Selection of dispatching rules evolved by genetic programming in dynamic unrelated machines scheduling based on problem characteristics

Marko Đurasević[a,*], Domagoj Jakobović[a]

[a] *Faculty of Electrical Engineering and Computing University of Zagreb*
Unska 3, 10000 Zagreb, Croatia

Abstract

Dispatching rules are fast and simple procedures used for creating schedules for various kinds of scheduling problems. However, manually designing DRs for all possible scheduling conditions and scheduling criteria is practically infeasible. For that reason, a great deal of research has focused on the automatic design of DRs by using different methods, most notably genetic programming. Still, even if genetic programming is used to evolve new DRs for optimising a certain criterion, it will still not yield good results on all possible problem instances it can be applied on. Because of the stochastic nature of genetic programming, the evolution of DRs must be executed several times to ensure that good DRs were obtained. However, in the end usually only one rule from the set of evolved DRs is selected and used for solving new scheduling problems. In this paper a DR selection procedure is proposed, which selects the appropriate DR from the set of evolved DRs, based on the features of the problem instances it is solving. The proposed procedure is executed simultaneously with the execution of the system, during which it approximates the characteristics of problem instances and selects the appropriate DR for the current conditions. The obtained results show that the proposed approach achieves better results than those obtained if only a single DR was selected and used for all problem instances.

Keywords: Dispatching rules, genetic programming, scheduling, unrelated machines environment, machine learning, dispatching rule selection

1. Introduction

Scheduling is a decision making process which is concerned with the allocation of scarce resources to a certain set of activities (Pinedo, 2012). The objective of the scheduling procedure is to construct a schedule which optimises some user defined objectives. Scheduling problems appear in many real world scenarios, like scheduling tasks in multiprocessor environments (Drozdowski, 1996; Kahraman et al., 2010) or production scheduling in different manufacturing environments (Pickardt et al., 2013). In many such environments, dispatching rules (DRs) are the methods of choice for solving scheduling problems, since they construct the schedule during the execution of the system and can thus adapt to the changing conditions in the system (Braun et al., 2001; Maheswaran et al., 1999). This is possible since the execution time of DRs is quite fast and can even be considered negligible when compared to the execution times of some metaheuristic methods like genetic algorithms (Durasević & Jakobović, 2016). However, there are two problems which are commonly associated with DRs.

The first problem associated with DRs is the fact that they are quite difficult to design manually. This usually leads to a lengthy trial and error process, which is repeated until a good DR is designed. To cope with this prob-

[*]Corresponding author
Email addresses: marko.durasevic@fer.hr (Marko Đurasević), domagoj.jakobovic@fer.hr (Domagoj Jakobović)

lem, researchers began applying different machine learning procedures to automatically create new DRs for various scheduling problems. Although different approaches like neural networks (Weckman et al., 2008) and C4.5 (Olafsson & Li, 2010) were used to create new DRs, genetic programming (GP) (Poli & McPhee, 2013) has received the most attention. GP is a powerful hyper-heuristic approach often used to design new heuristics for various problems (Burke et al., 2009, 2010), which obtains good results for many different problems Koza (2010). With the use of GP it is not only possible to automate the development of new DRs, but also to produce DRs which generally outperform standard manually designed DRs. Therefore a great amount of research has lately been focused on improving the quality of DRs generated by GP Zhang et al. (2021a); Jaklinović et al. (2021); Gil-Gala et al. (2021).

The second problem associated with DRs is the need to determine which of the numerous available DRs should be applied for solving a concrete problem instance. Branke & Pickardt (2011) demonstrated that it is quite easy to create problem instances for which some standard DRs achieve quite bad results due to certain suboptimal decisions they make during the scheduling process. Therefore, even if GP is used to generate several DRs of good quality, it is still unknown which of the generated DRs should be applied for unseen problem instances. By selecting an inappropriate automatically generated DR for a problem instance, it is possible to obtain quite poor results, even though the selected DR might have a good performance on average across other problem instances. One possibility would be to use the DR which obtained the best result on the problem set used to generate the DRs. However, this does not guarantee that good results will be obtained on all problems on which the DR will be applied. Nevertheless, in previous studies it was demonstrated that by using certain machine learning algorithms it is possible to create a simple selection method which could, based on certain problem instance characteristics, determine which of the available DRs should be used for solving the current problem instance (Priore et al., 2001, 2014; Shahzad & Mebarki, 2016). Unfortunately, the research related to this area has focused exclusively on using standard manually designed DRs, with neural networks being predominantly used for the selection process.

In this paper GP is used to create a set of good DRs, and an additional selection procedure is defined, that will, based on the different properties of problem instances, determine which of the generated DRs should be used for solving them. The aim of this paper is to analyse if by using such a combined procedure it is possible to acquire results which are better than by using a single automatically generated DR on all problem instances. If such a combined approach would obtain good results, then it would be possible to design scheduling procedures with improved performance and robustness, since they would be more adaptable to the problem instances they are solving. The proposed procedure was extensively tested with various parameter values and for solving different problem types. The paper provides an analysis on the parameter sensitivity of the procedure and outlines how it compares to the situation when a single manually selected DR generated by GP would be used for solving all problem instances. Finally, the paper provides an additional analysis of the selection procedure by outlining its performance and behaviour on several selected examples. The results which were obtained through an extensive suite of experiments demonstrate that the procedure is capable of selecting appropriate DRs, and obtain better results than by using a single manually selected DR.

The rest of the paper is organised as follows. In the first part of Section 2 an overview of the related work concerned with automatic design of DRs by using GP is given, while the second part of the section gives an overview of the research dealing with automatic selection of DRs. The background of the problem and solution method is given in Section 3, with Subsection 3.1 describing the unrelated machines environment and Subsection 3.2 describing the application of GP for creating new DRs. The details of the DR selection method are presented in Section 4. The benchmark setup and parameter analysis of the DR selection procedure are described in Section 5. Section 6 outlines the obtained results, while Section 7 provides a further analysis of the DR selection procedure. Finally, Section 8 gives a short conclusion about the achieved results, and outlines the possibilities for future work.

2. Related work

2.1. Automatic generation of DRs by using GP

One of the first attempts in creating DRs by using GP was done by Dimopoulos and Zalzala for the single machine environment (Dimopoulos & Zalzala, 1999, 2001). In their studies they demonstrated that GP can evolve DRs of better quality than some standard manually designed DRs. Miyashita (2000) applied GP to generate DRs for the job-shop environment. He proposed three models: the homogeneous model, in which each machine used the same DR; the distinct agent model, in which each machine used a different DR; and the mixed agent model, in which each machine uses one of two generated DRs. The idea behind the mixed agent model is that certain machines can represent bottlenecks, and therefore a specific DR should be generated for such machines, while all other machines would use a DR evolved for non-bottleneck machines. Although the mixed agent model achieved the best results, it has the disadvantage that it needs to know which machines represent bottleneck resources and which do not, since the mapping between DRs and machines is done before the system execution. In order to alleviate the aforementioned problem, Jakobović & Budin (2006) have used GP to evolve three expressions, instead of one. Two of the evolved expressions represent DRs, one of which is evolved for machines which represent bottleneck resources, while the other is evolved for regular machines. The third expression represents a decision function that for each machine determines which of the two aforementioned DRs will be used to schedule the jobs. Therefore, during system execution it can be determined which machine is a bottleneck resource, and thus use the appropriate DR.

Tay & Ho (2008) have analysed the generation of DRs for a scheduling problem in which three criteria were optimised simultaneously. The optimisation was performed in a way that all three criteria were combined into a single criterion, and thus the whole problem was reduced to a single objective problem and solved appropriately. Various multi-objective scheduling problems have been studied in many subsequent surveys (Nguyen et al., 2013a, 2015b; Karunakaran et al., 2016; Durasević & Jakobović, 2017b). Jakobović & Marasović (2012) investigated the creation of dispatching rules for the single machine environment with precedence constraints and setup times. They also analysed the influence of GP parameters on the quality of the evolved DRs for the single machine and job-shop environments. Nguyen et al. (2013b) have analysed the different solution representations for evolving DRs, and have shown that the quality of the evolved DRs strongly depends on the chosen representation. To improve the performances of DRs in the static scheduling environment, Nguyen et al. (2013c) proposed generating iterative dispatching rules (IDRs), which build schedules several times and use information from previous schedules to improve the quality of newly built schedules. Nie et al. applied gene expression programming (GEP) (Ferreira, 2001), a procedure similar to GP which instead of a tree representation uses an array representation, in order to create DRs for the single machine (Nie et al., 2010) and job-shop environments (Nie et al., 2011). Ðurasević et al. (2016) have compared several GP approaches (IDRs, GEP, standard GP and dimensionally aware GP Keijzer & Babovic (1999)) to create DRs for several scheduling criteria in the unrelated machines environment. To increase the performance of generated DRs, they are often grouped into ensembles and used to perform scheduling decision (Park et al., 2015; Hart & Sim, 2016; Durasević & Jakobović, 2017a; Park et al., 2018; Ðurasević & Jakobović, 2019).

Some recent studies examined the possibility of generating DRs for the resource constrained project scheduling problem (Chand et al., 2018; Ðumić et al., 2018) and the single machine problem with variable capacity that was modelled from a real world problem Gil-Gala et al. (2019, 2021). For both problem variants it was demonstrated that using GP it was possible to evolve DRs which perform significantly better than existing manually designed DRs. To improve the results of generated DRs and gain a better insight in the decision process which is performed by them, Ðurasević & Jakobović (2020) have analysed and compared different schedule generation schemes that can be used in automatically designed DRs. The results suggest that it is important that the GP can evolve a DR which performs both sequencing and scheduling decisions. Đurasević & Jakobović (2020) have examined how automatically designed DRs can be adapted for solving static scheduling problems. They applied several methods and demonstrated that such DRs can achieve results which can even be comparable to results obtained by some metaheuristic methods. Generating DRs for problems which include additional constraints like setup times, machine eligibility constraints, machine unavailability periods and job precedences was considered by Jaklinović et al. (2021). Zhang et al. (2021b) proposed a novel recombination mechanisms which is applied for generating DRs in the flexible job shop environment. The recombination strategy is based on choosing crossover points at places where good subtrees are located to enforce the exchange and preservation of promising building blocks between parents. Zhang et al. (2021a) propose a multitask GP hyperheuristic for dynamic scheduling. This model is tasked with generating heuristics for a wider class of problems, rather than individual problems. The obtained results demonstrate the efficiency of this methods over standard hyperheuristic methods. Zhang et al. (2021c) improve upon the multitask learning paradigm for evolving DRs. The goal of this study is to improve the effectiveness of the paradigm by using surrogate models by improving its efficiency but also for knowledge transfer in the multitask paradigm. Additional studies dealing with automatic design of DRs can be found in two detailed surveys (Branke et al., 2016; Nguyen et al., 2017).

2.2. Overview of DR selection literature

The selection of dispatching rules based on the system status and problem characteristics has been studied to a certain extent in the literature. Pierreval (1992) applied an expert system to dynamically select dispatching rules in the flexible manufacturing systems. Sun & Yih (1996) proposed a controller for a manufacturing cell which uses a neural network to select a proper DR based on the current performance of the system. Pierreval (1993) also used neural networks to select the appropriate DR for a simplified job shop environment consisting of two work centres. In the aforementioned work the neural network performs its decision by using information about the configuration of the shop floor, characteristics of the manufacturing program, and the optimised performance criteria. Liu & Dong (1996) proposed an algorithm for job shop scheduling problems which uses neural networks to determine appropriate DRs, and uses simulation to evaluate the efficiency of all the DRs. The results of the simulations are used for training the neural network which is applied in the real-time scheduling process, and selects the DRs that should be applied. Pierreval & Mebarki (1997) analysed dynamic rule selection in a manufacturing system, where the rule selection is carried out each time a machine becomes available, and the rules are selected based on certain system parameters.

Instead of using a machine learning approach, Lian Yu et al. (1998) opted to use a fuzzy inference based system to select appropriate DRs. The system uses two fuzzy rules and several environment variables for selecting the DR that should be used for scheduling. Subramaniam et al. (2000b) also used fuzzy logic for defining a scheduler which dynamically selects the most appropriate DR from a set of candidate DRs. Their results show that the fuzzy scheduler performs better than some commonly used DRs, and requires the same computational time as the simple DRs. In Subramaniam et al. (2000a) a scheduling method based on the analytic hierarchy process which dynamically selects the most appropriate DR from several available rules was proposed. El-Bouri & Shah (2006) used neural networks for DR selection in the job shop environment. Their approach does not only select a single DR, but rather it selects an appropriate DR for each machine in the shop. The experiments show that by using such an approach, they achieve better results than by using the same DR for every machine. Shiue & Guh (2006) used a hybrid learning methodology which applies a neural network to select the appropriate DR for the current system conditions, and a GA which is used to select the best set of attributes for the input layer of the neural network.

Mouelhi-Chibani & Pierreval (2010) used a neural network approach for selecting a suitable DR after each machine becomes available. In this study the neural network is not trained on prepared training instances, but rather by using a simulation approach. This means that weights of the neural network, which acts as a decision maker, are optimised to achieve the best performance on the simulation. The method can automatically select efficient DRs for the given situation. Heger et al. (2015) used Gaussian processes for the selection of an appropriate DR. In their approach they used two system parameters, system utilisation and the due date factor, as well as three manually defined DRs. Their research showed that it was possible to achieve better results by rule switching than by applying a single DR for the entire problem instance. Zahmani et al. (2015) also use a simulation approach to extract data that can be used to infer which DR should be used in which situations.

