Investigating a Machine Breakdown Genetic Programming Approach for Dynamic Job Shop Scheduling

John Park[1(✉)], Yi Mei[1], Su Nguyen[1,2], Gang Chen[1], and Mengjie Zhang[1(✉)]

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

Abstract. Dynamic job shop scheduling (JSS) problems with dynamic job arrivals have been studied extensively in the literature due to their applicability to real-world manufacturing systems, such as semiconductor manufacturing. In a dynamic JSS problem with dynamic job arrivals, jobs arrive on the shop floor unannounced that need to be processed by the machines on the shop floor. A job has a sequence of operations that can only processed on specific machines, and machines can only process one job at a time. Many effective genetic programming based hyper-heuristic (GP-HH) approaches have been proposed for dynamic JSS problems with dynamic job arrivals, where high quality dispatching rules are automatically evolved by GP to handle the dynamic JSS problem instances. However, research that focus on handling multiple dynamic events simultaneously are limited, such as both dynamic job arrivals and machine breakdowns. A machine breakdown event results in the affected machine being unable to process any jobs during the repair time. It is likely that machine breakdowns can significantly affect the effectiveness of the scheduling procedure unless they are explicitly accounted for. Therefore, this paper develops new machine breakdown terminals for a GP approach and evaluates their effectiveness for a dynamic JSS problem with both dynamic job arrivals and machine breakdowns. The results show that the GP approaches with the machine breakdown terminals do show improvements. The analysis shows that the machine breakdown terminals may indirectly contribute in the evolution of high quality rules, but occur infrequently in the output rules evolved by the machine breakdown GP approaches.

1 Introduction

Job shop scheduling (JSS) problems are combinatorial optimisation problems that have been studied over the past 60 years [1]. Due to their direct application in important real-world manufacturing systems, extensive research has been carried out for JSS problems to find effective and practical techniques which may be incorporated to a real-world scenario for the manufacturers so that they gain a

© Springer International Publishing AG, part of Springer Nature 2018
M. Castelli et al. (Eds.): EuroGP 2018, LNCS 10781, pp. 253–270, 2018.
https://doi.org/10.1007/978-3-319-77553-1_16

competitive edge in the respective markets [1]. In a JSS problem instance, there are *machines* on the *shop floor* that are used to process arriving *jobs*, and the manufacturer needs to make intelligent decisions to process the jobs as effectively as possible. In other words, machine resources need to be optimally allocated (given a specific criterion) by determining the *sequence* in which the jobs are processed. However, optimal allocation of machines can be a difficult task. Most job shop scheduling (JSS) problems are NP-hard [1], and mathematical optimisation techniques that return optimal solutions for problem instances do not scale effectively with the problem size. In addition, in a dynamic JSS problem instance there are unforeseen events that affect the properties of the shop floor, e.g., dynamic job arrivals and machine breakdowns [2]. To handle dynamic JSS problems, various heuristic approaches have been proposed to generate good solutions to problem instances while coping with the unforeseen events. For this paper, we handle dynamic JSS problems with dynamic job arrivals, where the jobs' properties and their arrival times are unknown until the job arrival times are reached during processing [3]. Dispatching rule approaches are the most prominent method of handling dynamic JSS problems with dynamic job arrivals due to their short reaction times and their ability to cope with the dynamic environment [4].

In addition to manually designing effective dispatching rules for dynamic JSS problems with dynamic job arrivals, researchers have proposed various genetic programming based hyper-heuristic (GP-HH) approaches to automatically evolving dispatching rule from heuristic subcomponents [5]. GP evolved dispatching rules generally perform better than man-made dispatching rules for JSS problems [6]. However, GP approaches that have been proposed for dynamic JSS problems have mainly focused on dynamic job arrivals [3–7]. In a real-world scenario, it is likely that there are different types of dynamic events that occur during processing. An example is machine breakdown, where the machines need to be serviced and repaired [2]. It is likely that disruptions caused by machine breakdowns can likely impact the performance of the scheduling algorithm if they are not specifically accounted for. The only GP approach that explicitly accounts for machine breakdowns in the literature deals with a single machine JSS problem with no dynamic job arrivals [8]. Developing machine breakdown specific GP approaches may allow us to improve the overall quality of rules evolved by GP for dynamic JSS problems with dynamic job arrivals and machine breakdowns (DJSS-MB).

1.1 Goal

The goal of this paper is to develop new machine breakdown terminals for a GP terminal set commonly used in the literature [4,7] to handle a DJSS-MB. By incorporating machine breakdown terminals into the GP terminal set, it may be possible to evolve rules that can account for machine breakdown information. This may result in the evolved rules being able to make better decisions for both machine breakdown and non-machine breakdown JSS problem instances than rules evolved without machine breakdown information, and generate better

solutions overall. In other words, developing new machine breakdown terminals may allow GP to consistently evolve high quality rules for DJSS-MB. Afterwards, by analysing specific machine breakdown GP evolved rules, it may be possible to develop an insight into how the rules behave in DJSS-MB, allowing us to potentially develop more effective machine breakdown GP approaches in the future. Overall, this paper carries out the following objectives:

(a) Develop and evaluate new machine breakdown terminals for an existing GP approach [4,7].
(b) Carry out a structural analysis of the machine breakdown GP rules to gain an understanding of the useful features and properties of GP rules that are evolved under machine breakdown.

1.2 Organisation

First, we cover the background to dynamic JSS in Sect. 2, which includes the problem definition and outlines existing GP approaches for dynamic JSS problems. Afterwards, Sect. 3 describes the existing GP approach used in the literature [4,7], the benchmark GP terminals, and the machine breakdown GP terminals investigated in this paper. Section 4 describes the dynamic JSS simulation model used in this paper, and the GP parameters. Finally, Sect. 5 gives the results and an analysis of the findings, and Sect. 6 gives the concluding remarks and the future works.

2 Background and Related Work

This section covers the problem definition for the DJSS-MB, and the GP approaches for dynamic JSS problems in the literature.

