A survey of dispatching rules for the dynamic unrelated machines environment

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

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

Abstract

In the real world, scheduling is usually performed under dynamic conditions, which means that it is not known when new jobs will be released into the system. Therefore, the procedure which is used to create the schedule must be able to adapt to the changing conditions during the execution of the system. In dynamic conditions, dispatching rules are one of the most commonly used methods for creating the schedules. Throughout the years, various dispatching rules were defined for a wide range of scheduling criteria. However, in most cases when a new dispatching rule is proposed, it is usually tested on only one or two scheduling criteria, and compared with only a few other dispatching rules. Furthermore, there are also no recent studies which compare all the different dispatching rules with each other. Therefore, it is difficult to determine how certain dispatching rules perform on different scheduling criteria and problem types. The objective of this study was to collect a large number of dispatching rules from the literature for the unrelated machines environment, and test them on nine scheduling criteria and four problem types with various machine and job heterogeneities. For each of the tested dispatching rules it will be outlined in which situations it achieves the best results, as well as which dispatching rules are best suited for solving each of the tested scheduling criteria.

Keywords: Dispatching rules, unrelated machines environment, dynamic

---

[*] Corresponding author

[1] marko.durasevic@fer.hr

[2] domagoj.jakobovic@fer.hr

Preprint submitted to Expert Systems with Applications *July 13, 2018* conditions, release times

1. Introduction

Scheduling is a decision-making process which deals with the allocation of resources to tasks over a given period of time (Leung, 2004; Pinedo, 2012). The goal of the scheduling process is to create a schedule which optimises one or more user defined criteria. Scheduling plays an important role in most manufacturing systems (Dimopoulos & Zalzala, 2000; Kofler et al., 2009), but is also used in many other real world scenarios, like scheduling planes on runways (Cheng et al., 1999; Hansen, 2004), scheduling for radiotherapy pre-treatment (Petrovic & Castro, 2011), scheduling tasks on CPUs (Pinedo, 2012), scheduling in railway traffic (Corman & Quaglietta, 2015), and many others. Because of its wide applicability, as well as its complexity, various scheduling problems have been studied in the last several decades.

Most instances of scheduling problems belong to the category of NP-hard problems, which makes it impossible to obtain an optimal solution in a reasonable amount of time. Because of this reason, scheduling problems are in most cases solved by using different heuristic methods, which obtain a satisfactory solution in a relatively small amount of time. Although many heuristic methods have been specifically designed for solving the unrelated machines environment (Fanjul-Peyro & Ruiz, 2010, 2011; Cota et al., 2014; de C. M. Nogueira et al., 2014), in most cases scheduling problems are solved by using different metaheuristic methods (like genetic algorithms, particle swarm optimisation, tabu search, and many other) (Hart et al., 2005). All the aforementioned methods can be applied only on scheduling problems under static conditions, where the information about all jobs is known before the execution of the system, and thus the schedule can be created beforehand. However, many scheduling problems occur in dynamic scheduling environments, in which it is not known in advance when jobs will arrive into the system, and what their properties will be. Therefore, it is not possible to create a schedule up front, but rather the schedule needs to be constructed simultaneously with the execution of the system. In order to solve scheduling problems under dynamic conditions, many simple scheduling methods, called *dispatching rules*, have been defined in the literature.

Dispatching rules (DRs) are simple constructive scheduling heuristics, which iteratively build up a schedule. This is done in a way that each time a certain machine is free, the DR determines which of the available, but yet unscheduled jobs, should be scheduled on the given machine. In order to determine which job should be scheduled next, DRs most commonly use a priority function to rank the jobs, and schedule the job with the best priority value. The priorities of jobs are usually calculated based on some characteristics of the jobs and the current state of the system. Therefore, DRs can be used under dynamic scheduling conditions, since they will only use the currently available information to decide which jobs should be scheduled next. Because DRs construct the schedule iteratively, they achieve much better execution times than metaheuristic methods (Đurasević & Jakobović, 2016), and can thus react quickly to changes which happen in the scheduling environment. However, designing good DRs is usually a lengthy trial and error process, which needs to be performed by domain experts. To tackle this problem, different machine learning and evolutionary computation methods have been used to automatically design new DRs (Branke et al., 2016; Nguyen et al., 2017). Although automatically designed DRs usually achieve better performance than manually designed DRs, they are also more complex and not as interpretable. Additionally, manually designed DRs are often used as a baseline for evaluating the performance of automatically designed DRs. Because of all these reasons, it is still important to design new and improved DRs, and also to be aware of how the various manually designed DRs perform on different scheduling criteria.