Zhang & Roy (2018) propose a semantics-based approach to scheduling jobs in the job shop environment. The proposed approach calculates a semantic similarity value based on the expression of the DRs, based on which it then performs the decision on which DR to select for the given production objectives. Amin & El-Bouri (2018) consider the problem of selecting DRs in problems where multiple criteria have to be optimised. They apply a minimax linear programming model which used a preference voting system to select DRs which perform well across all the criteria, and not only on a single criterion. Zahmani & Atmani (2019) use data mining techniques to select DRs for makespan minimisation in the job shop scheduling environment. The proposed approach demonstrated an improved performance over individual DRs. Ahn & Hur (2020) apply a neural network to select the appropriate DR for scheduling job batches which are first created using clustering methods. An overview of other research concerned with the automatic selection of DRs can be found in (Priore et al., 2001, 2014; Shahzad & Mebarki, 2016).

3. Background

3.1. Scheduling in the unrelated machines environment

The unrelated machines environment is a scheduling environment which consists of m machines which independently execute jobs, and n jobs which need to be scheduled on one of the available machines. Each job has a processing time p_{ij} which determines the amount of time that is needed to process the job with index j on the machine with index i. Since the processing times depend both on the jobs and the machines, it is hard to infer relations between the machines (for example that one machine executes all jobs two times faster than another machine). Release time r_j is another important characteristic defined for each job, which determines when the job will be released into the system, i.e. when it becomes available for scheduling. Each job can also have a due date d_j, which defines the time until which the job needs to finish with its execution. The job can finish at a later moment in time, but then it induces a certain penalty. Finally, each job can also have a weight w_j, which defines the importance of each job.

To determine the quality of the created schedule, one or more scheduling criteria are used. Although many different scheduling criteria exist, this research will focus solely on optimising the weighted tardiness criterion, which is defined as

$$Twt = \sum_{j} w_j T_j, \tag{1}$$

where T_j denotes the tardiness of a single job and is defined as

$$T_j = \max\{C_j - d_j, 0\}, \tag{2}$$

and C_j denotes the completion time of job j.

By using the $\alpha|\beta|\gamma$ notation of scheduling problems (Pinedo, 2012), the problem which is considered in this paper can be denoted as $Rm|r_j|\sum_j w_j T_j$. Additionally, the scheduling conditions under which the problem is solved also need to be defined. Scheduling problems can be divided into static and dynamic scheduling problems. In static scheduling all relevant information about the jobs is known before the system starts with the execution. Therefore the schedule can be constructed before the system begins with its execution. In such scheduling environments different metaheuristic procedures are often used to obtain a good solution. On the other hand, in the dynamic scheduling environment job parameters become available only when the corresponding job is released into the system. Since in such systems all required parameters for creating the solution are not present beforehand, metaheuristic methods usually cannot be used to solve such problems. Therefore, DRs are the method of choice for solving problems in such conditions. This paper presumes that scheduling is carried out in the dynamic scheduling environment, in which job parameters become available when they are released into the system and that the schedule is created in parallel with the system execution.

3.2. Evolving DRs by using GP

To generate DRs for the unrelated machines environment, the GP methodology proposed by Đurasević et al. (2016) is used in this study. The DRs which are constructed in this paper consists of two parts: a schedule generation scheme (SGS) and a priority function (PF). The PF determines the priority value for a given job machine pair. The SGS uses the PF to obtain priority values for job-machine pairs and determine which job should be scheduled on which machine and in what order. The SGS used in this paper is presented in Algorithm 1. In the first step this SGS determines the priority values using the PF for each job-machine pair, based on which it determines the "best" machine for each job. In the second step it tries to schedule each job on its "best" machine (in the order of their priorities). If the machine on which the job would need to be scheduled is occupied, then the scheduling of this job is postponed to a later moment in time, otherwise the job is scheduled on the appropriate machine. This SGS was designed manually, however, it can be used with any PF that is provided to it and does not need further adaptation after a certain PF was evolved.

Algorithm 1 Schedule generation scheme used for GP scheduling

```
 1: while unscheduled jobs are available do
 2:     wait until a job becomes ready or a job finishes;
 3:     for all available jobs and all machines do
 4:         obtain the priority $\pi_{ij}$ of job $j$ on machine $i$;
 5:     end for
 6:     for all available jobs do
 7:         determine the best machine (the one for which
 8:         the best value of priority $\pi_{ij}$ is achieved);
 9:     end for
10:     while jobs whose best machine is available exist
11:      do
12:         determine the best priority of all such jobs;
13:         schedule the job with the best priority;
14:     end while
15: end while
```

The PF, which is used in the SGS in line 4 to determine the priorities of job machine pairs, is evolved automatically by using GP. In order for GP to design new PFs, the basic building blocks used in the evolution process need to be defined. This is done by defining the terminal and function nodes which are used by GP. Table 1 represents the list of terminal and function nodes applied in the evolution process. As it can be seen from the table, all the terminal nodes represent certain job or system characteristics. The *time* variable represents the current time of the system. For the

Table 1: Terminal and functional nodes

Node name	Description
pt	processing time of job j on the machine i (p_{ij})
pmin	the minimal job processing time on all machines: $\min(p_{ij}) \forall i$
pavg	the average processing time on all machines
PAT	patience - the amount of time until the machine with the minimal processing time for the current job will be available
MR	machine ready - the amount of time until the current machine becomes available
age	the time that the job spent in the system: $time - r_j$
dd	due date (d_j)
w	weight (w_j)
SL	positive slack: $max(d_j - p_{ij} - time, 0)$
+	binary addition operator
-	binary subtraction operator
*	binary multiplication operator
/	secure binary division: $/(a,b) = \begin{cases} 1, \text{ if } \lvert b \rvert < 0.000001 \\ \frac{a}{b}, \text{else} \end{cases}$
POS	unary operator: $POS(a) = max(a,0)$

function nodes only five operators are used, since in a previous study it was shown that introducing more sophisticated function nodes does not lead to improvement in the performances of the developed DRs (Đurasević et al., 2016).

The parameters which were used by GP for the construction of DR are denoted in Table 2. All the listed parameters were previously thoroughly optimised by using a problem set to evolve new DRs, and an independent problem set to determine for which parameter values GP obtained the best average results. Both of these sets were different from those which were used in this study to validate the proposed method. Since several crossover and mutation operators are used, each time such an operator needs to be applied, one of the defined operators is selected randomly with the same probability. The reason why a set of operators are used is because in preliminary parameter tuning it was demonstrated that GP on average produces better DRs when a larger set of genetic operators is used.

4. DR selection procedure

The aim of the DR selection procedure proposed in this paper is to determine the appropriate DR for solving the currently considered problem instance. However, to select the appropriate DR the procedure needs to learn which DRs should be used in which situations, and then apply this knowledge on unseen scheduling problems. The rest of this section will describe all the individual parts of the DR selection procedure.

4.1. Problem instance features

The first, and probably most important step is to identify the most informative features of the problem instances that can be used to select the DR that should be used for the current problem instance. The goal here is to find the

Table 2: Parameters for the GP

Parameter	**Value**
Population size	1000
Termination criteria	maximum number of iterations (80 000)
Selection	steady state GP using tournament selection
Tournament size	3
Initialization method	*ramped half-and-half*
Mutation probability	0.3
Maximum tree depth	5
Crossover operator set	subtree, uniform, context-preserving, size-fair
Mutation operator set	subtree, Gauss, hoist, node complement, node replacement, permutation, shrink

smallest, but also the most informative set of features. In this section the influence of the following six features will be analysed:

- Expected total duration of the schedule: $\hat{p} = \frac{\sum_{j=1}^{n}\sum_{i=1}^{m} p_{ij}}{m^2}$,
- Due date tightness: $DDT = 1 - \frac{\sum_j (d_j - r_j)}{n\hat{p}}$,
- Due date range: $DDR = \frac{\max_j (d_j - r_j) - \min_j (d_j - r_j)}{\hat{p}}$,
- Machine job ratio: $MJR = \frac{m}{n}$,
- Release time tightness: $RTT = 1 - \frac{\bar{r}}{\hat{p}}$,
- Load disbalance: $LD = \frac{\max_i (L_i) - \min_i (L_i)}{\max_i (L_i)}$,

where n represents the total number of jobs in the problem instance, m represents the total number of machines in the problem instance, $\bar{r}$ represents the average value of all job release times in the problem instance. L_i represents the sum of processing times p_{ij} for machine i only of those jobs which have the lowest processing time on that machine. This means that when calculating For example, in a problem with two machines and 5 jobs, where job 1 has processing times $p_{11} = 2$ and $p_{21} = 4$, job 2 $p_{12} = 3$ and $p_{22} = 1$, job 3 $p_{13} = 1$ and $p_{23} = 3$, job 4 $p_{14} = 3$ and $p_{24} = 4$, and job 5 $p_{15} = 2$ and $p_{25} = 3$. In this case $L_1 = 8$, since jobs 1, 3, 4, and 5 have a lower processing time on machine 1 than on machine 2 and therefore the values of those four are summed up. On the other hand $L_2 = 1$ since only job 2 has a lower processing time on machine 2 than on machine 1.

Since the *Twt* criterion will be optimised in the experiments, the due date tightness and due date range are the most important features, since they provide the most information about the due dates of the jobs. The expected total duration approximates the makespan of the schedule, while the machine job ratio gives an information on the average number of jobs that will be scheduled per machine. The release time tightness gives information of the distribution of the release times.

4.2. The learning process

The next step in the automatic rule selection procedure is to define the learning process which will be used to learn which problem instance should be solved by which DR. To achieve this, three things need to be defined: the set of available DRs, the *learning set* on which the classifier will learn which DR should be associated with which problem instances, and a classifier which will be trained on the aforementioned learning set, and will decide which DR to use for a new problem instance.

The set of available DRs which can be used and based on which the selection will be obtained are generated by using the GP procedure. The way in which a set of DRs is obtained is by simply executing the GP procedure several times independently, and storing the best solution that was obtained after the optimisation process. This process can be repeated an arbitrary amount of times to obtain a set of DRs of the desired cardinality. In this study, a set consisting of 50 generated DRs was generated and used for constructing the learning set.

The learning set specifies which DRs should be used for solving problems with which characteristics, and is then used to train a classification algorithm which should learn such associations to determine which rules to use for new and unseen problem instances. This set consists of a set of pairs (instance,rule), where the instances are characterized by a set of features, and the rule denotes which of the available DRs is associated to the specific problem instances. In the last subsection the feature calculation technique of the problem instances was defined, but still it is required to determine which DR should be used for solving which problem instance. For the purpose of associating DRs to problem instances a procedure called the *grouping association scheme* (GAS) is defined.

Algorithm 2 represents the pseudo-code of GAS. This procedure uses a set of problem instances and a set of DRs to construct the learning examples. In the first step, for each problem instance the set of DRs which obtain the best solution for that problem instance is determined. All problem instances which have only one DR in their set are marked as *solved*, since for them there is no ambiguity regarding to which DR should be associated with them. In addition, all DRs which are associated with *solved* problem instances are collected in the *rlist* set. In the next phase, it is determined for which problem instances the DRs contained in *rlist* cannot obtain the best solution, and those problem instances are collected in a set denoted as *nonopt*. The idea is now to add the minimum number of DRs to *rlist*, so that for each problem instance there is a DR which can find the best solution for it. To achieve this, for each DR the number of problem instances in *nonopt* for which it finds the best solution is calculated. The DR which finds the best solution for most of the problem instances in *nonopt* is added to *rlist*, while the problem instances for which it finds the best solution, are removed from *nonopt*, and marked as solved. This is repeated until *nonopt* becomes empty, since then *rlist* contains DRs which can find the best solution for every problem instance. The entire procedure is done in this way to minimise the number of DRs which will be used as labels.

In the final step the procedure determines the DRs which should be associated with problem instances which are not yet marked as solved, and for each such instance and each DR in *rlist* which finds the best solution for this problem instance, the average Euclidean distance is calculated between the features of the current problem instance and all problem instances which are until now associated with the considered DR. The DR with the smallest Euclidean distance is chosen and associated with the considered problem instance. This part represents a very simple grouping procedure, which groups problem instances with similar characteristics to the same DR, and thus balances the dataset so that the best DR is not used as the label for most of the problem instances, which would lead the procedure to mostly use a single DR on new problem instances. Finally, the features of each problem instance are calculated based on the characteristics of all the jobs in the problem instance, and the DR associated to that problem instance is used as the label. An important characteristic of the GAS procedure is that it does not necessarily generate a learning set which contains all of the DRs which were supplied to the procedures, meaning that it automatically eliminates DRs for which it determined that they do not provide useful information.