2.1 Problem Definition

In a dynamic JSS problem instance, an arriving job j has a sequence of operations $\sigma_{1j}, \ldots, \sigma_{N_j j}$. The operations must be processed sequentially (e.g. σ_{1j} must be processed before σ_{2j}) and need to be processed on specific machines $m(\sigma_{1j}), \ldots, m(\sigma_{N_j j})$. An operation σ_{ij} needs to be processed on machine $m(\sigma_{ij})$ for time p_{ij} (which is called the *processing time*), and a machine can only process one job at a time. The time that a job arrives at the machine for an operation σ_{ij} is denoted as r_{ij}, and the time that the job arrives on the shop floor (r_{1j}) is called the *job arrival time* r_j. For this paper, the objective is to minimise total weighted tardiness (MWT). MWT objective has been studied extensively in the literature [1], and tardiness related objectives have been shown to be quite sensitive to machine breakdown events [9,10]. In a MWT objective, an arriving job has a due date d_j and a weight w_j. If job j's completion time C_j (the time when the last operation of job j completes) is below due date d_j, i.e., $C_j \leq d_j$, then no penalty is incurred. Otherwise, job j is considered *tardy*, and has tardiness

$T_j = d_j - C_j$ [11]. After all N arriving jobs are completed, the MWT of the schedule is given by $\frac{1}{N}\sum_{j=1}^{N} w_j T_j$.

The two types of dynamic events that occur during processing are dynamic job arrivals and machine breakdowns. With a dynamic job arrival, the information about a job j, including its properties, are not known in advance until the job arrives at time r_j. With a machine breakdown event, a machine m breaks down at time b^m and the machine needs to be repaired for time r^m. During the repair time, the machine is unavailable to process any operations. If a job's operation is currently being processed at the machine at the time of breakdown, then the job's operation is resumed from the time it was interrupted after the machine is repaired. This means that if a job j's operation σ_{ij} is started at time s_t and is interrupted by machine m's breakdown, then the operation completes at time $s_t + p_{ij} + r^m$. A job operation resuming from the point of interruption is consistent with the machine breakdown definition in the literature [8,9] to handle dynamic JSS problems with machine breakdown. In addition, a common assumption in the literature is that machine breakdowns events are unforeseeable [2]. However, for this paper we simplify the problem by allowing the shop floor to know when the machine breakdown occurs in advance. This is because GP approaches that handle both dynamic job arrivals and machine breakdowns have not yet been proposed in the literature. By doing this, we can carry out a preliminary investigation of machine breakdown terminals that incorporate informations about future machine breakdowns (described in Sect. 3.2). After analysing these terminals that take full information about future machine breakdowns into account, it may be possible to develop machine breakdown terminals in the future that can cope effectively even when the machine breakdown events are unforeseen events.

2.2 GP for Dynamic JSS Problems

GP approaches have been extensively applied to dynamic JSS problems to evolve dispatching rules [5,6]. Many GP approaches use single arithmetic function trees as priority dispatching rules [5,6]. Geiger et al. [12] showed that GP can evolve optimal priority dispatching rules for static JSS problems that are not NP-hard. They also showed that priority dispatching rules evolved for NP-hard static JSS problems and dynamic JSS problems perform better than benchmark man-made dispatching rules. Hildebrandt et al. [3] provided a GP approach for a dynamic JSS problem with the flowtime minimisation objective, and showed that the GP evolved rules outperform state-of-the-art man-made dispatching rules. Branke et al. [5] and Nguyen et al. [6] both carry out extensive survey of GP-HH approaches to scheduling problems in the literature. Nguyen et al. [13] also provide a unified framework for GP-HH to scheduling problems and categorises existing GP approaches using the framework.

The following are GP approaches that evolve scheduling rules for dynamic JSS problems with machine breakdowns. Yin et al. [8] proposed a two-tree GP approach for a single machine JSS problem. The first tree acts as a dispatching rule, and the second tree is used to calculate the idle time to add in between

processing different jobs on a machine. They showed that the evolved rules outperform the benchmark man-made heuristics designed for JSS problems with machine breakdowns. Park et al. [10] carried out an investigation on the generality of GP over a JSS problem with different frequencies and durations of machine breakdowns, and found that the proportion of time the machines are being repaired is a significant factor in the qualities of the evolved rules. In addition, they showed that GP is not general enough to cover for all different scenarios effectively, and that it is likely that machine breakdown specific information is required to improve the generality of GP.

3 Machine Breakdown GP for the Dynamic JSS Problem

As machine breakdown GP approach for DJSS-MB have not been proposed, this paper proposes simple but novel machine breakdown terminals which are incorporated to a GP approach. This allows GP to evolve dispatching rules that may make better decisions during decision situations, potentially leading to better performance than GP evolved rules which do not incorporate machine breakdowns. First, we describe the GP representation, the benchmark terminal and function sets. This is followed by the descriptions and justifications for the machine breakdown GP terminals. The first approach replaces existing terminals related to operation processing times and add repair time of machines if necessary. This approach is denoted as "augmented" approach, as it attempts to *improve* certain benchmark terminals by incorporating machine breakdown information. The second approach adds *new* machine breakdown terminals, which "react" to the machine breakdowns happening on the shop floor, to the existing set of GP terminals.

3.1 GP Representation, Terminal Set and Function Set

For this paper, we use a tree-based GP representation [14]. The GP individuals represent arithmetic function trees that calculate the priorities of jobs during decision situations. Arithmetic GP representation has been used prominently in the literature to evolve effective priority dispatching rules for JSS problems [5]. A GP terminal corresponds to a job, machine and shop floor attribute value at a decision situation, or combines multiple base level shop floor attributes as a part of the terminal. For example, the RT terminal returns the sum remaining total processing times of job j waiting at a machine to process operation σ_{ij}, i.e., $\mathrm{RT}(j) = \sum_{k=i}^{N_j} p_{kj}$. The GP terminals and the arithmetic operators used by the benchmark GP approach are listed in Table 1, which is based off existing terminal and function sets used by GP approaches to dynamic JSS problems in the literature [7,10]. The function set consists of the arithmetic operators $+$, $-$, $\times$, protected /, binary operators max, min and a ternary operator `if`. The protected / returns one if the denominator is zero, and carries out the standard division operator otherwise. max and min returns the maximum and the minimum value of the two arguments respectively. `if` returns the value of

the second argument if the value of the first argument is greater than or equal to zero, or returns the value of the third argument otherwise.

Table 1. Terminal set for GP, where a job j is waiting at the available machine m at a decision situation.

