.3 GP Parameter Settings

The parameters used for GP are modified from the parameters used by GP-HH approaches to DJSS problems in the literature [4, 8] after carrying out parameter tuning on the population size and the crossover, mutation and reproduction rates. After the parameter tuning, the population size is set to 256 to reduce the computational cost, and the number of generations is set to 51. The crossover, mutation and reproduction rates are 80%, 10% and 10% respectively. The maximum depth of an individual is 8, and the maximum depth of an individual that can be initialised is 2. Tournament selection of size 7 is used during the selection process. Finally, the parameter value used for the ATC reference rule is set to $k = 3.0$ [10].

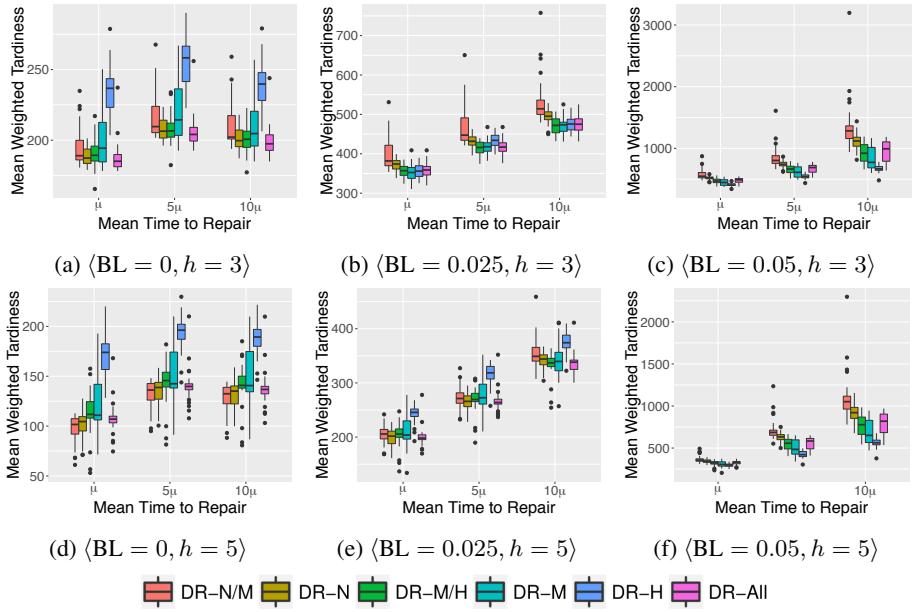
5 Experimental Results

This section covers the evaluation of the GP-HH approach over the DJSS problem with dynamic job arrivals and machine breakdowns. First, 30 independent runs of the GP processes are carried out over the training sets, resulting in DR-N, DR-M, DR-H, DR-N/M, DR-M/H and DR-All each consisting of 30 rules. The sets of GP evolved rules are applied to the problem instances in the test set, and the qualities of the schedules are compared against each other as part of the general evaluation procedure. Afterwards, an analysis on the structures of the evolved rules is carried out.

5.1 Evolved Rule Performance Evaluation

Each configuration in the test set is used to generate 30 different problem instances as part of the test set. In total, this results in a total of $18 \times 30 = 540$ test instances using the 18 different configurations. The sets of GP evolved rules are then applied to the test instances to generate schedules for the problem instances, and the MWT of the schedules are compared against each other as part of the general evaluation procedure. A set of GP evolved rules is *significantly* better than another rule set if the difference in the MWT values satisfies the two sided Student's t-test at $p = 0.05$. The performances of the rule sets over the different problem instances are shown in Figure 2. Each box plot shows the results over problem instances in a configuration, and the configurations are categorised by the breakdown level and due date tightness, where $\langle \text{BL} = 0.025, h = 3 \rangle$ denotes that breakdown level is 2.5% and due date tightness is 3.