After its construction, the learning set can be used to train a classifier, that will determine which DRs should be applied for unseen problem instances. The selection of a good classifier for a given classification problem is in itself a very difficult task, especially since many different classification methods exist. Therefore, several popular classification

Algorithm 2 Pseudo-code of the GAS procedure

```
1: Let $P$ denote the set of all problem instances
2: Let $R$ denote the set of all available DRs
3: Let $rlist$ represent an empty set of DRs
4: Evaluate all DRs in $R$ on $P$
5: for each problem instance $i$ in $P$ do
6:     Let $opt[i]$ denote the set of DRs which achieve the best result for this problem instance
7:     Let $rule[i]$ denote the DR chosen for this problem instance (empty in the beginning)
8: end for
9:
10: for each problem instance $i$ with only one DR in $opt[i]$ do
11:     Set $rule[i]$ to the DR in $opt[i]$, and add the DR to $rlist$
12:     Mark the problem instance as solved
13: end for
14:
15: Let $nonopt$ denote the set of all unsolved problem instances $i$ for which $opt[i] \cap rlist = \emptyset$
16: while $nonopt$ is not empty do
17:     For each DR define $count$ which denotes the number of problem instances in $nonopt$ for which the given DR
        achieves the best solution
18:     Let $BR$ denote the rule with the largest $count$ value. If there are more DRs with the same $count$ value, choose
        the one which has the best total fitness on all problem instances
19:     Add $BR$ to $rlist$
20:     for each problem instance $i$ in $nonopt$ that contains $BR$ in $opt[i]$ do
21:         Remove $i$ from $nonopt$
22:         Mark $i$ as solved
23:     end for
24: end while
25:
26: for each unsolved problem instance $i$ do
27:     for each DR in $rlist$ which is also contained in $opt[i]$ do
28:         Calculate the average Euclidean distance between the features of the current problem instance, and all
            problem instances which have the chosen DR in their $rule$ field
29:     end for
30:     Set $rule[i]$ to the DR with the smallest average Euclidean distance
31: end for
32:
33: for each problem instance $i$ in $P$ do
34:     Create a learning example with features of the problem instance and with $rule[i]$ representing the label associ-
        ated with the features
35: end for
```

methods will be used: k nearest neighbours (knn) (Fix & Hodges, 1951), naive Bayes (Alpaydin, 2014), artificial neural network (ANN) (Werbos, 1974) and C4.5 (Quinlan, 1993). Other classification methods were also tested (logistic regression (Cox, 1958), support vector machine (Cortes & Vapnik, 1995)), but they mostly achieved inferior results. A potential reason why this could be the case is that the original algorithms were designed for binary classification, therefore it might be that their adaptation for multi class problems lead to slightly worse results than those which were obtained by the other algorithms. Most of the aforementioned classification methods have parameters which should also be fine-tuned to achieve better results. However, fine-tuning parameters of all the methods would be too time consuming, therefore the values of the parameters where chosen as a rule of thumb. The classification process was performed by using the Accord machine learning library (Souza, 2018).

4.3. The decision process

With the learning set and the classification method available, it is now only left to specify how the selection of DRs will be performed during the execution of the system. The idea of the decision process is to approximate the features of the problem instances based on the characteristics of already released jobs, and based on those approximations to determine which of the available DRs would be the most appropriate for scheduling under such conditions. The selection procedure can be applied one or more times during the execution of the system, and the jobs which will be used for calculating the approximated features of the problem instance can be selected in various ways.

One thing still needs to be defined so that the entire procedure can be applied, and that is which DR will be used in the beginning, until the DR selection procedure is executed for the first time. At this moment in time no system information is available, and as such there is no way to select which DR should be used based on some problem instance characteristics. Naturally, any DR can be selected as the initial rule, and the performance of the entire procedure will depend on this selection. In the experiments performed in this paper, the rule which achieved the best results on the training set will be selected as the initial rule.

It should be stressed out that when training the classification methods on a set of problem instances, they use features which are calculated from the entire data of the problem instances. This is the case since all parameters are known for these instances, and thus it is easier to calculate their feature values. However, when the classification methods are executed to select the appropriate DR in dynamic environments, then not all information is available, but rather just the information about the jobs which were released until the current point in time that the classification methods are applied. Therefore, on the validation and tests sets, these methods will not have access to information about jobs which were not yet released into the system until the current decision point, and will have to perform their decisions only based on the information about the jobs that were released until now.

4.4. Overview of the DR selection procedure parameters

In this section the parameters which need to be defined by the DR selection procedure will be shortly described, and the values which were tested for each of the parameters will be enumerated. The parameters of the procedure are:

- **Classification method** - the classification method used for determining which DR should be used:
  - C4.5
  - k nearest neighbours - will be used with 5 and 7 neighbours
  - Naive Bayes
  - Artificial neural network - will be used with a single hidden layer with five neurons (ANN-5). The weights will be randomly initialised and tuned by using the backpropagation algorithm with Levenberg-Marquandt.
- **Feature set** - four feature sets made out of the baseic six features are used:

1. *DDR*, *DDT*
2. *DDR*, *DDT*, *LD*
3. *DDR*, *DDT*, $\hat{p}$, *MJR*, *RTT*
4. *DDR*, *DDT*, $\hat{p}$, *MJR*, *RTT*, *LD*

- **Decision interval** - the interval between performing two DR selections:
  - Number of released jobs - the decision will be performed after a certain number of jobs are released into the system. For this parameter the following values are used: 50, 100, 150, 200, 250, 300, 350, 400, 450, 500.
  - Elapsed time - the decision will be performed after a certain amount of time has elapsed. The choice of the values for this parameter depends on the processing time distribution of the jobs. For this parameter the following values are used: 50, 100, 150, 200, 250, 300, 350, 400, 450, 500.
- **Number of DRs** - the number of DRs used for constructing the learning set. This parameter allows us to investigate whether it is better to supply larger or smaller sets of DRs to the learning algorithm. In this study 10, 15, 20, 25, 30, 35, and 40 best DRs will be used when constructing the learning set.
- **Feature calculation** - determines the method of selecting the jobs which will be used for feature approximations:
  - Released - approximates the features based on the released jobs
  - Scheduled - approximates the features based on the scheduled jobs
- **Decision method** - determines which jobs will be selected for approximating the features of the problem instance:
  - No-repeat - performs the selection only once after a certain decision interval has elapsed. The selection is performed based on all jobs released or scheduled from the start until the moment the given interval has elapsed. The motivation of this strategy is to wait until a critical amount of jobs are released so that the features of the problem instance can be calculated, and then to perform the switch to a more appropriate rule. However, no further switches are performed until the end of the system execution
  - Repeat - performs the selection each time the decision interval has elapsed by using all released or scheduled jobs from the start. This means that the strategy is repeated several times during the execution of the system. The reasoning for this strategy is to repeat the selection procedure several times so that the information about new jobs that were released can be used to more correctly approximate the features of the instance and perform a better decision
  - Window - performs the selection each time the decision interval has elapsed by using all released or scheduled jobs during the last decision interval. Similar as the repeat method this one performs several selections of DRs during system execution. However, unlike the repeat method, this method calculates the features of the instance only based on the jobs from the last interval. The rationale behind this method is that the instance characteristic might change slightly over time, and as such it might be better to consider only the most recent jobs when calculating the features of the instance

5. Benchmark setup and parameter analysis

5.1. Experimental design

To train and test the DR selection procedure, four disjunct problem instance sets need to be defined: training set, learning set, validation set, and test set. The training set is used by GP to generate the individual DRs. To obtain a set of DRs which can be used for selection, 50 independent GP runs were executed. From each run the best DR in the final population was saved to obtain a set of 50 DRs that will be used in the subsequent steps. The learning set is used to train the classifier to select which of the evolved DRs to apply given the the current problem instance characteristics. This set is constructed as described in the previous section, and used to train the classification method. The validation set is used to analyse the sensitivity and tune the parameters of the DR selection procedure. Finally, the test set is a problem instance set which was not used in any of the learning or tuning processes, and is used to determine the performance of the procedure on a yet unseen set of problem instances. The training set, validation set, and test set all consist out of 60 problem instances, while the learning set consists out of 180 problem instances. The reason for this is that through preliminary experiments it was demonstrated that the selection procedure did not perform well if only 60 problem instances would be used in the learning set. Each problem instance in the validation set and test set contain 1000 jobs and 10 machines. On the other hand, the training and learning sets are constructed by using smaller problem instances, which consist out of 10 to 100 jobs and 3 to 10 machines. In this way GP and the DR selection procedure will be trained on smaller problems which are likely to occur in larger and long lasting system executions. GP will therefore generate DRs for various problem types and situations, and the DR selection procedure then needs to learn which DRs are appropriate for which situation. The instances used in the training and learning set were constructed in a similar way as in a previous study (Đurasević et al., 2016), whereas the validation and training sets were constructed differently, as described in continuation.

The properties of jobs in the problem instances will be generated in the following way

- Processing times: $p_{ij} \in [0, 100]$
- Job weights: $w_j \in < 0, 1]$
- Release times: $r_j \in \left[0, \frac{\hat{p}}{2}\right]$, where $\hat{p}$ is defined as $\hat{p} = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m} p_{ij}}{m^2}$
- Due dates $d_j \in \left[r_j + p_{min} + \max\left(p_{avg} * \left(1 - T - \frac{R}{2}\right), 0\right), r_j + p_{min} + \max\left(p_{avg} * \left(1 - T + \frac{R}{2}\right), 0\right)\right]$,where p_{min} represents the minimum processing time of job j, p_{avg} the average processing time of job j on all machines, R and T the due date range and due date tightness, respectively.

The due date range specifies the dispersion of the due dates of jobs, while with the due date tightness the amount of jobs that will be late can be adjusted. Both of those parameters assumed values of 0.2, 0.4, 0.6, 0.8 and 1 in various combinations while generating the problem set. With this, the problem sets will contain instances which have different due date characteristics, and as such will consist of problems with different characteristics, from those which are more easily solvable, to harder ones.

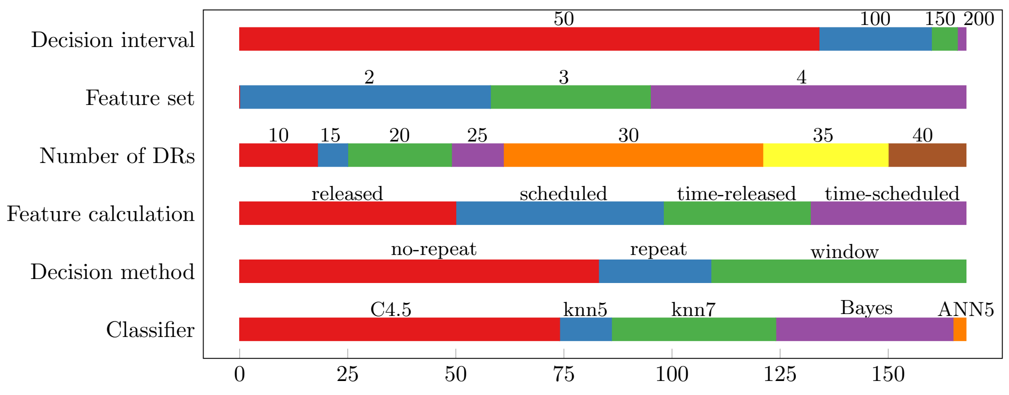
To test the generality of the proposed approach and its ability to adapt to problem instances with different characteristics and behaviour during execution, three problem instance types were used for the validation and test set. These three problem types include: problems with constant characteristics, problems with changing due date characteristics, and problems with changing due date and release time characteristics. In the first problem type, all jobs were generated with the same due date tightness and due date range parameters which were defined for the entire problem instance. Therefore, the characteristics of all jobs should be roughly the same during the execution of the system. In the second problem set type, the due date tightness and due date range characteristics of jobs in the problem instance will change over time. In this problem type, each instance will have three values defined for those two parameters. The first third of the released jobs will have due dates generated by the first values of due date tightness and due date range parameters. The due date values of the second third of released jobs will be generated by using the second pair of values, while the due dates for the last third of the released jobs will be generated by the third pair of values. In the final problem type, the due dates were generated in the same way as described for the second problem set type. However, in addition to the changing due date characteristics of the system, the release time of jobs will also be distributed more unevenly. One third of jobs will have their release times randomly generated from the interval $r_j \in [0, 0.2 * \hat{p}]$, the second third will have their release times generated from the interval $r_j \in [0.2 * \hat{p}, 0.3 * \hat{p}]$, while the release times of the remaining jobs are generated from the interval $r_j \in [0.3 * \hat{p}, 0.5 * \hat{p}]$. Therefore, less jobs will be released at the beginning and the end of the system, while most of the jobs will be released during the middle of the system execution. This will also result in more tardy jobs, which will be seen from larger *Twt* values in later sections. For all three problem types a separate validation and test set will be generated to test the performance of the DR selection procedure on them.

5.2. Parameter analysis of the DR selection procedure

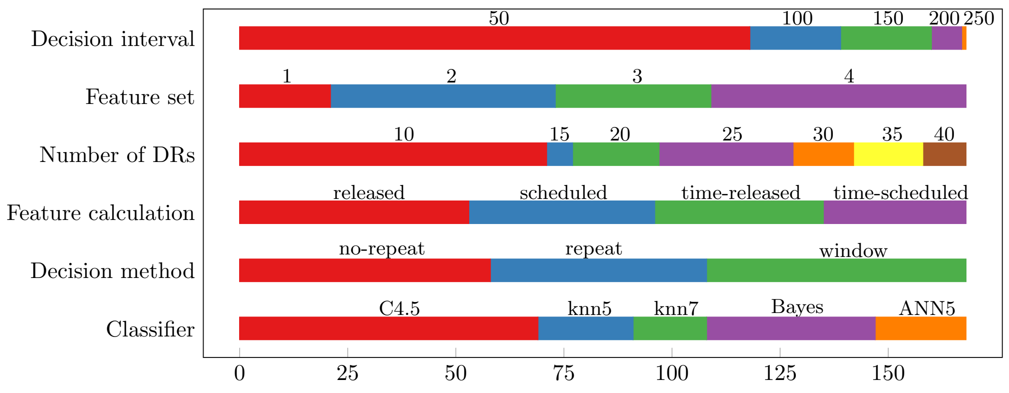
This section will analyse the influence of the DR selection procedure parameters on its performance on the validation set. Since a large number of parameters were optimised, it is difficult to determine the influence of each parameter, especially since they seem to be quite heavily correlated, and for some parameter values a quite small difference was obtained. Therefore, all possible combinations of the parameter values denoted in the previous section were tested, which in the end lead to over 16000 obtained results. However, for many of the tested combinations the procedure performed quite poorly, since a bad combination of parameters was used. Thus, for the parameter sensitivity analysis only the parameter combinations which achieve the top 1% of results will be considered. From these top results the frequency of each of the parameter values will be calculated and used to determine the best parameter values.