Terminal	Description
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
WINQ	Work in next queue for job j
DD	Job's due date d_j
SL	Slack of job j
W	Job's weight w_j
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

A GP individual ω is evaluated over a set of dynamic JSS training instances to calculate its fitness as follows. GP individual ω is applied to a JSS problem instance γ as a *non-delay* dispatching rule [11]. During a simulation when a machine m becomes available, the jobs that are currently waiting at machine m are assigned priority values by the dispatching rule. The job that has the highest priority value is selected to be processed at machine m. This continues until the termination criteria for the simulation has been reached (e.g. after a certain number of jobs have been completed), at which point the objective function value $Obj(\omega, \gamma)$ is calculated from the solution generated by individual ω. The individual ω is applied to all problem instances in the training sets, obtaining objective values $Obj(\omega, \gamma_1), \ldots, Obj(\omega, \gamma_{T_{train}})$.

After the GP individual ω is applied to the problem instances in the training set, the objective values obtained are normalised to reduce the likelihood that the GP individuals are biased towards specific instances [3,15]. In other words, an objective value $Obj(\omega, \gamma)$ for a solution generated by individual ω for instance γ is normalised using *reference* objective value $Obj(R, \gamma)$ as shown in Eq. (1). The reference objective value $Obj(R, \gamma)$ is calculated from the solution generated by applying a *reference rule* R to the problem instance γ. The reference rule used is the weighted apparent tardiness cost (wATC) rule [16], a man-made dispatching rule effective for weighted tardiness related objectives. This was also used by Park et al. [10] in the fitness function to evolve GP rules for the DJSS-MB.

$$Obj'(\omega, \gamma) = \frac{Obj(\omega, \gamma)}{Obj(R, \gamma)} \tag{1}$$

3.2 Augmented GP Terminals

The following terminals in the original GP terminal set (as described in Table 1) are replaced by terminals that add repair times of the machines: job's *operation processing time* (PT), job's *next operation processing time* (NPT) and *work in next queue* (WINQ). The replaced GP terminals return the original value if the job's operation is not interrupted by a machine breakdown, and adds the repair time of the machine otherwise. The terminals that incorporate the machine breakdown information is denoted with the prefix 'MB-' (e.g. MBPT for machine breakdown adjusted processing time). The GP approach that incorporates the MBPT, MBNPT and MBWINQ terminals is denoted as GP-Aug.

Machine breakdown adjusted processing time (MBPT): The machine breakdown adjusted processing time terminal (MBPT) replaces the processing time terminal (PT) in Table 1. Given that the current time during the decision situation is t and the processing time of job's current operation j is p_{ij}, MBPT terminal returns the actual duration of time required to process the job's operation by factoring the machine breakdown interruption into account. In other words, if the job is *not* interrupted by a machine breakdown, i.e., if the operation completes earlier than the breakdown time b_t^m of the current machine m, then the job's actual processing time p'_{ij} is equal to the expected processing time p_{ij}. Otherwise, the actual processing time is the sum of the processing time and the machine repair time r_t^m required to get the machine back up and running before the operation is resumed. The value returned by MBPT$(j) = p'_{ij}$, where the calculation for p'_{ij} is shown in Eq. (2).

$$p'_{ij} = \begin{cases} p_{ij} & \text{if } t + p_{ij} < b_t^m \\ p_{ij} + r_t^m & \text{otherwise} \end{cases} \tag{2}$$

Machine breakdown adjusted next processing time (MBNPT): The machine breakdown adjusted next processing time terminal (MBNPT) replaces the next processing time terminal (NPT) in Table 1. MBNPT terminal returns zero if the job j's current operation σ_{ij} is the last operation before job j's completion. Otherwise, given that the next operation $\sigma_{(i+1)j}$ is processed on machine m', the repair time of m' is added to the next processing time $p_{(i+1)j}$ if it is expected to be interrupted by a breakdown at machine m' at operation $\sigma_{(i+1)j}$ earliest possible completion at machine m'. The earliest possible time that job j can be completed is if operation σ_{ij} is selected immediately by machine m, and then the successive operation $\sigma_{(i+1)j}$ is then processed by machine m' as soon as operation σ_{ij} is completed. The time when operation σ_{ij} completes is given by the current time t and the actual processing time p'_{ij}, which depends on whether the operation is interrupted by machine breakdown (Eq. (2)). In other words, the earliest time operation $\sigma_{(i+1)j}$ can be processed at machine m'

is at $t + p'_{ij}$ after operation σ_{ij} is expected to complete. Therefore, if machine m' breaks down before time $t + p'_{ij} + p_{(i+1)j}$, then repair time $r_t^{m'}$ of machine m' is added to the operation $\sigma_{(i+1)j}$'s processing time $p_{(i+1)j}$ as shown in Eq. (3).

$$\text{MBNPT}(j) = \begin{cases} p_{(i+1)j} & \text{if } t + p'_{ij} + p_{(i+1)j} < b_t^{m'} \\ p_{(i+1)j} + r_t^{m'} & \text{otherwise} \end{cases} \tag{3}$$

Machine breakdown adjusted work in next queue (MBWINQ): The machine breakdown adjusted work in next queue terminal (MBWINQ) replaces the work in next queue terminal (WINQ) in Table 1. Both WINQ and MBWINQ terminals return zero if the job j's current operation σ_{ij} is the last operation before job j's completion. Otherwise, given that machine m' is required by operation $\sigma_{(i+1)j}$, the standard WINQ terminal returns the sum processing times of the jobs that are currently waiting at machine m' plus the remaining time required to process the operation currently being processed by machine m', i.e., the work remaining. MBWINQ modifies the work remaining time calculated by WINQ, and adds the machine m''s repair time if the work is interrupted by machine breakdown at time $b_t^{m'}$. In other words, $\text{MBWINQ}(j) = wr'_{m',j'} + \sum_{i=1}^{N_{m'}} p_{j_i}$, where $p_{j_1}, \ldots, p_{j_{N_{m'}}}$ are the processing times of jobs waiting at machine m', $wr'_{m'j'}$ is the actual work remaining required on j' being processed on machine m' before it becomes available. The calculation for actual work remaining $wr'_{m'j'}$ is given in Eq. (4), where $s_{j'}$ denotes the time when j' started, $p_{j'}$ is the processing time required by j' at machine m' and t is the current time.

$$wr'_{m'j'} = \begin{cases} s_{j'} + p_{j'} - t & \text{if } s_{j'} + p_{j'} < b_t^{m'} \\ s_{j'} + p_{j'} - t + r_t^{m'} & \text{otherwise} \end{cases} \tag{4}$$

In summary, the augmented GP approach replaces three existing terminals (PT, NPT and WINQ) with equivalent terminals that incorporate information about future machine breakdowns. The existing terminals are related to the processing times of the jobs waiting on the shop floor, where repair times need to be added onto the processing times if we expect the jobs to be interrupted by machine breakdowns. By doing this, we expect the GP rule to be able to use the "actual" processing times of the jobs to make better decisions on what job should be processed next by a machines during decision situations.