From the results, we can see that for the problem instances generated with $\text{BL} = 0$ and $\text{BL} = 0.05$ DR-N and DR-H generally perform well over the respective problem domain they are trained on, but perform poorly on problem instances with high level of machine breakdowns (for DR-N) and problem instances with no machine breakdowns (for DR-H). Under the statistical test, the difference in the performance is significant



(a) $\langle \text{BL} = 0, h = 3 \rangle$ (b) $\langle \text{BL} = 0.025, h = 3 \rangle$ (c) $\langle \text{BL} = 0.05, h = 3 \rangle$

(d) $\langle \text{BL} = 0, h = 5 \rangle$ (e) $\langle \text{BL} = 0.025, h = 5 \rangle$ (f) $\langle \text{BL} = 0.05, h = 5 \rangle$

Fig. 2: The comparisons of the mean weighted tardiness performances of the GP rules evolved over different training sets over the problem instances in the test set.

between DR-N and DR-H. When the best-fit rules DR-N and DR-H are compared to the intermediate rules DR-N/M and DR-M/H, the two best-fit rules perform slightly better than the intermediate rules over their respective machine breakdown levels the specialise rules are evolved on. The difference in the performances are significant between DR-H and DR-M/H, but not between DR-N and DR-N/M. However, on machine breakdown level BL $= 0.025$, it is observed that most sets of rules with the exception of DR-H have a similar performance to each other, where the slight differences in the performances are not significant. Finally, the generalist rule DR-All perform well over problem instances with no machine breakdowns and problem instances with machine breakdown level BL $= 0.025$, but performs significantly worse than DR-H and DR-M/H for problem instances with BL $= 0.05$. Overall, it may be likely that standard GP-HH approach may not be able to generalise well when it comes to DJSS problems with dynamic job arrivals with machine breakdowns, and the quality of the rules evolved by a standard GP-HH approach is likely to sensitive to the proportion of time that the machine is broken down during the simulation.

5.2 GP Terminal Distribution Analysis

After evaluating the performances of the GP evolved rules, the terminals that are used by the GP rules are compared against each other to analyse the make up of the rules. For the sets of GP rules, the proportion of the rule structure made up of the terminals are shown in Figure 3.

From Figure 3, the most prominent terminals that are used by all sets of rules is the processing time of current operation (PT), followed by the due date of the job (DD) and

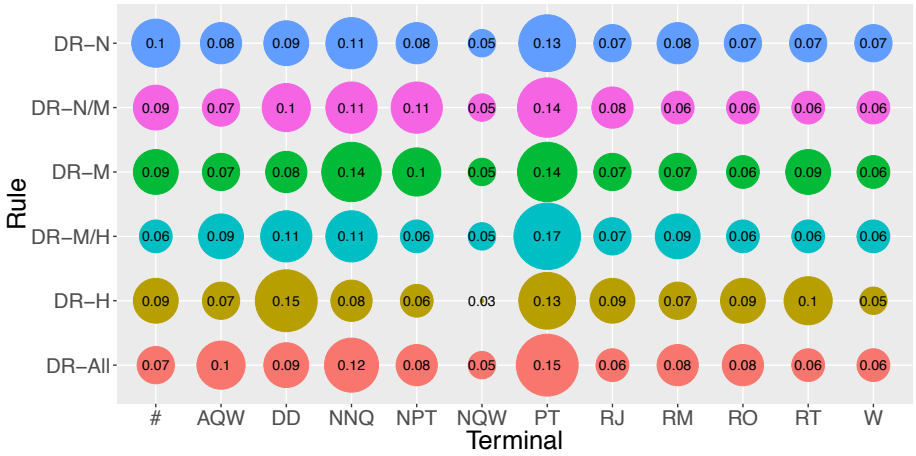


Fig. 3: The proportion of terminals used by the sets of rules evolved by the GP-HH approaches.

the number of jobs waiting at the next machine (NNQ). On the other hand, DR-H have a higher proportion of due date terminal compared to the other sets of evolved rules. This is likely due to the fact that in problem instances with high machine breakdown level (e.g. BL $= 0.05$) processing time becomes unreliable in determining on predicting the expected duration of time that a job requires on the machine for the operation to complete. Compared to problem instances with lower machine breakdown levels (e.g. BL $= 0$ and BL $= 0.025$), it is more likely in the problem instances with high machine breakdown levels that the machine breaks down during processing of a job. This results in the job getting stuck on the machine while it is being repaired and taking longer than expected to finish processing. Instead, processing urgent jobs may be more reliable method of generating good schedules for problem instances with high level of machine breakdowns. Therefore, this may potentially result in individuals that use high proportion of the processing time terminal, that prioritise processing shorter job processing times, generating worse schedules for the problem instances than individuals that use high proportion of the due date terminal, that prioritise processing urgent jobs. In addition, DR-H has a lower proportion of terminals that take the attributes for when the job reaches the next machine (NNQ, NPT and NQW), as the additional uncertainty introduced by the high level of machine breakdown may make the terminals less effective at reducing the myopic nature of dispatching rules [8].