Figure 1 shows the frequency of each of the tested parameter values for the three different problem types. For the classification method, the frequency of all parameter values is similar across all three problem types. The C4.5 classifier has most often resulted in the best parameter values. Out of the other classification methods knn with 7 neighbours and naive Bayes also most often lead to the best results. The artificial neural networks rarely resulted in the best values, although it can be seen that the performance of this classification method improves on the problem types with changing characteristics. The decision method parameter shows to be influenced by the problem type a bit more. For the problems with constant characteristics the no-repeat method most often leads to the best results. This is expected, since the characteristics of the problem instances do not change over time, therefore performing the decision only once seems to be enough to select the appropriate DR. The repeat method has resulted in the best results the least number of times, which denotes that calculating the features based on all jobs released or scheduled until now does not seem to be beneficial. For the problems with changing due date characteristics the window method most often results in the best results, although followed closely by the no-repeat method. Thus, as the due date characteristics of the problem instances change over time, the selection procedure should be performed several times, but by using only the jobs released or scheduled during the last decision interval. However, for the problem type with changing characteristics, the no-repeat method most often leads to the best results, which is quite surprising. For the decision intervals it can be seen that there is no difference when approximating the features by using either scheduled or released jobs. In addition, neither by performing the decision after a fixed time interval nor after a certain number of jobs are released does have an influence on the results of the procedure.

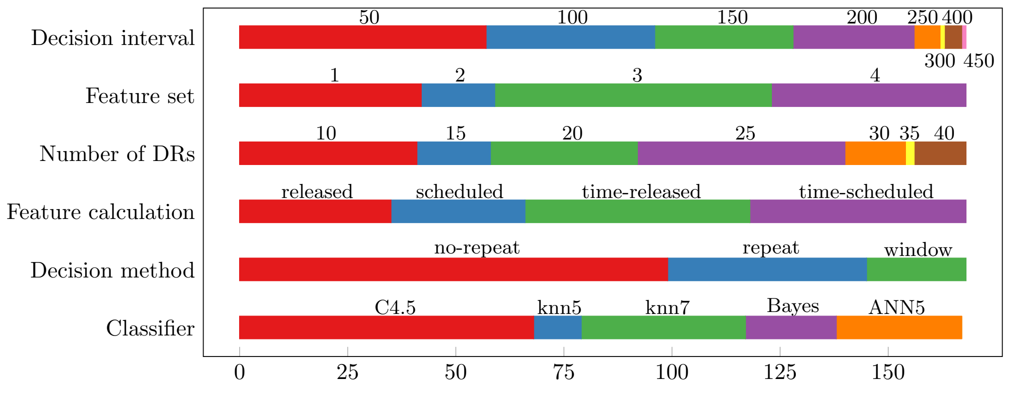
The parameter which defines the number of DRs which are used for generating the learning set also shows to be influenced slightly by the problem type. For the problems with constant characteristics it is evident that the frequency of most parameter values for the number of DRs parameter is similar, with the only exceptions being for the value 30 which appears more often than the others, and for values 15 and 25 which appear less often than the others. For the problems with changing characteristics it is evident that learning sets which were created by using a smaller or medium sized number of DRs more often result in good results. For the feature sets, quite interesting things can be observed from the illustrated frequencies. For the problem type with constant characteristics, no single result used the feature set consisting of only the due date features. Out of the other feature sets the ones which contain the *LD* feature seem to be more informative and thus lead to better results. It seems that using only the due date related features is not



(a) Frequency of parameter values for problems with constant characteristics



(b) Frequency of parameter values for problems with changing due date characteristics



(c) Frequency of parameter values for problems with changing due date and release time characteristics

Figure 1: The frequency of parameter values for the 1% best solutions

445 enough, instead the procedure should also be supplied with other features as well. A similar situation can be observed for the problems with changing due date characteristics. The problem type with the changing due date and release time characteristics shows to prefer feature sets without the *LD* feature, which probably is not informative in such problem types. Finally, for the decision interval it can be seen that for the first two problem types the best results are most often obtained when using the smallest intervals. It is best to perform the procedure as soon as possible to reduce

the effects of the initially selected DR, but also to perform the decision quite often to adapt to any possible changes. For the problems with changing due date and release time characteristics it is evident that even some larger decisions intervals lead to good results, but nevertheless for decision intervals larger than 200, good results are once again quite rarely obtained. The reason that for this problem type some larger intervals can lead to good results could be due to the fact that during a great part of the system execution a very small number of jobs are released, and therefore there is a smaller need to perform the decision frequently during those periods.

6. Results

In this section the performance of the proposed DR selection procedure will be compared to a manually selected DR from the 50 rules which were obtained by GP in the start. The DR which obtained the best results on the training set (the same which is used as the initial DR by the selection procedure) was selected as the reference rule to which the method will be compared.

6.1. Results for the problems with constant characteristics

This section will present the results of the DR selection procedure for several selected parameter value combinations, when applied for solving problems with constant characteristics. Table 3 represents the results obtained for five selected parameter combinations, as well as for the DR which achieved the best overall results on the training set. The first thing which can be noticed is that the worst result is obtained when the ANN classification method is used. In this case the procedure achieved an improvement of only 0.42% on the validation set, however, on the test set it achieved an improvement of 8.1%.

For the other four results, the procedure performs quite well for all of the tested parameters. In three occasions, the procedure performs its decision after a small time period, or after a small number of jobs is released. For all these experiments the procedure achieved good results on both problem instance sets. On the other hand, for the experiment where the decision is performed after 150 jobs are released into the system, the procedure achieved to a certain extent worse results than in other experiments, where a smaller number of released jobs was used. Therefore, the DR selection procedure obtains a good approximation of the characteristics of the problem instance after only a small number of jobs is released. As a consequence, it is preferred to use the DR selection procedure as soon as possible to select which of the available DRs is appropriate for solving the current problem instance. This way the influence of the initially applied DR, which might not be suitable for solving the current problem instance, will be reduced. When using a smaller feature set, the procedure can perform well by using only a smaller number of DRs, as can be seen from last two examples in the table. As the number of features in the feature set increases, the procedure achieves better results as more DRs are used by the procedure. This is demonstrated by the first two examples in the table. The best results for the validation and test set are achieved when the DR selection procedure uses the no-repeat method, which is expected since the characteristics of the scheduling problem do not change over time. By using the repeat or window methods, the procedure achieves to a certain extent worse results, but it still easily achieves better results than by using only a single manually selected DR.

The DR selection procedure achieved an improvement of at most 14.11% on the validation set, and 15.9% on the test set, when compared to the manually selected DR. Therefore, the procedure does not achieve only good improvements on the validation set, on which its parameters were tuned, but also on an unseen problem set. More importantly, the results denoted in the table show that the procedure performs well on both problem instance sets for the same parameter combination. This demonstrates that with a good choice of parameters, the DR selection procedure can perform well on various problem instances, and not only on the one for which the values of the parameters were fine-tuned.

Table 3: Results of the dynamic DR selection procedure for several selected parameter values, when applied on problems with constant characteristics

DR selection procedure parameters	*Twt* value on the validation set	Improvement on the validation set	*Twt* value on the test set	Improvement on the test set
Feature set: 3 Classification method: knn-7 Number of DRs used: 30 Decision interval: released Decision method: no-repeat Number of released jobs: 50	3.444	14.11%	3.667	15.21%
Feature set: 4 Classification method: C4.5 Number of DRs used: 30 Decision interval: interval Decision method: window Interval length: 100	3.452	13.92%	3.656	15.47%
Feature set: 4 Classification method: ANN-5 Number of DRs used: 20 Decision interval: interval Decision method: no-repeat Interval length: 100	3.993	0.42%	3.976	8.07%
Feature set: 2 Classification method: C4.5 Number of DRs used: 10 Decision interval: released Decision method: repeat Number of released jobs: 150	3.526	12.07%	3.783	12.5%
Feature set: 2 Classification method: C4.5 Number of DRs used: 15 Decision interval: released Decision method: no-repeat Number of released jobs: 50	3.456	13.82%	3.639	15.86%
Manually selected DR	4.010	-	4.325	-

6.2. Results for the problems with changing due date characteristics

In this section the performance of the DR selection procedure will be analysed when the procedure is applied for solving problems in which the due date characteristics of jobs change throughout the problem. Table 4 represents the results of the five selected parameter combinations on the validation and test set.

The worst results are achieved by the second experiment, where the smallest feature set, consisting only of the due date related features, was used. Nevertheless, even in this case a good improvement over the reference DR were obtained. For the other parameter combinations denoted in the table the selection procedure obtained even better results on both problem instance sets. The table denotes that the C4.5 classification method was in both cases applied with a small number of DRs in it, while the knn and naive Bayes classification methods used a larger number of DRs in the learning set. Therefore, C4.5 seems to generally perform better when a smaller number of DRs is used for the generation of the learning set. For all experiments the procedure uses a small interval or number of released jobs between subsequent selections of DRs. This again proves that an earlier and more frequent application of the DR selection procedure leads to better results. Most of the presented experiments use the feature set which consists out of all features, which leads to the conclusion that the procedure is more likely to achieve good results if it has access to a larger number of problem characteristics. It is interesting to observe that the best results were in several cases obtained when using the no-repeat decision method, which indicates that the initially selected DR is appropriate for solving the problem instance until the end. The reason for such a behaviour could be due to the fact that the problem characteristics are not changed to an extent which would be large enough so that the selected rule would perform badly. Nevertheless, the two best results for the test set denoted in the table were obtained when using the repeat method, which means that by performing the DR selection several times can nevertheless lead to even slightly better results.

The best result was achieved by the first experiment, for both the validation and test set. With these parameter values the DR selection procedure outperformed the manually selected DR by 14.7% on the validation set, and 16.6% on the test set. When achieving the best results, the DR selection procedure was used with the C4.5 classification method, and with only ten DRs for creating the learning set. Thus, the procedure does not require the use of many DRs to perform well. Based on the results it is evident that the DR selection procedure performs quite well for the selected parameter combinations on both the validation and the test set, even when applied on problems with changing characteristics.

6.3. Results for the problems with changing due date and release time characteristics

In this section, the performance of the DR selection procedure will be analysed on a scheduling problem in which the due date and release time characteristics change during the execution of the system. Table 5 represents the results for the five selected parameter combinations of the DR selection procedure.

The DR selection procedure achieved the worst results when the ANN classification method was used. In this case, the procedure outperformed the manually selected DR by 5.3% on the validation set, and by 3.4% on the test set. Therefore, the ANN classification method is once again unable to achieve equally good performance as the other applied methods. Most of the experiments used around 20 DRs, regardless of the classification method, which seems to be an optimal choice for this problem type. It is interesting to observe that the best results were achieved by experiments which used the window and no-repeat methods. The experiment which used the repeat method achieved inferior results when compared to both of those methods, without considering the experiment when the ANN classification method was used. Thus, it seems that the repeat method is inferior to the other two methods when the release times in the problem also change. In three experiments the procedure used the largest feature set, while in the remaining two the procedure used the feature set consisting of the due date related features and the *LD* feature. Therefore, it seems easier for the

Table 4: Results of the DR selection procedure for several selected parameter values, when applied on problems with changing due date characteristics

DR selection procedure parameters	*Twt* value on the validation set	Improvement on the validation set	*Twt* value on the test set	Improvement on the test set
Feature set: 2 Classification method: C45 Number of DRs used: 10 Decision interval: released Decision method: repeat Number of released jobs: 50	3.425	14.65%	3.353	16.63%
Feature set: 1 Classification method: knn-5 Number of DRs used: 30 Decision interval: released Decision method: no-repeat Number of released jobs: 50	3.559	11.31%	3.673	8.68%
Feature set: 4 Classification method: Bayes Number of DRs used: 25 Decision interval: interval Decision method: no-repeat Interval length: 200	3.556	11.39%	3.440	14.47%
Feature set: 4 Classification method: knn-7 Number of DRs used: 25 Decision interval: interval Decision method: no-repeat Interval length: 100	3.523	12.21%	3.512	12.68%
Feature set: 4 Classification method: C4.5 Number of DRs used: 10 Decision interval: released Decision method: repeat Number of released jobs: 50	3.525	12.16%	3.412	15.17%
Manually selected DR	4.013	-	4.022	-

Table 5: Results of the DR selection procedure for several selected parameter values, when applied on problems with changing due date and release time characteristics

DR selection procedure parameters	*Twt* value on the validation set	Improvement on the validation set	*Twt* value on the test set	Improvement on the test set
Feature set: 4 Classifier: knn-7 Number of DRs used: 25 Decision interval: released Decision method: window Number of released jobs: 50	4.901	12.92%	5.450	10.70%
Feature set: 4 Classifier: C4.5 Number of DRs used: 20 Decision interval: released Decision method: no-repeat Number of released jobs: 150	5.009	11.00%	5.395	11.60%
Feature set: 4 Classifier: C4.5 Number of DRs used: 15 Decision interval: interval Decision method: repeat Interval length: 100	5.177	8.01%	5.545	9.14%
Feature set: 2 Classifier: C4.5 Number of DRs used: 20 Decision interval: interval Decision method: no-repeat Interval length: 100	4.975	11.60%	5.292	13.29%
Feature set: 2 Classifier: ANN-5 Number of DRs used: 20 Decision interval: interval Decision method: no repeat Interval length: 200	5.328	5.33%	5.895	3.41%
Manually selected DR	5.628	-	6.103	-

procedure to obtain good results when using more information about the problem instances. In addition, all feature sets contain the *LD* feature, which means that this feature seems to be useful for this problem type, probably due to the increased load which occurs in the middle of the system. Although for most of the experiments the selection procedure was applied after a short time interval, or after only a small number of jobs were released, the second experiment shows that the procedure achieves good results even if the selection procedure is applied after a larger number of jobs was released.