3.3 Reactive GP Terminals

Reactive machine breakdown terminals are added to the GP terminal set described in Table 1 and incorporate information about current machine status. As the two terminals incorporate informations about the potential wait time of a job waiting at a machine for the next machine it visits, they are investigated separately. The two terminals being investigated are the *repair time remaining next machine terminal* (RTR) and the *minimum wait time next machine terminal* (WT). The two reactive GP terminals may allow rules to make better

decisions by prioritising jobs with low expected wait time compared to jobs with high expected wait time. This may lead to jobs spending less time waiting at busy machines, and the evolved rules may generate higher quality schedules. The GP approach that incorporates the RTR terminal is denoted as GP-RTR, and the GP approach that incorporates the WT terminal is denoted as GP-WT.

Repair Time Remaining Next Machine (RTR): The repair time remaining next machine RTR returns zero if a job j waiting at a machine at time t is currently on its last operation or the next machine m' visited by j is currently not broken down. Otherwise, given that machine m' broke down at time $b_t^{m'}$ and the repair time is $r_t^{m'}$, the value given by RTR $= b_t^{m'} + r_t^{m'} - t$.

Minimum Wait Time Next Machine (WT): The minimum wait time next machine WT returns the earliest time that the machine to be visited by job j next becomes available. If the current operation of j is the last operation before completion, then WT returns zero. In addition, if the next machine m' that job j visits is currently not busy and is not broken down, i.e., is completely available, then WT returns zero. Otherwise, the WT returns the duration of time required for machine m' to be available. If machine m' is currently processing a job j' or is broken down with an interrupted job, then it returns the actual work remaining $wr'_{m'j'}$ which is given in Eq. (4). Otherwise, if the m' is broken down and a job was not interrupted by the machine breakdown, WT returns the remaining repair time of machine m' as given by the terminal RTR.

4 Experimental Design

This section describes the setup used to evaluate the different GP approaches to tackle the DJSS-MB. To evaluate the machine breakdown GP approaches, a simulation model that is slightly modified from existing simulation models in the literature [9,10] is used to both evolve and evaluate the evolved rules. Afterwards, we provide the parameter used by the GP approaches.

4.1 Dynamic Simulation Model with Machine Breakdown

Discrete-event simulations are the most prominent method of generating dynamic JSS problem instances [5,6]. In a discrete-event simulation, the dynamic events such as the job arrivals and the machine breakdowns are stochastically generated from a set of parameters. A simulation configuration is the set of parameters required, along with a seed value, to generate a dynamic JSS problem instance. In a dynamic JSS problem instance, there are $M = 10$ machines on the shop floor. The problem instance has a "warm-up" period of 500 jobs, where the first 500 jobs completed do not contribute towards the objective. The simulation is terminated after 2500 jobs are completed, and the objective function value is calculated from the 2000 jobs completed after the warm-up phase. The job arrivals times follow a Poisson process with arrival rate $\lambda = \rho \times \mu \times p_M$. In the equation, ρ is the utilisation rate, μ the mean processing time, and p_M

the mean number of job operations to machine ratio. Utilisation rate (ρ) is the expected proportion of time the machines are spent processing job operations, and $\rho = 80\%$ for all problem instances. If the utilisation rate plus the machine breakdown level is too high, it is very likely that the shop will be *unstable* [11], i.e., job arrival rate is greater than the rate at which the shop floor can process them. Therefore, 80% utilisation rate is used instead of higher utilisation rates used in the literature (e.g. 90% or 95% [4,9]) to accommodate for the high level of machine breakdown used by the simulation model (described below). The mean processing time (μ) is used in a uniform distribution with the interval $[1, 2\mu - 1]$ that the jobs' processing times follow, and $\mu = 25$ for all problem instances. The mean number of job operations to machine ratio (p_M) is the expected number of machines that a job will visit divided by the total number of machines. The number of operations a job has follows a uniform distribution in the interval $[2, 10]$, i.e., the minimum and the maximum number of operations that a job can have is 2 and 10 respectively. Therefore, the expected number of operations is 6 for a job arriving on the shop floor and $p_M = 0.6$ for all problem instances. In addition, a job's weight has the value of 1, 2 or 4 with 20%, 60% or 20% probabilities respectively. Given a job j's arrival time r_j, total processing time $\sum_{i=1}^{N_j} p_{ij}$ and the due date tightness simulation parameter h, the due date of job j is $d_j = r_j + h \times \sum_{i=1}^{N_j} p_{ij}$.

For generating machine breakdown events, the inter-breakdown times of the machines follow an exponential distribution, and the expected breakdown rate is given by $\eta = r_t/\pi - r_t$. In the equation, π is the breakdown level and r_t is the machine repair time. The breakdown level is the expected proportion of time for the machine to be broken down over the course of the simulation, and varies between the different simulation configurations used to generate the problem instances. For a problem instance the machine repair time is the same across all machines for a problem instance.

The dataset parameters for generating job arrivals and machine breakdowns are shown in Table 2. First, the simulation configurations have the possible breakdown levels $\pi = 0\%$, 2.5%, 5%, 10%, or 15% for a simulation configuration. Second, fixed repair times for the machine breakdowns are either $r_t = 37.5$, 137.5, or 262.5 for a simulation configuration. These parameters were selected after running the benchmark GP approach on different breakdown levels and durations of repair times as part of a preliminary experiment. The simulation configuration consists of a combination of the dataset parameters, which means that there are $3 \times 5 \times 2 = 30$ different simulation configurations available in the dataset. We use the simulation configuration with $\pi = 15\%$, $r_t = 262.5$ and $h = 3$ to generate the training problem instances. In addition, a different seed is used with the training simulation configuration every generation during the GP process, resulting in different dynamic JSS problem instances being used every generation.

Table 2. Dynamic JSS parameter settings

Simulation model parameter	Value
Number of machines (M)	10
Utilisation rate (ρ)	80%
Mean processing time (μ)	25
Weight/probability ((w, p))	$\{(1, 20\%), (2, 60\%), (4, 20\%)\}$
Due date tightness (h)	3 or 5
Machine breakdown level (π)	0%, 2.5%, 5%, 10% or 15%
Repair time (r_t)	37.5, 137.5 or 262.5

4.2 GP Parameters

The GP parameters are used by the GP approaches are shown in Table 3. The GP parameters are the same as the parameters that are same as the ones used by Park et al. [10] in their investigation into GP approaches for a DJSS-MB, which allows our benchmark GP approach to be consistent with the GP approach that was used during their investigation.