6 Conclusions

This paper investigates the generality of a standard GP-HH approach to a DJSS problem subject to dynamic job arrivals and machine breakdowns. This is done by first developing a DJSS dataset for evaluating GP-HH approaches. Afterwards, a standard GP-HH approach that evolves priority-based dispatching rules is applied to a new dataset for generating DJSS problem instances. Finally, the distributions of the terminals in the GP rules evolved from different machine breakdown scenarios are analysed. From the

performance results and the results of analysing the terminal distributions, this paper makes the following findings:

(a) GP-HH approaches in the literature have been shown to be generalise well over different problem domains (including JSS) [6, 7]. However, for the DJSS problem with dynamic job arrivals and machine breakdowns, the results show that a standard GP-HH approach is sensitive to the level of machine breakdowns. In addition, a generalist set of rules evolved by a standard GP-HH approach is unable to cover for high level of machine breakdowns.
(b) Analysis of the distribution of the terminals for the evolved rules show that there are higher proportions of DD terminal and lower proportion of NNQ, NPT, NQW and PT terminals in rules evolved on training instances with high levels of machine breakdowns compared to the other evolved rules. This is likely due to the added uncertainty associated with the duration of time required to process a job.

For future work, a GP-HH approach that incorporates terminals that use machine breakdown specific attributes could potentially evolve rules that can outperform rules evolved by a standard GP-HH approach and improve the generality of GP rules over different machine breakdown scenarios. For example, it is likely that incorporating terminals such as the next time the machine is expected to break down and the expected true processing time may generate rules which perform well over both DJSS problem instances with and without machine breakdowns.

References

1. Pinedo, M.L.: Scheduling: Theory, Algorithms, and Systems. 4 edn. Springer (2012)
2. Ouelhadj, D., Petrovic, S.: A survey of dynamic scheduling in manufacturing systems. Journal of Scheduling **12**(4) (2009) 417–431
3. Hildebrandt, T., Branke, J.: On using surrogates with genetic programming. **23**(3) (2015) 1–25
4. Nguyen, S., Zhang, M., Johnston, M., Tan, K.C.: A computational study of representations in genetic programming to evolve dispatching rules for the job shop scheduling problem. IEEE Transactions on Evolutionary Computation **17**(5) (2013) 621–639
5. Hildebrandt, T., Heger, J., Scholz-Reiter, B.: Towards improved dispatching rules for complex shop floor scenarios: A genetic programming approach. In: Proceedings of the 12th Annual Conference on Genetic and Evolutionary Computation. (2010) 257–264
6. Burke, E.K., Hyde, M., Kendall, G., Woodward, J.: A genetic programming hyper-heuristic approach for evolving 2-d strip packing heuristics. IEEE Transactions on Evolutionary Computation **14**(6) (2010) 942–958
7. Branke, J., Nguyen, S., Pickardt, C.W., Zhang, M.: Automated design of production scheduling heuristics: A review. IEEE Transactions on Evolutionary Computation **20**(1) (2016) 110–124
8. Hunt, R., Johnston, M., Zhang, M.: Evolving "less-myopic" scheduling rules for dynamic job shop scheduling with genetic programming. In: Proceedings of the 2014 Conference on Genetic and Evolutionary Computation. (2014) 927–934
9. Holthaus, O.: Scheduling in job shops with machine breakdowns: an experimental study. Computers & Industrial Engineering **36**(1) (1999) 137–162
10. Vepsalainen, A.P.J., Morton, T.E.: Priority rules for job shops with weighted tardiness costs. Management Science **33**(8) (1987) 1035–1047