The largest improvement that the DR selection procedure can achieve over the manually selected DR are 12.9% for the validation set, and 13.3% for the test set. Unfortunately, the procedure was unable to achieve the best results on both problem sets for the same parameter values. On the validation set the best result was achieved by the knn classification method, while the best result on the test set was achieved by the C4.5 classification method. It seems as if the C4.5 method can better extract general knowledge from the learning set and thus perform better on unseen problem instances, whereas knn can adapt to the learning set more easily. As for the previous problem types, the DR selection procedure has again shown to achieve consistently good results on both the validation and test set, given that good parameter values are chosen.

7. Analysis of the rule selection procedure

In this section the DR selection procedure will be analysed to obtain deeper insights about the procedure.

Figure 2 represents the change of the fitness value in relation to the number of the released jobs, when using the DR selection procedure for solving the three problem types with different characteristics. The denoted fitness value represents the total value obtained on all 60 problem instances of the test set. The figure compares the influence of the different decision methods when all the other parameter values are the same. For the problems with constant characteristics it is evident that the best performance is obtained by using the no-repeat and window methods, whereas when using the repeat method the DR selection procedure achieves a similar performance as the referent DR. Even from the start of the system execution, after around 100 jobs are released, it is evident that the referent DR obtains a larger *Twt* value, while all the decision methods achieve a similar performance. However, after around the 300th job is released into the system it can be seen that the repeat method slowly starts to achieve worse results than the other two. Although the *Twt* value is smaller than the one obtained by the referent rule for most of the system execution, it increases faster than the *Twt* value obtained by the referent rule, and thus at the end of the system execution the *Twt* values of both the referent DR and the repeat method are mostly the same. For the problems with changing due date characteristics it can be observed that all three decision methods obtain a similar performance, with the window method achieving slightly worse results. Nevertheless, all three methods obtain a better performance than the referent DR.

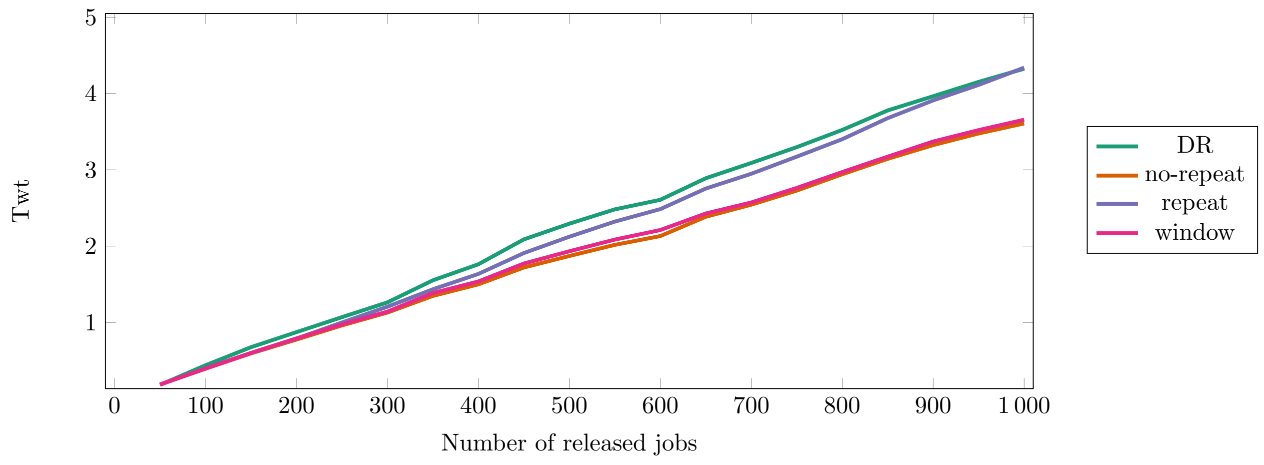
It is interesting to note that around the time when the 350th job is released into the system the *Twt* value obtained by the repeat method starts to increase slowly, until around the 500th job is released, when the criterion values starts to rapidly decrease and again becomes equal to the value obtained by the other two decision methods. This is is probably due to the fact that at that time the due date characteristics of jobs changed and the repeat method required some time to adapt to those changes, since when approximating the characteristics of the problem instance all the jobs from the start of the system execution are used. Finally, for the problem types with changing due date and release time characteristics all the decision methods obtain a better result than the referent DR, with the repeat method obtaining a slightly worse result than the other two. An interesting thing which can be noticed about this problem type is that the *Twt* value starts to rapidly increase after the 350th job is released into the system, and this continues until the 700th job is released.

This is caused by the fact that most of the jobs are released in the middle of the system, which then leads to a lot of jobs being tardy and thus leading to the increase of the *Twt* value. As more jobs are being released into the system the repeat decision method starts to obtain poor *Twt* values and can not keep up with the other two methods, but still unlike for the previous problem type it performs better than the referent DR.

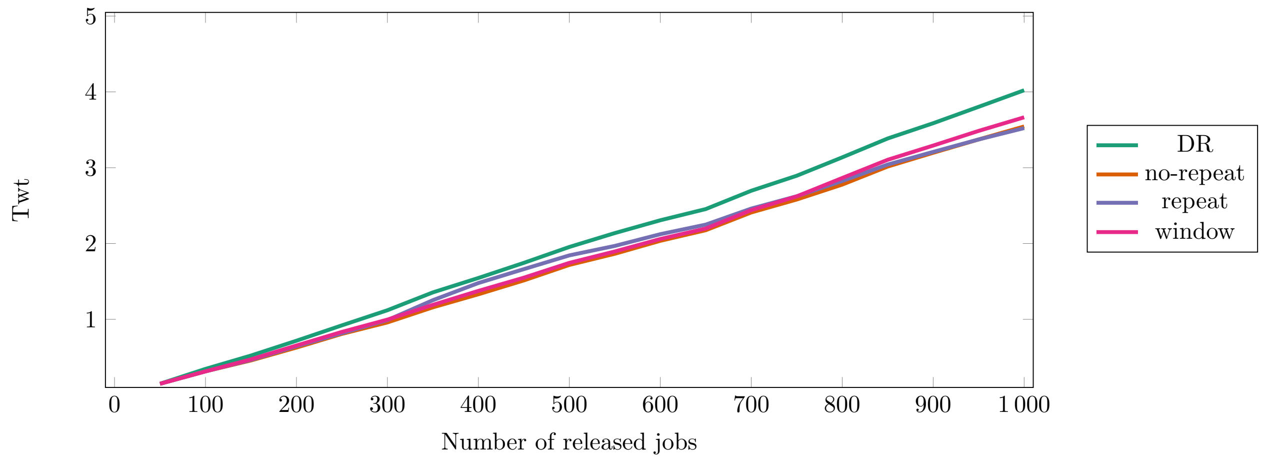
The behaviour of the DR selection procedure will be further illustrated on several selected problem instances with the changing due date and release time characteristics. The parameters for the DR selection procedure will be set to those of the first experiment denoted in Table 5, but all three decision methods of the DR selection procedure will be tested to illustrate the differences in behaviour of the procedure. Table 6 represents the behaviour of the DR selection procedure during the execution of several problem instances. At each decision point, when enough jobs were released, the table shows the index of the DR that is selected for further scheduling, as well as the *Twt* value achieved from the start of the system until that decision point. The starting rule in all procedures is DR 13. In addition, since the no-repeat method performs the decision only once after 50 jobs are released, all subsequent cells will be marked with "-". The results of the manually selected DR are also included in the table to additionally illustrate its behaviour. At the end of the table, the results for all methods on the entire test set are also included.

Problem instance 6 shows that all three methods perform better than the manually selected DR. Even at the start of the problem instance the selected DR performs poorly, while the DR selection procedure changes the rule which is used, and is therefore able to keep the value of *Twt* at zero until 150 jobs are released. It is interesting to note that the no-repeat method achieved better results at the beginning of the schedule. However, as more jobs were released into the system, the other two methods slowly decreased the difference between them and the no-repeat method. This can best be seen the between 400 and 450 released jobs, where the window method achieved the smallest increase in the *Twt* value by switching to rule 5. A similar thing can also be observed between 650 and 700 released jobs. In the end with these changes the window method outperforms the other two methods, while the repeat method slightly outperformed the no-repeat method. On the other hand, on problem instance 8 only the window method achieved a better performance than the manually selected DR. The window method again shows that by performing good rule changes it outperforms all the other results. The no-repeat method, on the other hand, achieved a quite bad result. The reason for this is that it selects a good DR for the start of the schedule, which unfortunately performs poorly as soon as the load of the system increases. Therefore, after around 400 jobs are released, the selected rule starts to perform bad decisions, and the *Twt* value of the system increases.

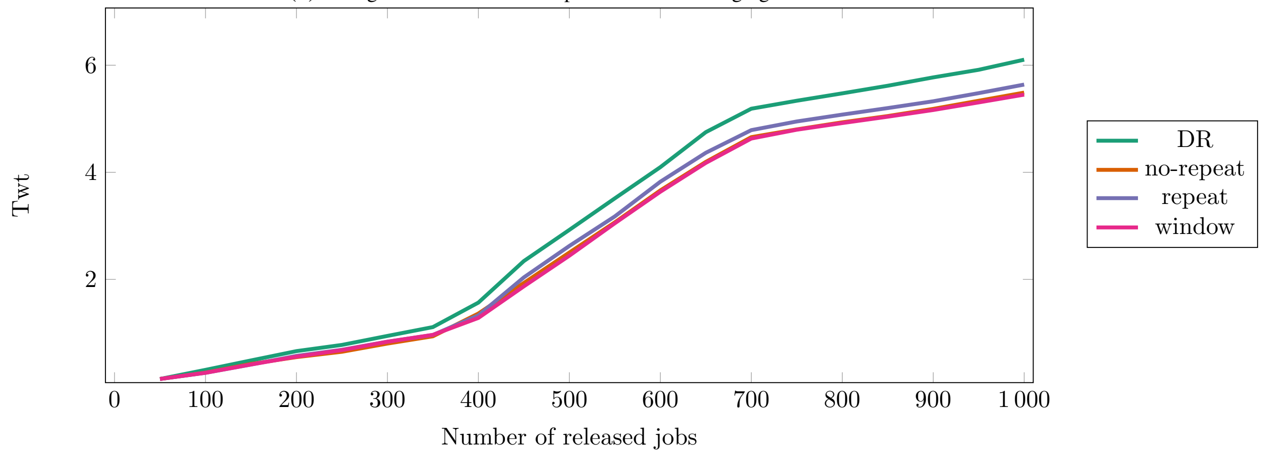
For problem instance 30, all three methods achieved better results than the manually selected DR. Because it made a good decision at the beginning of the schedule, the no-repeat method outperforms the repeat method. The window method again achieved the best results, even though at the end of the schedule it made some bad decisions which resulted in an increase of the *Twt* value. However, at the beginning and the middle of the schedule, the window method performed some good decisions, which allowed it to perform better than the no-repeat method. The manually selected DR makes bad decision even in the beginning of the schedule, because of which the *Twt* of the schedule increases significantly even from the start. For problem instance 37 the no-repeat method achieved the best results among all three methods. The reason for this is that the rule which is first selected performs well for the given problem instance. The window method also uses this rule for scheduling most of the jobs, however, in certain cases it switched to rule 5, which in the end did not lead to good improvements in the results, and even later on had a negative effect when rule 3 was again selected for scheduling. This is best evident in the period between the release of the 600th and 700th job, where although the no-repeat and window methods used the same DR, the window method obtained a slightly higher *Twt* value, as a result of the decisions which were previously performed when rule 5 was used by the procedure. Finally, for problem instance 50, all three methods show to outperform the manually selected DR, however, in this case



(a) Change of the fitness on the problems with constant characteristics



(b) Change of the fitness on the problems with changing due date characteristics



(c) Change of the fitness on the problems with changing due date and release time characteristics

Figure 2: Change of the fitness in relation with the number of released jobs

all three methods achieved similar results. It is interesting to note how the window method uses DR 3 for the most of the schedule, and only in two occasions it switches to other DRs. Although the first switch was unnecessary, since it did not lead to any improvement in the *Twt* value, the second one resulted in a slightly better *Twt* value.