Table 3. GP parameter settings

GP Parameter	Value
Population size	1024
Number of generations	51
Crossover rate	80%
Mutation rate	10%
Reproduction rate	10%
Max initial depth	2
Max depth	8
Initialisation method	Ramped half-and-half
Selection method	Tournament selection
Selection size	2

5 Experimental Results

In this investigation, we first carry out a performance evaluation of the GP approaches. The performance evaluation first compares the GP approaches and how consistently they can evolve high quality dispatching rules for the dynamic JSS problem, i.e., measures the effectiveness of the GP approaches. This is done by evolving a set of rules for each approach and applying them over the dynamic JSS simulation model. Afterwards, the best rules are extracted from the sets of

evolved rules for the GP approaches and compared individually to determine whether an individual machine breakdown rule can outperform an evolved rule generated by the benchmark GP approach. Finally, we carry out a structural analysis of the best rules evolved by the machine breakdown GP approaches to find out the useful properties from the evolve rules.

5.1 Performance Evaluation

For the performance evaluation, each GP approach is applied to a training set (described in Sect. 4.1) thirty times to evolve thirty independent rules. Afterwards, each of the rule is applied to the dynamic JSS simulation model as follows.

Performance Measure: First, an evolved rule ω is run multiple times over each simulation configuration in the simulation model. A single run consists of a seed value and a simulation configuration, which are used to generate a dynamic JSS problem instance. The rule is then applied to the problem instance and generates a schedule, which has a MWT objective value. Afterwards, the subsequent runs over the simulation configuration use unique seeds so that new problem instances are generated from the same simulation configuration. In other words, given a simulation configuration sim and rule ω, schedules with MWT values $Obj(\omega, \gamma_{(sim)1})$, ..., $Obj(\omega, \gamma_{(sim)30})$ are generated by the rule for the given simulation configuration. These are used slightly differently for the rule set evaluation and best rule evaluation, which are described below.

Rule Set Results: In the rule set evaluation, the MWT values $Obj(\omega, \gamma_{(sim)1})$, ..., $Obj(\omega, \gamma_{(sim)30})$ generated by a rule ω for a simulation configuration sim is averaged out to obtain the "performance" $Perf$ of the rule over the simulation configuration, i.e., $Perf(\omega) = \frac{1}{30}\sum_{i=1}^{30} Obj(\omega, \gamma_{(sim)i})$. The rule performances are then used to compare between the different sets of rules evolved by the GP approaches.

The results of the performance evaluation is shown in Table 4. In the table, $\langle \pi, r_t, h \rangle$ denotes that the simulation configuration has the respective breakdown level π, repair time r_t, and due date tightness h. In addition, each entry $\mu \pm \sigma$ is the mean (μ) and standard deviation (σ) of the performance $Perf$ of the rules for the simulation configuration respectively. If a set of GP evolved rules that use the machine breakdown terminals is *significantly* better than the set of benchmark GP rules for a simulation configuration by satisfying the two sided Student's t-test at $p = 0.05$, then the particular entry is highlighted.

Although the differences are not significant, the results show that the three machine breakdown approaches (GP-Aug, GP-WT and GP-RTR) have slightly better performances than the benchmark GP for some simulation configurations. In particular, the GP-WT rules have slightly better performances for all simulation configurations than the benchmark GP rules. In addition, the GP-RTR rules

Table 4. Comparison of the rule sets evolved by the GP approaches over the simulation configurations. Rules are evolved from $\langle 0.15, 262.5, 3\rangle$.

Model Subset		MB			GP
		GP-Aug	GP-WT	GP-RTR	
MWT ($\times 10^2$)	$\langle 0, 37.5, 5\rangle$	0.74 ± 0.17	0.66 ± 0.07	0.67 ± 0.13	0.67 ± 0.16
	$\langle 0, 37.5, 3\rangle$	1.12 ± 0.16	1.05 ± 0.07	1.06 ± 0.12	1.07 ± 0.15
	$\langle 0, 137.5, 5\rangle$	0.60 ± 0.16	0.53 ± 0.06	0.53 ± 0.11	0.53 ± 0.14
	$\langle 0, 137.5, 3\rangle$	1.31 ± 0.18	1.24 ± 0.07	1.24 ± 0.12	1.26 ± 0.16
	$\langle 0, 262.5, 5\rangle$	0.74 ± 0.17	0.66 ± 0.07	0.66 ± 0.13	0.66 ± 0.16
	$\langle 0, 262.5, 3\rangle$	1.36 ± 0.19	1.28 ± 0.08	1.29 ± 0.13	1.30 ± 0.16
	$\langle 0.025, 37.5, 5\rangle$	1.60 ± 0.89	1.54 ± 0.52	1.55 ± 0.75	1.59 ± 0.69
	$\langle 0.025, 37.5, 3\rangle$	2.92 ± 0.98	2.78 ± 0.52	2.86 ± 0.76	2.95 ± 0.88
	$\langle 0.025, 137.5, 5\rangle$	38.39 ± 11.90	36.07 ± 11.10	38.91 ± 10.38	39.20 ± 12.65
	$\langle 0.025, 137.5, 3\rangle$	42.53 ± 12.23	40.49 ± 11.10	42.96 ± 10.87	43.23 ± 12.89
	$\langle 0.025, 262.5, 5\rangle$	88.51 ± 25.15	89.94 ± 23.00	92.91 ± 23.63	94.43 ± 27.18
	$\langle 0.025, 262.5, 3\rangle$	92.11 ± 24.34	93.81 ± 21.81	96.36 ± 22.91	98.17 ± 26.54
	$\langle 0.05, 37.5, 5\rangle$	1.64 ± 0.44	1.53 ± 0.20	1.57 ± 0.35	1.58 ± 0.36
	$\langle 0.05, 37.5, 3\rangle$	2.76 ± 0.57	2.68 ± 0.30	2.74 ± 0.52	2.78 ± 0.50
	$\langle 0.05, 137.5, 5\rangle$	9.04 ± 2.17	8.29 ± 2.03	8.91 ± 1.89	9.05 ± 2.42
	$\langle 0.05, 137.5, 3\rangle$	11.14 ± 2.18	10.65 ± 1.87	11.04 ± 1.97	11.28 ± 2.29
	$\langle 0.05, 262.5, 5\rangle$	36.43 ± 11.35	34.32 ± 10.75	37.00 ± 10.13	37.38 ± 12.40
	$\langle 0.05, 262.5, 3\rangle$	36.56 ± 11.29	34.33 ± 10.20	36.74 ± 9.61	37.27 ± 12.09
	$\langle 0.1, 37.5, 5\rangle$	3.69 ± 0.61	3.53 ± 0.27	3.60 ± 0.50	3.63 ± 0.60
	$\langle 0.1, 37.5, 3\rangle$	4.60 ± 0.54	4.51 ± 0.26	4.57 ± 0.44	4.63 ± 0.51
	$\langle 0.1, 137.5, 5\rangle$	6.11 ± 1.21	5.79 ± 0.35	5.89 ± 0.81	6.07 ± 1.26
	$\langle 0.1, 137.5, 3\rangle$	8.15 ± 1.28	7.91 ± 0.44	8.02 ± 0.86	8.29 ± 1.39
	$\langle 0.1, 262.5, 5\rangle$	11.72 ± 1.95	11.07 ± 1.56	11.43 ± 1.56	11.74 ± 2.22
	$\langle 0.1, 262.5, 3\rangle$	13.14 ± 1.50	12.61 ± 1.52	13.08 ± 1.21	13.29 ± 1.76
	$\langle 0.15, 37.5, 5\rangle$	6.20 ± 0.67	5.89 ± 0.21	6.07 ± 0.50	6.06 ± 0.80
	$\langle 0.15, 37.5, 3\rangle$	7.73 ± 0.63	7.52 ± 0.18	7.66 ± 0.50	7.74 ± 0.82
	$\langle 0.15, 137.5, 5\rangle$	9.02 ± 0.94	8.53 ± 0.59	8.69 ± 0.64	8.81 ± 1.17
	$\langle 0.15, 137.5, 3\rangle$	11.73 ± 0.81	11.38 ± 0.42	11.55 ± 0.58	11.71 ± 1.10
	$\langle 0.15, 262.5, 5\rangle$	12.55 ± 1.23	12.01 ± 1.29	12.31 ± 0.95	12.48 ± 1.44
	$\langle 0.15, 262.5, 3\rangle$	16.60 ± 1.27	16.08 ± 1.33	16.37 ± 0.97	16.59 ± 1.50