By comparing all methods on the entire problem set, the manually selected DR starts to perform worse than the DR selection procedure immediately from the beginning of the schedule. On the other hand, the DR selection procedure performs much better since it immediately switches the rule it is using. The repeat method achieved the worst result among the three tested methods, which shows that calculating the features from all released jobs might not be beneficial in some cases. Nevertheless, all three methods achieved much better results than the manually selected DR, and thus prove their advantage over selecting and using only a single DR.

In addition to analysing the change of fitness throughout the execution of the procedure, it is also interesting to observe the frequency of the DRs which are selected by the procedure. Figure 3 shows the frequency by which the DRs were selected by the DR selection procedure. Since the frequencies tend to obtain quite large values in some cases, instead of using the total number of selecting a DR the percentage of the times a DR was selected is denoted in the figures. It should be stressed out that only the DRs which appear as labels in the learning set are denoted in the figures. In addition to the frequencies which are obtained by each of the decision methods, the frequencies of the DRs on the learning set are also outlined. For all three learning sets which were constructed for the different problem types it is evident that most of the DRs have a similar appearance frequency. This shows that the proposed GAS procedure can produce a learning set with a balanced distribution of DRs. On the other hand, when the procedure performs the selection on the test set, it can be evident that it is more biased towards a smaller set of DRs, which should be appropriate for solving the considered problem instances. The no-repeat and window methods usually have a similar frequency for the same DRs, which are usually different from those obtained by the repeat method (except for the problems with changing due date characteristics).

Table 6: Dynamic DR selection procedure behaviour analysis

Problem instance	Method		Number of released jobs																			
			50	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800	850	900	950	1000
7	no-repeat	DR index	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		Twt	0	0	0	0.002	0.002	0.004	0.013	0.024	0.051	0.077	0.096	0.126	0.151	0.172	0.176	0.176	0.176	0.176	0.176	0.176
	Repeat	DR index	3	8	2	2	2	18	11	6	9	9	8	8	8	8	8	8	8	8	8	8
		Twt	0	0	0	0.005	0.005	0.008	0.019	0.030	0.050	0.088	0.103	0.127	0.149	0.171	0.174	0.174	0.174	0.174	0.174	0.174
	Window	DR index	3	15	13	1	3	8	3	5	3	3	2	3	3	2	3	3	3	0	3	8
		Twt	0	0	0	0.005	0.005	0.006	0.016	0.028	0.041	0.073	0.095	0.121	0.141	0.155	0.159	0.159	0.159	0.159	0.159	0.159
	Selected DR		0.000	0.008	0.008	0.013	0.013	0.019	0.035	0.046	0.079	0.122	0.137	0.179	0.203	0.219	0.224	0.224	0.224	0.224	0.224	0.224
8	no-repeat	DR index	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		Twt	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.010	0.013	0.022	0.031	0.032	0.039	0.041	0.041	0.041	0.041	0.041	0.041
	Repeat	DR index	1	8	8	3	2	11	11	6	9	9	8	8	8	8	8	8	8	8	8	8
		Twt	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.013	0.016	0.016	0.021	0.028	0.028	0.029	0.029	0.029	0.029	0.029	0.029
	Window	DR index	1	3	3	3	3	3	0	3	3	3	3	3	3	3	0	3	1	3	0	0
		Twt	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.005	0.007	0.012	0.012	0.013	0.013	0.013	0.013	0.013	0.013
	Selected DR		0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.008	0.009	0.009	0.011	0.019	0.020	0.020	0.020	0.020	0.020	0.020	0.020
30	no-repeat	DR index	1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		Twt	0.001	0.007	0.016	0.017	0.022	0.022	0.022	0.022	0.022	0.022	0.023	0.025	0.029	0.032	0.032	0.032	0.032	0.033	0.033	0.033
	Repeat	DR index	1	8	8	2	2	2	8	8	6	6	9	9	8	8	8	8	8	8	8	8
		Twt	0.001	0.007	0.015	0.019	0.019	0.020	0.020	0.020	0.022	0.026	0.028	0.031	0.034	0.035	0.035	0.035	0.035	0.036	0.036	0.037
	Window	DR index	1	1	1	3	1	3	1	3	3	3	3	3	3	3	15	3	1	4	3	3
		Twt	0.001	0.007	0.016	0.017	0.017	0.017	0.017	0.017	0.019	0.019	0.019	0.020	0.024	0.025	0.025	0.025	0.025	0.026	0.027	0.028
	Selected DR		0.001	0.015	0.029	0.033	0.036	0.036	0.036	0.036	0.039	0.041	0.043	0.045	0.047	0.049	0.049	0.049	0.050	0.051	0.051	0.052

Table 6: Dynamic DR selection procedure behaviour analysis

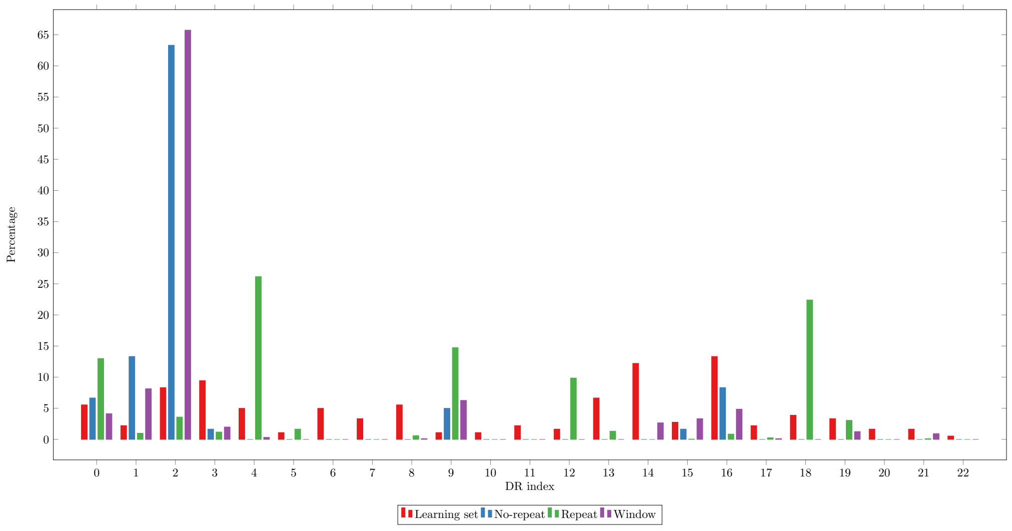
Problem instance	Method		Number of released jobs																			
			50	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800	850	900	950	1000
37	no-repeat	DR index	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		Twt	0.002	0.004	0.006	0.009	0.012	0.015	0.018	0.038	0.055	0.081	0.094	0.109	0.118	0.139	0.140	0.140	0.140	0.140	0.140	0.140
	Repeat	DR index	3	8	2	2	2	2	11	6	9	9	8	8	8	8	8	8	8	8	8	8
		Twt	0.002	0.004	0.006	0.009	0.018	0.023	0.027	0.056	0.084	0.109	0.123	0.146	0.164	0.188	0.189	0.189	0.189	0.189	0.189	0.189
	Window	DR index	3	1	3	3	3	3	3	3	5	3	5	3	3	3	0	3	0	3	3	3
		Twt	0.002	0.004	0.006	0.009	0.012	0.015	0.017	0.038	0.055	0.083	0.114	0.131	0.143	0.164	0.165	0.165	0.165	0.165	0.165	0.165
	Selected DR		0.002	0.005	0.006	0.010	0.012	0.015	0.018	0.042	0.061	0.100	0.119	0.142	0.186	0.212	0.212	0.212	0.212	0.212	0.212	0.212
50	no-repeat	DR index	3	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
		Twt	0.000	0.000	0.000	0.000	0.000	0.003	0.003	0.003	0.005	0.017	0.031	0.032	0.034	0.034	0.038	0.040	0.047	0.052	0.063	0.064
	Repeat	DR index	3	8	8	2	2	11	11	9	9	8	8	8	8	8	8	8	8	8	8	8
		Twt	0.000	0.000	0.000	0.000	0.000	0.005	0.005	0.005	0.007	0.013	0.028	0.030	0.031	0.031	0.034	0.037	0.044	0.049	0.062	0.064
	Window	DR index	3	3	3	1	3	3	3	3	3	3	3	3	3	3	3	2	3	3	3	3
		Twt	0.000	0.000	0.000	0.000	0.000	0.003	0.003	0.003	0.005	0.017	0.030	0.032	0.034	0.034	0.038	0.040	0.046	0.051	0.062	0.063
	Selected DR		0.000	0.000	0.001	0.001	0.005	0.009	0.009	0.009	0.016	0.030	0.043	0.045	0.049	0.055	0.058	0.060	0.067	0.072	0.074	0.076
Test set	no-repeat		0.140	0.259	0.421	0.548	0.648	0.801	0.937	1.362	1.933	2.501	3.065	3.661	4.194	4.656	4.801	4.933	5.050	5.184	5.336	5.488
	Repeat		0.140	0.255	0.409	0.571	0.685	0.837	0.962	1.328	2.037	2.624	3.176	3.823	4.364	4.787	4.949	5.077	5.199	5.326	5.478	5.638
	Window		0.140	0.255	0.410	0.557	0.679	0.835	0.964	1.280	1.870	2.442	3.048	3.636	4.175	4.630	4.795	4.919	5.038	5.163	5.305	5.450
	Selected DR		0.140	0.310	0.486	0.660	0.776	0.944	1.109	1.566	2.342	2.924	3.513	4.095	4.749	5.187	5.336	5.474	5.616	5.772	5.914	6.103

Table 7 shows the performance of the five DRs which are most often selected by the DR selection procedure when considering all three decision methods at the same time. The table clearly shows that for all three problem types the selection procedure mostly selects those DRs which truly achieve a better performance on the test set. Unfortunately, the procedure does not achieve a fitness value which is equal to the best DR in the learning set. This is expected since the selection procedure does not know which DR would perform the best on the problem instance set. Therefore the obtained results demonstrate that the DR selection procedure was in most of the cases able to identify which DRs are appropriate, and apply them for scheduling. Although this will not lead the procedure to obtain equally good results as the best individual DRs, it will nevertheless lead to the situation that the performance of the procedure is similar to the performance of the better individual DRs. Furthermore, it is also evident that the frequency of the DRs on the learning set does not influence the frequency on the test set, which additionally proves that the procedure managed to extract certain knowledge from the learning set and that it does not just apply the DR which was most commonly used on the learning set.

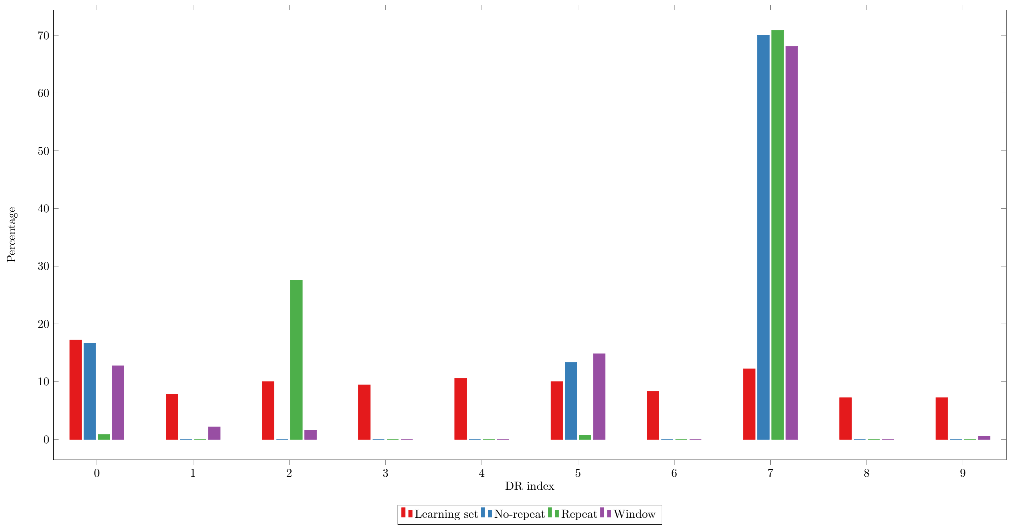
8. Conclusion

Since a single DR cannot perform well on all possible problem instances, it is important to select an appropriate DR to obtain better a performance. However, in dynamic scheduling conditions it is difficult to know in advance which of the available DRs would be best suited for solving the concrete problem instance. This paper proposes a procedure for selecting automatically generated DRs based on the characteristics of the problem instance currently solved. By using the proposed procedure it is possible to adapt to the changing conditions of the system and achieve a better performance than if only a single DR would be used for solving the entire problem. The proposed procedure has also shown to be quite resilient to the different problem types on which it was tested, since it obtained an equally good performance regardless whether it was applied on problems with constant characteristics throughout the system execution, or on problems in which certain characteristics changed during the execution of the system. The only drawback of the procedure is the large number of parameters for which good values need to be selected, since the performance of the procedure depends heavily on the selected values.

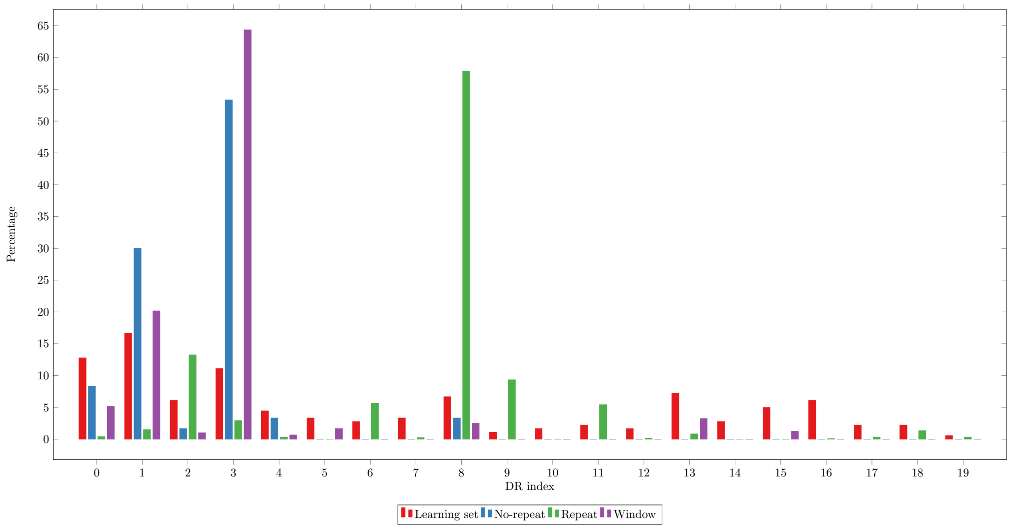
Based on all the previous observations, it can be concluded that the described DR selection procedure represents a viable addition to automatic generation of DRs, since it enables that an appropriate DR is selected for each problem instance. However, even if good results were achieved by this procedure, there are still many open topics which can be researched in the future. One possible research direction is to further examine which characteristics of scheduling problems could be extracted into features that could be used to describe the problem instance. One possibility would be to extract as many features as possible, and then to try out different feature selection and reduction methods to obtain the most informative set of features, which could lead to a better performance of the DR selection procedure. The procedure can also be tested with other classification methods and machine learning methods to determine whether other classification models would be more suitable for this task. An especially interesting topic here would be to use GP to create the classification method for selecting the DRs, and evolve it at the same time when the DRs are evolved. This would allow for the entire scheduling procedure to be designed in only one step. Furthermore, it could possibly allow more flexibility to GP, since it could immediately evolve DRs for problematic instances. In addition, instead of performing the DR selection at fixed time moments, the procedure could also be extended so that it determines when the selection should be performed. This would increase the flexibility and possibly allow it to better adapt to the problems which it is solving.



(a) Frequency of DRs selected on problems with constant characteristics



(b) Frequency of DRs selected on problems with changing due date characteristics



(c) Frequency of DRs selected on problems with changing due date and release time characteristics

Figure 3: Frequency of DR selected by the DR selection procedure

Table 7: Performance of the five most selected DRs by the DR selection procedure

Problem type	DR index	Fitness	Learning set	No-repeat	Repeat	Window
Constant characteristics	0	3.921	10	4	195	62
	2	3.495	15	38	54	985
	4	4.526	9	0	392	5
	9	3.738	2	3	221	94
	7	4.826	7	0	336	0
	DR selection procedure fitness			3.608	4.339	3.656
Changing due date characteristics	0	3.501	31	10	10	153
	1	3.593	14	0	0	26
	2	3.875	18	0	331	19
	5	5.910	18	8	9	178
	7	3.313	22	42	850	817
	DR selection procedure fitness			3.544	3.523	3.665
Changing due date and release time characteristics	1	5.546	30	18	18	242
	2	5.696	11	1	159	12
	3	5.169	20	32	35	772
	8	5.474	12	2	694	30
	9	5.542	2	0	112	0
	DR selection procedure fitness			5.488	5.638	5.450

Funding information

This work has been supported in part by Croatian Science Foundation under the project IP-2019-04-4333.

References

Ahn, G., & Hur, S. (2020). Clustering and dispatching rule selection framework for batch scheduling. *Mathematics*, *8*. URL: https://www.mdpi.com/2227-7390/8/1/80. doi:10.3390/math8010080.

Alpaydin, E. (2014). *Introduction to Machine Learning*. The MIT Press.

Amin, G. R., & El-Bouri, A. (2018). A minimax linear programming model for dispatching rule selection. *Computers & Industrial Engineering*, *121*, 27–35. URL: https://www.sciencedirect.com/science/article/pii/S0360835218302225. doi:https://doi.org/10.1016/j.cie.2018.05.021.

Branke, J., Nguyen, S., Pickardt, C. W., & Zhang, M. (2016). Automated Design of Production Scheduling Heuristics: A Review. *IEEE Transactions on Evolutionary Computation*, *20*, 110–124. URL: http://ieeexplore.ieee.org/document/7101236/. doi:10.1109/TEVC.2015.2429314.

Branke, J., & Pickardt, C. W. (2011). Evolutionary search for difficult problem instances to support the design of job shop dispatching rules. *European Journal of Operational Research*, *212*, 22–32. URL: http://linkinghub.elsevier.com/retrieve/pii/S0377221711000981. doi:10.1016/j.ejor.2011.01.044.

Braun, T. D., Siegel, H. J., Beck, N., Bölöni, L. L., Maheswaran, M., Reuther, A. I., Robertson, J. P., Theys, M. D., Yao, B., Hensgen, D., & Freund, R. F. (2001). A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. *Journal of Parallel and Distributed Computing*, *61*, 810–837. URL: http://linkinghub.elsevier.com/retrieve/pii/S0743731500917143. doi:10.1006/jpdc.2000.1714.

Burke, E. K., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., & Woodward, J. R. (2010). A Classification of Hyper-heuristic Approaches. In *Handbook of Metaheuristics* (pp. 449–468). URL: http://dx.doi.org/10.1007/978-1-4419-1665-5_15. doi:10.1007/978-1-4419-1665-5_15. arXiv:0102188v1.

Burke, E. K., Hyde, M. R., Kendall, G., Ochoa, G., Ozcan, E., & Woodward, J. R. (2009). Exploring Hyper-heuristic Methodologies with Genetic Programming. *Computational Intelligence*, *1*, 177–201. URL: http://www.cs.nott.ac.uk/~gxo/papers/ChapterGPasHH09.pdf. doi:doi:10.1007/978-3-642-01799-5_6.

Chand, S., Huynh, Q., Singh, H., Ray, T., & Wagner, M. (2018). On the use of genetic programming to evolve priority rules for resource constrained project scheduling problems. *Information Sciences*, *432*, 146 – 163. URL: http://www.sciencedirect.com/science/article/pii/S0020025517311350. doi:https://doi.org/10.1016/j.ins.2017.12.013.

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*, 273–297. URL: http://link.springer.com/10.1007/BF00994018. doi:10.1007/BF00994018.

Cox, D. R. (1958). The regression analysis of binary sequences (with discussion). *J Roy Stat Soc B*, *20*, 215–242.

Dimopoulos, C., & Zalzala, A. (1999). A genetic programming heuristic for the one-machine total tardiness problem. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)* (pp. 2207–2214). IEEE. URL: `http://ieeexplore.ieee.org/document/785549/`. doi:`10.1109/CEC.1999.785549`.

Dimopoulos, C., & Zalzala, A. (2001). Investigating the use of genetic programming for a classic one-machine scheduling problem. *Advances in Engineering Software*, *32*, 489–498. URL: `http://linkinghub.elsevier.com/retrieve/pii/S0965997800001095`. doi:`10.1016/S0965-9978(00)00109-5`.

Drozdowski, M. (1996). Scheduling multiprocessor tasks — an overview. *European Journal of Operational Research*, *94*, 215 – 230. doi:`http://dx.doi.org/10.1016/0377-2217(96)00123-3`.

Durasević, M., & Jakobović, D. (2016). Comparison of solution representations for scheduling in the unrelated machines environment. In *2016 39th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)* (pp. 1336–1342). IEEE. URL: `http://ieeexplore.ieee.org/document/7522347/`. doi:`10.1109/MIPRO.2016.7522347`.

Durasević, M., & Jakobović, D. (2017a). Comparison of ensemble learning methods for creating ensembles of dispatching rules for the unrelated machines environment. *Genetic Programming and Evolvable Machines*, . URL: `https://doi.org/10.1007/s10710-017-9302-3`. doi:`10.1007/s10710-017-9302-3`.

Durasević, M., & Jakobović, D. (2017b). Evolving dispatching rules for optimising many-objective criteria in the unrelated machines environment. *Genetic Programming and Evolvable Machines*, . URL: `https://doi.org/10.1007/s10710-017-9310-3`. doi:`10.1007/s10710-017-9310-3`.

El-Bouri, A., & Shah, P. (2006). A neural network for dispatching rule selection in a job shop. *The International Journal of Advanced Manufacturing Technology*, *31*, 342–349. URL: `http://link.springer.com/10.1007/s00170-005-0190-y`. doi:`10.1007/s00170-005-0190-y`.

Ferreira, C. (2001). Gene expression programming: a new adaptive algorithm for solving problems. *Complex Systems*, *13*, 87–129. URL: `http://arxiv.org/abs/cs/0102027`.

Fix, E., & Hodges, J. L. (1951). Discriminatory analysis, nonparametric discrimination: Consistency properties. *US Air Force School of Aviation Medicine*, *Technical Report 4*.

Gil-Gala, F. J., Mencía, C., Sierra, M. R., & Varela, R. (2019). Evolving priority rules for on-line scheduling of jobs on a single machine with variable capacity over time. *Applied Soft Computing*, *85*, 105782. URL: `https://www.sciencedirect.com/science/article/pii/S1568494619305630`. doi:`https://doi.org/10.1016/j.asoc.2019.105782`.

Gil-Gala, F. J., Sierra, M. R., Mencía, C., & Varela, R. (2021). Genetic programming with local search to evolve priority rules for scheduling jobs on a machine with time-varying capacity. *Swarm and Evolutionary Computation*, *66*, 100944. URL: `https://www.sciencedirect.com/science/article/pii/S2210650221001061`. doi:`https://doi.org/10.1016/j.swevo.2021.100944`.

Hart, E., & Sim, K. (2016). A Hyper-Heuristic Ensemble Method for Static Job-Shop Scheduling. *Evolutionary Computation*, *24*, 609–635. URL: `http://www.mitpressjournals.org/doi/10.1162/EVCO_a_00183`. doi:`10.1162/EVCO\_a\_00183`.

Heger, J., Hildebrandt, T., & Scholz-Reiter, B. (2015). Dispatching rule selection with Gaussian processes. *Central European Journal of Operations Research*, *23*, 235–249. URL: http://link.springer.com/10.1007/s10100-013-0322-7. doi:10.1007/s10100-013-0322-7.

Jaklinović, K., Ðurasević, M., & Jakobović, D. (2021). Designing dispatching rules with genetic programming for the unrelated machines environment with constraints. *Expert Systems with Applications*, *172*, 114548. URL: https://www.sciencedirect.com/science/article/pii/S0957417420311921. doi:https://doi.org/10.1016/j.eswa.2020.114548.

Jakobović, D., & Budin, L. (2006). Dynamic scheduling with genetic programming. In P. Collet, M. Tomassini, M. Ebner, S. Gustafson, & A. Ekárt (Eds.), *Genetic Programming: 9th European Conference, EuroGP 2006, Budapest, Hungary, April 10-12, 2006. Proceedings* (pp. 73–84). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: https://doi.org/10.1007/11729976_7. doi:10.1007/11729976_7.

Jakobović, D., & Marasović, K. (2012). Evolving priority scheduling heuristics with genetic programming. *Applied Soft Computing*, *12*, 2781–2789. URL: http://linkinghub.elsevier.com/retrieve/pii/S1568494612001780. doi:10.1016/j.asoc.2012.03.065.

Kahraman, C., Engin, O., Kaya, I., & Ozturk, R. E. (2010). Multiprocessor task scheduling in multistage hybrid flowshops: A parallel greedy algorithm approach. *Applied Soft Computing*, *10*, 1293 – 1300. doi:http://dx.doi.org/10.1016/j.asoc.2010.03.008.

Karunakaran, D., Chen, G., & Zhang, M. (2016). Parallel Multi-objective Job Shop Scheduling Using Genetic Programming. In T. Ray, R. Sarker, & X. Li (Eds.), *Artificial Life and Computational Intelligence: Second Australasian Conference, ACALCI 2016, Canberra, ACT, Australia, February 2-5, 2016, Proceedings* (pp. 234–245). Springer International Publishing. URL: http://link.springer.com/10.1007/978-3-319-28270-1_20. doi:10.1007/978-3-319-28270-1_20.

Keijzer, M., & Babovic, V. (1999). Dimensionally Aware Genetic Programming. *Proceedings of the Genetic and Evolutionary Computation Conference*, *2*, 1069–1076. URL: http://www.cs.bham.ac.uk/~wbl/biblio/gecco1999/GP-420.pdf.

Koza, J. R. (2010). Human-competitive results produced by genetic programming. *Genetic Programming and Evolvable Machines*, *11*, 251–284. URL: http://link.springer.com/10.1007/s10710-010-9112-3. doi:10.1007/s10710-010-9112-3.

Lian Yu, Shih, H., & Sekiguchi, T. (1998). Dynamic selection of dispatching rules by fuzzy inference. In *1998 IEEE International Conference on Fuzzy Systems Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98CH36228)* (pp. 979–984). IEEE volume 2. URL: http://ieeexplore.ieee.org/document/686251/. doi:10.1109/FUZZY.1998.686251.

Liu, H., & Dong, J. J. (1996). Dispatching rule selection using artificial neural networks for dynamic planning and scheduling. *Journal of Intelligent Manufacturing*, *7*, 243–250. URL: http://link.springer.com/10.1007/BF00118083. doi:10.1007/BF00118083.