have slightly better performance than the benchmark GP rules for most simulation configurations except configurations $\langle 0, 37.5, 5\rangle$ and $\langle 0.15, 37.5, 5\rangle$. Finally, the results of the comparison between the GP-Aug rules and the benchmark GP rules is most mixed, where GP-Aug rules are slightly better or worse than the benchmark rules on roughly the equal number of simulation configurations. However, due to the lack of statistical significance, no significant conclusions can be drawn on whether the machine breakdown GP approaches is more consistent in evolving higher quality dispatching rules than the benchmark GP approach. However, by analysing the rules further, it may be possible to gain a better understanding of how GP can be applied effectively to the machine breakdown problem.

Best Rule Results: The best rule from each GP approach are compared against each other after the performances of the rules are compared. The best rule is defined to be the rule that has the lowest average performance values over all the simulation configurations out of the evolved rules. The best rules are then compared on each simulation configuration by the MWT values from the generated schedules. In other words, best rule comparison uses the results

$Obj(\omega, \gamma_{(sim)1}), \ldots, Obj(\omega, \gamma_{(sim)30})$ from the rules being applied to the 30 problem instances generated by each simulation configuration sim. The results of the best rules being applied to each simulation configuration is shown in Table 5, where each entry $\mu \pm \sigma$ is the mean (μ) and standard deviation (σ) of the MWT values generated by the best rule after being applied to 30 independent problem instances generated from the simulation configuration.

Table 5. Comparison of the best rules over the simulation configurations.