Maheswaran, M., Ali, S., Siegel, H. J., Hensgen, D., & Freund, R. F. (1999). Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems. *Journal of Parallel and Distributed Computing*, *59*, 107–131.

URL: http://linkinghub.elsevier.com/retrieve/pii/S0743731599915812. doi:10.1006/jpdc.1999.1581.

Miyashita, K. (2000). Job-shop scheduling with genetic programming. In *Proceedings of the 2Nd Annual Conference on Genetic and Evolutionary Computation* GECCO'00 (pp. 505–512). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc. URL: http://dl.acm.org/citation.cfm?id=2933718.2933809.

Mouelhi-Chibani, W., & Pierreval, H. (2010). Training a neural network to select dispatching rules in real time. *Computers & Industrial Engineering*, *58*, 249–256. URL: http://linkinghub.elsevier.com/retrieve/pii/S0360835209000953. doi:10.1016/j.cie.2009.03.008.

Nguyen, S., Mei, Y., & Zhang, M. (2017). Genetic programming for production scheduling: a survey with a unified framework. *Complex & Intelligent Systems*, *3*, 41–66. URL: http://link.springer.com/10.1007/s40747-017-0036-x. doi:10.1007/s40747-017-0036-x.

Nguyen, S., Zhang, M., & Johnston, M. (2014). A sequential genetic programming method to learn forward construction heuristics for order acceptance and scheduling. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1824–1831). IEEE. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6900347. doi:10.1109/CEC.2014.6900347.

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2013a). A Computational Study of Representations in Genetic Programming to Evolve Dispatching Rules for the Job Shop Scheduling Problem. *IEEE Transactions on Evolutionary Computation*, *17*, 621–639. URL: http://ieeexplore.ieee.org/document/6353198/. doi:10.1109/TEVC.2012.2227326.

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2013b). Learning iterative dispatching rules for job shop scheduling with genetic programming. *The International Journal of Advanced Manufacturing Technology*, *67*, 85–100. URL: http://link.springer.com/10.1007/s00170-013-4756-9. doi:10.1007/s00170-013-4756-9.

Nguyen, S., Zhang, M., Johnston, M., & Tan, K. C. (2013c). Learning Reusable Initial Solutions for Multi-objective Order Acceptance and Scheduling Problems with Genetic Programming. In K. Krawiec, A. Moraglio, T. Hu, A. Ş. Etaner-Uyar, & B. Hu (Eds.), *Genetic Programming: 16th European Conference, EuroGP 2013, Vienna, Austria, April 3-5, 2013. Proceedings* (pp. 157–168). Berlin, Heidelberg: Springer Berlin Heidelberg. URL: http://link.springer.com/10.1007/978-3-642-37207-0_14. doi:10.1007/978-3-642-37207-0_14.

Nguyen, S., Zhang, M., & Tan, K. C. (2015a). A Dispatching rule based Genetic Algorithm for Order Acceptance and Scheduling. In *Proceedings of the 2015 on Genetic and Evolutionary Computation Conference - GECCO '15* (pp. 433–440). New York, New York, USA: ACM Press. URL: http://dl.acm.org/citation.cfm?doid=2739480.2754821. doi:10.1145/2739480.2754821.

Nguyen, S., Zhang, M., & Tan, K. C. (2015b). Enhancing genetic programming based hyper-heuristics for dynamic multi-objective job shop scheduling problems. In *2015 IEEE Congress on Evolutionary Computation (CEC)* (pp. 2781–2788). IEEE. URL: http://ieeexplore.ieee.org/document/7257234/. doi:10.1109/CEC.2015.7257234.

Nie, L., Gao, L., Li, P., & Zhang, L. (2011). Application of gene expression programming on dynamic job shop scheduling problem. In *Proceedings of the 2011 15th International Conference on Computer Supported Cooperative*

Work in Design (CSCWD) (pp. 291–295). IEEE. URL: http://ieeexplore.ieee.org/document/5960088/. doi:10.1109/CSCWD.2011.5960088.

Nie, L., Shao, X., Gao, L., & Li, W. (2010). Evolving scheduling rules with gene expression programming for dynamic single-machine scheduling problems. *The International Journal of Advanced Manufacturing Technology*, *50*, 729–747. URL: http://link.springer.com/10.1007/s00170-010-2518-5. doi:10.1007/s00170-010-2518-5.

Olafsson, S., & Li, X. (2010). Learning effective new single machine dispatching rules from optimal scheduling data. *International Journal of Production Economics*, *128*, 118–126. URL: http://linkinghub.elsevier.com/retrieve/pii/S0925527310002124. doi:10.1016/j.ijpe.2010.06.004.

Park, J., Mei, Y., Nguyen, S., Chen, G., & Zhang, M. (2018). An investigation of ensemble combination schemes for genetic programming based hyper-heuristic approaches to dynamic job shop scheduling. *Applied Soft Computing*, *63*, 72 – 86. URL: http://www.sciencedirect.com/science/article/pii/S156849461730683X. doi:https://doi.org/10.1016/j.asoc.2017.11.020.

Park, J., Nguyen, S., Zhang, M., & Johnston, M. (2015). Evolving ensembles of dispatching rules using genetic programming for job shop scheduling. In *Genetic Programming: 18th European Conference, EuroGP 2015, Copenhagen, Denmark, April 8-10, 2015* (pp. 92–104). Cham: Springer International Publishing. URL: https://doi.org/10.1007/978-3-319-16501-1_8. doi:10.1007/978-3-319-16501-1_8.

Pickardt, C. W., Hildebrandt, T., Branke, J., Heger, J., & Scholz-Reiter, B. (2013). Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems. *International Journal of Production Economics*, *145*, 67–77. URL: http://linkinghub.elsevier.com/retrieve/pii/S0925527312004574. doi:10.1016/j.ijpe.2012.10.016.

Pierreval, H. (1992). Expert system for selecting priority rules in flexible manufacturing systems. *Expert Systems with Applications*, *5*, 51–57. URL: http://linkinghub.elsevier.com/retrieve/pii/0957417492900949. doi:10.1016/0957-4174(92)90094-9.

Pierreval, H. (1993). Neural Network to Select Dynamic Scheduling Heuristics. *Journal of Decision Systems*, *2*, 173–190. URL: http://www.tandfonline.com/doi/abs/10.1080/12460125.1993.10511572. doi:10.1080/12460125.1993.10511572.

Pierreval, H., & Mebarki, N. (1997). Dynamic scheduling selection of dispatching rules for manufacturing system. *International Journal of Production Research*, *35*, 1575–1591. URL: http://www.tandfonline.com/doi/abs/10.1080/002075497195137. doi:10.1080/002075497195137.

Pinedo, M. L. (2012). *Scheduling: Theory, algorithms, and systems: Fourth edition* volume 9781461423614. Boston, MA: Springer US. URL: http://link.springer.com/10.1007/978-1-4614-2361-4. doi:10.1007/978-1-4614-2361-4. arXiv:arXiv:1011.1669v3.

Poli, R., & McPhee, N. F. (2013). Parsimony pressure made easy: Solving the problem of bloat in GP. In *Theory and Principled Methods for the Design of Metaheuristics* (pp. 181–204). Springer Berlin Heidelberg. URL: https://doi.org/10.1007/978-3-642-33206-7_9. doi:10.1007/978-3-642-33206-7_9.

Priore, P., DE LA FUENTE, D., GOMEZ, A., & PUENTE, J. (2001). A review of machine learning in dynamic scheduling of flexible manufacturing systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, *15*, 251–263.

Priore, P., Gómez, A., Pino, R., & Rosillo, R. (2014). Dynamic scheduling of manufacturing systems using machine learning: An updated review. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*, *28*, 83–97. URL: `http://www.scopus.com/inward/record.url?eid=2-s2.0-84896442633&partnerID=40&md5=c6d5b3ad421a09adf167cf12f86aa134`. doi:`10.1017/S0890060413000516`.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Shahzad, A., & Mebarki, N. (2016). Learning dispatching rules for scheduling: A synergistic view comprising decision trees, tabu search and simulation. *Computers*, *5*. URL: `http://www.mdpi.com/2073-431X/5/1/3`. doi:`10.3390/computers5010003`.

Shiue, Y.-R., & Guh, R.-S. (2006). Learning-based multi-pass adaptive scheduling for a dynamic manufacturing cell environment. *Robotics and Computer-Integrated Manufacturing*, *22*, 203–216. URL: `http://linkinghub.elsevier.com/retrieve/pii/S0736584505000402`. doi:`10.1016/j.rcim.2005.03.004`.

Souza, C. (2018).

Subramaniam, V., Lee, G. K., Hong, G. S., Wong, Y. S., & Ramesh, T. (2000a). Dynamic selection of dispatching rules for job shop scheduling. *Production Planning & Control*, *11*, 73–81. URL: `http://www.tandfonline.com/doi/abs/10.1080/095372800232504`. doi:`10.1080/095372800232504`.

Subramaniam, V., Ramesh, T., Lee, G. K., Wong, Y. S., & Hong, G. S. (2000b). Job Shop Scheduling with Dynamic Fuzzy Selection of Dispatching Rules. *The International Journal of Advanced Manufacturing Technology*, *16*, 759–764. URL: `http://link.springer.com/10.1007/s001700070029`. doi:`10.1007/s001700070029`.

Sun, Y.-L., & Yih, Y. (1996). An intelligent controller for manufacturing cells. *International Journal of Production Research*, *34*, 2353–2373. URL: `http://www.tandfonline.com/doi/abs/10.1080/00207549608905029`. doi:`10.1080/00207549608905029`.

Tay, J. C., & Ho, N. B. (2008). Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems. *Computers & Industrial Engineering*, *54*, 453–473. URL: `http://linkinghub.elsevier.com/retrieve/pii/S0360835207002008`. doi:`10.1016/j.cie.2007.08.008`.

Đumić, M., Šišejković, D., Čorić, R., & Jakobović, D. (2018). Evolving priority rules for resource constrained project scheduling problem with genetic programming. *Future Generation Computer Systems*, *86*, 211 – 221. URL: `http://www.sciencedirect.com/science/article/pii/S0167739X1732441X`. doi:`https://doi.org/10.1016/j.future.2018.04.029`.

Đurasević, M., & Jakobović, D. (2020). Comparison of schedule generation schemes for designing dispatching rules with genetic programming in the unrelated machines environment. *Applied Soft Computing*, *96*, 106637. URL: `http://www.sciencedirect.com/science/article/pii/S1568494620305755`. doi:`https://doi.org/10.1016/j.asoc.2020.106637`.

Đurasević, M., & Jakobović, D. (2019). Creating dispatching rules by simple ensemble combination. *Journal of Heuristics*, *25*, 959–1013. URL: https://doi.org/10.1007%2Fs10732-019-09416-x. doi:10.1007/s10732-019-09416-x.

Đurasević, M., & Jakobović, D. (2020). Automatic design of dispatching rules for static scheduling conditions. *Neural Computing and Applications*, . URL: https://doi.org/10.1007/s00521-020-05292-w. doi:10.1007/s00521-020-05292-w.

Đurasević, M., Jakobović, D., & Knežević, K. (2016). Adaptive scheduling on unrelated machines with genetic programming. *Applied Soft Computing*, *48*, 419–430. URL: http://linkinghub.elsevier.com/retrieve/pii/S1568494616303519. doi:10.1016/j.asoc.2016.07.025.

Weckman, G. R., Ganduri, C. V., & Koonce, D. A. (2008). A neural network job-shop scheduler. *Journal of Intelligent Manufacturing*, *19*, 191–201. URL: http://link.springer.com/10.1007/s10845-008-0073-9. doi:10.1007/s10845-008-0073-9.

Werbos, P. J. (1974). *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. Ph.D. thesis Harvard University.

Zahmani, M. H., & Atmani, B. (2019). A data mining based dispatching rules selection system for the job shop scheduling problem. *Journal of Advanced Manufacturing Systems*, *18*, 35–56. URL: https://doi.org/10.1142/S0219686719500021. doi:10.1142/S0219686719500021. arXiv:https://doi.org/10.1142/S0219686719500021.

Zahmani, M. H., Atmani, B., Bekrar, A., & Aissani, N. (2015). Multiple priority dispatching rules for the job shop scheduling problem. In *2015 3rd International Conference on Control, Engineering & Information Technology (CEIT)* (pp. 1–6). IEEE. URL: http://ieeexplore.ieee.org/document/7232991/. doi:10.1109/CEIT.2015.7232991.

Zhang, F., Mei, Y., Nguyen, S., Tan, K. C., & Zhang, M. (2021a). Multitask genetic programming-based generative hyperheuristics: A case study in dynamic scheduling. *IEEE Transactions on Cybernetics*, (pp. 1–14). doi:10.1109/TCYB.2021.3065340.

Zhang, F., Mei, Y., Nguyen, S., & Zhang, M. (2021b). Correlation coefficient-based recombinative guidance for genetic programming hyperheuristics in dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, *25*, 552–566. doi:10.1109/TEVC.2021.3056143.

Zhang, F., Mei, Y., Nguyen, S., Zhang, M., & Tan, K. C. (2021c). Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling. *IEEE Transactions on Evolutionary Computation*, *25*, 651–665. doi:10.1109/TEVC.2021.3065707.

Zhang, H., & Roy, U. (2018). A semantics-based dispatching rule selection approach for job shop scheduling, . *30*, 2759–2779. URL: https://doi.org/10.1007/s10845-018-1421-z. doi:10.1007/s10845-018-1421-z.