Data Subset		MB			GP
		GP-Aug	GP-WT	GP-RTR	
MWT ($\times 10^2$)	$\langle 0, 37.5, 5\rangle$	0.69 ± 0.31	0.63 ± 0.36	0.63 ± 0.31	0.63 ± 0.29
	$\langle 0, 37.5, 3\rangle$	1.06 ± 0.27	1.01 ± 0.28	1.01 ± 0.27	1.01 ± 0.25
	$\langle 0, 137.5, 5\rangle$	0.55 ± 0.17	0.47 ± 0.16	0.49 ± 0.14	0.50 ± 0.15
	$\langle 0, 137.5, 3\rangle$	1.24 ± 0.34	1.21 ± 0.39	1.23 ± 0.37	1.18 ± 0.33
	$\langle 0, 262.5, 5\rangle$	0.67 ± 0.32	0.60 ± 0.30	0.61 ± 0.27	0.63 ± 0.29
	$\langle 0, 262.5, 3\rangle$	1.30 ± 0.46	1.25 ± 0.46	1.27 ± 0.43	1.24 ± 0.41
	$\langle 0.025, 37.5, 5\rangle$	1.42 ± 0.96	1.72 ± 1.49	1.13 ± 0.70	1.19 ± 0.56
	$\langle 0.025, 37.5, 3\rangle$	2.47 ± 2.17	2.54 ± 2.14	2.15 ± 2.04	2.33 ± 2.48
	$\langle 0.025, 137.5, 5\rangle$	15.84 ± 9.08	16.73 ± 10.11	15.46 ± 8.12	17.85 ± 7.83
	$\langle 0.025, 137.5, 3\rangle$	19.36 ± 11.74	21.39 ± 12.86	18.30 ± 12.22	21.28 ± 11.67
	$\langle 0.025, 262.5, 5\rangle$	44.44 ± 24.05	51.07 ± 23.93	48.60 ± 26.00	46.87 ± 24.62
	$\langle 0.025, 262.5, 3\rangle$	50.49 ± 35.66	56.92 ± 37.17	54.08 ± 37.06	52.61 ± 35.43
	$\langle 0.05, 37.5, 5\rangle$	1.59 ± 0.63	1.63 ± 0.79	1.41 ± 0.53	1.42 ± 0.49
	$\langle 0.05, 37.5, 3\rangle$	2.92 ± 1.16	2.89 ± 1.25	2.54 ± 0.94	2.55 ± 0.88
	$\langle 0.05, 137.5, 5\rangle$	4.59 ± 2.45	4.44 ± 2.91	4.61 ± 2.41	5.62 ± 2.84
	$\langle 0.05, 137.5, 3\rangle$	7.00 ± 3.67	7.06 ± 4.24	7.01 ± 3.47	8.03 ± 3.90
	$\langle 0.05, 262.5, 5\rangle$	14.29 ± 7.32	15.76 ± 8.02	14.78 ± 7.12	16.50 ± 6.88
	$\langle 0.05, 262.5, 3\rangle$	14.30 ± 5.66	14.56 ± 6.10	14.97 ± 4.69	16.87 ± 4.78
	$\langle 0.1, 37.5, 5\rangle$	3.83 ± 1.34	3.83 ± 1.26	3.29 ± 1.10	3.40 ± 1.24
	$\langle 0.1, 37.5, 3\rangle$	4.96 ± 1.64	4.84 ± 1.42	4.45 ± 1.23	4.38 ± 1.18
	$\langle 0.1, 137.5, 5\rangle$	5.65 ± 1.13	5.41 ± 1.45	5.23 ± 1.12	5.58 ± 1.28
	$\langle 0.1, 137.5, 3\rangle$	7.71 ± 1.41	7.62 ± 1.60	7.16 ± 1.50	7.64 ± 1.69
	$\langle 0.1, 262.5, 5\rangle$	8.39 ± 2.58	7.56 ± 3.37	8.18 ± 2.77	9.28 ± 2.95
	$\langle 0.1, 262.5, 3\rangle$	10.12 ± 1.38	9.27 ± 1.59	10.44 ± 1.52	11.12 ± 1.48
	$\langle 0.15, 37.5, 5\rangle$	5.89 ± 1.48	5.52 ± 1.59	5.60 ± 1.31	5.56 ± 1.16
	$\langle 0.15, 37.5, 3\rangle$	7.70 ± 1.33	7.52 ± 1.27	7.37 ± 1.26	7.07 ± 0.99
	$\langle 0.15, 137.5, 5\rangle$	7.66 ± 1.27	6.79 ± 1.30	7.55 ± 1.15	7.90 ± 1.16
	$\langle 0.15, 137.5, 3\rangle$	10.97 ± 1.56	10.62 ± 1.88	10.51 ± 1.34	10.74 ± 1.11
	$\langle 0.15, 262.5, 5\rangle$	10.27 ± 1.25	8.85 ± 1.04	9.89 ± 1.11	10.91 ± 1.03
	$\langle 0.15, 262.5, 3\rangle$	13.76 ± 1.99	13.10 ± 2.15	13.90 ± 1.75	14.81 ± 1.67

The best rules from the machine breakdown GP approaches show greater difference in the performance to the best rule from the benchmark GP approach. The best machine breakdown GP rules are significantly better than the best benchmark GP rule for certain simulation configurations, e.g., all three machine breakdown GP rules perform better than the GP rule for the $\langle 0.15, 262.5, 3\rangle$ simulation configuration. Therefore, it is likely for GP approaches with the machine breakdown terminals to evolve high quality individual rules than the benchmark GP approach.

5.2 Rule Analysis

The evolved machine breakdown GP rules are analysed further by carrying out a qualitative analysis based on the structures of the evolved rules. First, the

best rules are simplified to remove any redundant branches (e.g. if an `if` will only return the "if" sub branch, then the `if` operator is replaced with the "if" branch) before analysing the structures of the rules. The simplified best rules for GP-Aug, GP-WT, and GP-RTR are shown in Fig. 1a, b and c respectively.

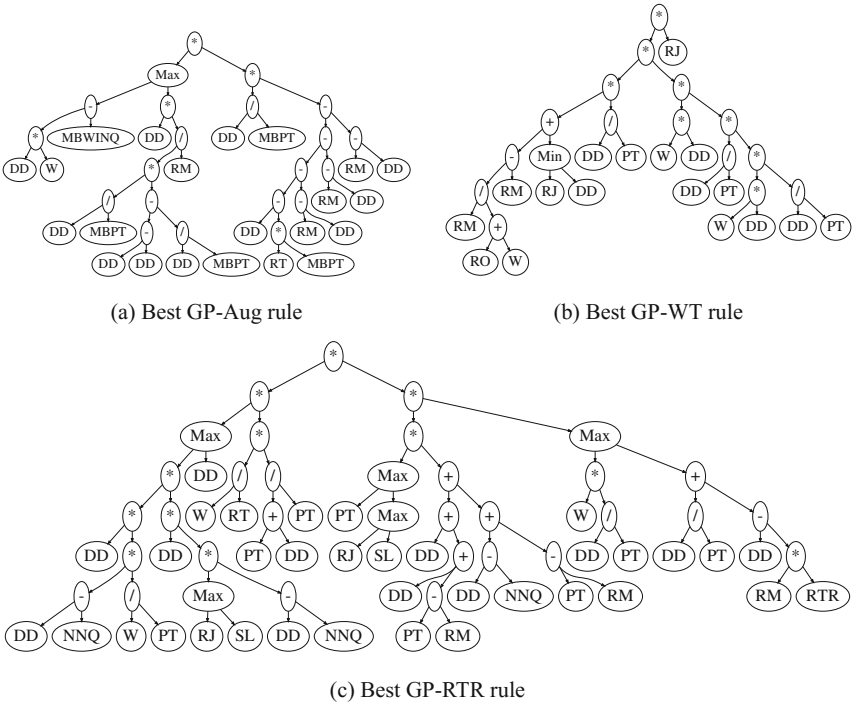

Fig. 1. The structures of the best rules found by the GP approaches.

An important observation from the best rules evolved by GP-WT and GP-RTR is the lack of machine breakdown terminals that make up the best rules. The best rule from GP-WT has *no* occurrence of the WT terminal that is incorporated into the terminal set, and the best rule from GP-RTR has *one* occurrence of the RTR terminal. Therefore, it may be the case that the machine breakdown terminals do not directly contribute towards the qualities of the final evolved rules. Instead, the machine breakdown terminals may facilitate the evolution of good GP rules, and are discarded from the best GP individuals near the end of the GP process. This may explain the lack of machine breakdown terminals in best GP-WT and GP-RTR rules, but why the best rules generally perform better than the best benchmark rule In addition, it may also explain why GP-WT and GP-RTR rules also perform slightly better than the benchmark GP rules.

For the best rules from the GP approaches, the method in which the non-machine breakdown related terminals are combined may also be a factor in the

effectiveness of the rules. These include the frequent occurrence of important terminals such as the job's weights and processing time in the best evolved rules. Intuitively, important jobs with short processing time should be prioritised out of the jobs waiting at the available machine. However, in all three machine breakdown GP rules (and the best benchmark GP rule), there are many segments of the tree that form DD/PT, which indicates that the best rules prioritise jobs with high due date and low processing time. This is contrary to the expectation that jobs with low due date (i.e. jobs that are more urgent) should be prioritised first. A possible explanation is that the due date terminal is *time variant*, i.e., expected due dates of jobs steady increases with the duration of the simulation. On the other hand, the processing time terminal is *time invariant*, i.e., the expected processing times of jobs remains relatively the same over the whole duration of the simulation. Therefore, the relative differences in the due date between an urgent job and a non-urgent job waiting on a machine late in the simulation may not be as big as the differences in their processing time, due to the large due date values of both the urgent and non-urgent jobs. This may result in the due date of a job for long simulations being used as an arbitrary large value that can be combined with the processing time terminal using the protected / operator to form a composite that prioritises short processing times. Further experiments can be carried out to determine whether the same phenomenon occurs by replacing the processing time terminal with 1/PT terminal in future GP approaches.

6 Conclusions and Future Work

This paper is a very first piece of work that develops new machine breakdown GP terminals to improve the qualities of GP evolved rules for a DJSS-MB. The first set of GP terminals (called "augmented terminals") replace existing processing time related terminals (PT, NPT and WINQ) with equivalent terminals that take potential machine breakdown into account. The second set of GP approaches (called "reactive terminals") add new terminals (RTR and WT) that gives information on current state of the shop floor. The machine breakdown GP approach does not evolve significantly better rules overall, but the best rules evolved by the machine breakdown GP significantly outperform the best rule evolved by the benchmark GP. The analysis shows very interesting results and insights, where the machine breakdown terminals appear infrequently in the best rules for GP-WT and GP-RTR. Hypotheses have been raised to explain why this is the case, and further work will be needed in this direction. We hope that this work can attract more people to start their work in this direction in the near future.

For the future work, further analysis based on the behaviours of the evolved rules will be carried out. Analysis of evolved rule behaviours in JSS problems have been carried out in the literature [17,18], and further investigation into the behaviours of rules evolved for DJSS-MB may allow better machine breakdown specific approaches to be developed. In addition, the relation between the utilisation rate of job shop scheduling problems and the machine breakdown level

will be explored further by analysing rule behaviours in different shop environments. For example, a rule evolved for shop with low utilisation rate and high machine breakdown will be compared against a rule evolved for shop with high utilisation rate and low machine breakdown. This relation may help us develop further insight into machine breakdowns and how the properties of the shop changes with such disruptions.

References

1. Potts, C.N., Strusevich, V.A.: Fifty years of scheduling: a survey of milestones. J. Oper. Res. Soc. **60**(1), S41–S68 (2009)
2. Ouelhadj, D., Petrovic, S.: A survey of dynamic scheduling in manufacturing systems. J. Sched. **12**(4), 417–431 (2009)
3. Hildebrandt, T., Heger, J., Scholz-Reiter, B.: Towards improved dispatching rules for complex shop floor scenarios: a genetic programming approach. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2010), pp. 257–264. ACM, New York (2010)
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 Trans. Evol. Comput. **17**(5), 621–639 (2013)
5. Branke, J., Nguyen, S., Pickardt, C.W., Zhang, M.: Automated design of production scheduling heuristics: A review. IEEE Trans. Evol. Comput. **20**(1), 110–124 (2016)
6. Nguyen, S., Mei, Y., Ma, H., Chen, A., Zhang, M.: Evolutionary scheduling and combinatorial optimisation: applications, challenges, and future directions. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC 2016), pp. 3053–3060 (2016)
7. Hunt, R., Johnston, M., Zhang, M.: Evolving "less-myopic" scheduling rules for dynamic job shop scheduling with genetic programming. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2014), pp. 927–934. ACM, New York (2014)
8. Yin, W.J., Liu, M., Wu, C.: Learning single-machine scheduling heuristics subject to machine breakdowns with genetic programming. In: Proceedings of IEEE Congress on Evolutionary Computation, CEC 2003, pp. 1050–1055 (2003)
9. Holthaus, O.: Scheduling in job shops with machine breakdowns: an experimental study. Comput. Ind. Eng. **36**(1), 137–162 (1999)
10. Park, J., Mei, Y., Nguyen, S., Chen, G., Zhang, M.: Investigating the generality of genetic programming based hyper-heuristic approach to dynamic job shop scheduling with machine breakdown. In: Wagner, M., Li, X., Hendtlass, T. (eds.) ACALCI 2017. LNCS (LNAI), vol. 10142, pp. 301–313. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-51691-2_26
11. Pinedo, M., Hadavi, K.: Scheduling: theory, algorithms and systems development. In: Gaul, W., Bachem, A., Habenicht, W., Runge, W., Stahl, W.W. (eds.) ORP 1991. Operations Research Proceedings 1991, vol. 1991. Springer, Heidelberg (1992). https://doi.org/10.1007/978-3-642-46773-8_5
12. Geiger, C.D., Uzsoy, R., Aytuğ, H.: Rapid modeling and discovery of priority dispatching rules: an autonomous learning approach. J. Sched. **9**(1), 7–34 (2006)
13. Nguyen, S., Mei, Y., Zhang, M.: Genetic programming for production scheduling: a survey with a unified framework. Complex Intell. Syst. **3**(1), 41–66 (2017)

14. Koza, J.R.: Genetic Programming: on the Programming of Computers by Means of Natural Selection. MIT Press, Cambridge (1992)
15. Mei, Y., Zhang, M., Nguyen, S.: Feature selection in evolving job shop dispatching rules with genetic programming. In: Proceedings of the 2016 Conference on Genetic and Evolutionary Computation, pp. 365–372 (2016)
16. Vepsalainen, A.P.J., Morton, T.E.: Priority rules for job shops with weighted tardiness costs. Manag. Sci. **33**(8), 1035–1047 (1987)
17. Hildebrandt, T., Branke, J.: On using surrogates with genetic programming. Evol. Comput. **23**(3), 343–367 (2015)
18. Hart, E., Sim, K.: A hyper-heuristic ensemble method for static job-shop scheduling. Evol. Comput. **24**(4), 609–635 (2016)