Although a wide range of DRs have been defined for the unrelated machines environment, very little research was performed to compare the performance of all the proposed DRs, and test how they perform for different scheduling criteria. Maheswaran et al. (1999) compared eight DRs for minimising the makespan criterion. In the paper the selected DRs were applied in a dynamic environment where jobs were released during the execution of the system. Braun et al. (2001) analysed the performance of six DRs in the static scheduling environment for minimising the makespan criterion. They additionally compared the considered DRs with five other methods which can be applied for solving static scheduling problems, like genetic algorithms and similar search based heuristic methods. Du Kim & Kim (2004) propose a new DR for minimising the makespan and compare it with three existing DRs for scheduling in the unrelated machines environment under dynamic conditions. Izakian et al. (2009) compared six DRs for scheduling tasks in heterogeneous distributed environments. The authors compared the results achieved by the different DRs when optimising the mean makespan and mean flowtime criteria. Pfund et al. (2008) compared several DRs for the unrelated parallel machines with setup and ready times, for optimising the total weighted tardiness criterion. However, only the static scheduling environment was considered in the previous study. Yang-Kuei & Chi-Wei (2013) have considered scheduling in the unrelated machines environment with release times, however, once again only for the static scheduling conditions. They compared several DRs for optimising three criteria independently, the makespan, total completion time, and total weighted tardiness. Tseng et al. (2009) compare six DRs for optimising the makespan and total weighted tardiness criteria when scheduling jobs in the heterogeneous computing environment. The DRs were applied in a dynamic scheduling environment where jobs were released into the system during the execution of the system.

The aim of this paper is to provide an overview of DRs which can be applied for solving the unrelated machines scheduling problem with release times. The considered scheduling problem will be solved under dynamic conditions, meaning that the schedule needs to be constructed simultaneously with the execution of the system. To collect most of the proposed DRs which would be applicable for solving such scheduling problems, an extensive survey of the existing literature on DRs for the unrelated machine environment was conducted. In addition, a new DR called *just in time* and a new version of the *work queue* DR are proposed in this paper, both of which were designed manually. All the DRs will be tested on nine scheduling criteria to give a notion on how the collected DRs perform on various scheduling objectives. Furthermore, the DRs will be tested on four different problem sets, each of which will be generated with a different machine and job heterogeneity, to analyse how the selected DRs perform on different problem configurations. Based on all the conducted experiments, the paper will draw conclusions on which of the tested DRs were the most appropriate for optimising the tested criteria, as well as how different methods compare to each other. This should allow for an easier selection of appropriate DRs for a given criterion and heterogeneity conditions.

The rest of the paper is organised as follows. Section 2 gives an overview of the unrelated machines environment and the objectives which will be used to measure the performance of the created schedules. The DRs which were selected from the literature are enumerated and described in Section 3. The design of the experiments is described in Section 4. Section 5 outlines the results achieved by all the selected DRs on the nine scheduling criteria. Section 6 gives a discussion about the main conclusion which can be drawn from the obtained results. Finally, Section 7 gives the conclusion of this survey and outlines some possible future research directions.

2. Unrelated machines environment

The unrelated machines environment consists of n jobs which need to be scheduled on one of the m available machines. It is presumed that both the number of machines and jobs are finite. Each job can be scheduled on only a single machine, and once it starts with its execution it can not be interrupted until it is completed. Additionally, each machine can execute one job at a time. The index j is usually used to denote a concrete job, while the index i is used to denote a concrete machine. For each job and machine pair a processing time p_{ij} is defined, which determines the amount of time needed for machine i to execute job j. Each job also has a release time r_j which determines when the job becomes available and is released into the system, a due date d_j which determines the time until a job should finish with its execution or otherwise a certain penalty will be invoked, and a weight w_j which determines the importance of the job. In this paper three job weights will be used, based on the criterion which is optimised: tardiness weight (w_{T_j}), earliness weight (w_{E_j}), and completion time weight (w_{C_j}). All three weights can have different values for a single job. Scheduling in the unrelated machines environment can be found in many practical real world examples, such as in: multiprocessor computers, landing lanes in airports, operating rooms in hospitals, circuit board manufacturing, semiconductor manufacturing, group technology cells, painting and plastic industries, injection moulding process and remanufacturing, railway rescheduling (Fanjul-Peyro & Ruiz, 2012; Lee et al., 2013; Wang et al., 2013; Quaglietta et al., 2016).

After the schedule is constructed, several metrics can be calculated for each job. These metrics will later on be used to calculate the values of the different scheduling criteria. The following metrics are most commonly used (Leung, 2004; Pinedo, 2012):

- **Completion time of a job** (C_j) - the moment in time at which job j finishes with its execution and exits the system.

- **Flowtime of a job** (F_j) - the amount of time that job j spent in the system:
$$F_j = C_j - r_j. \tag{1}$$

- **Tardiness of a job** (T_j) - the amount of time that job j spent executing after its due date:
$$T_j = \max\{C_j - d_j, 0\}. \tag{2}$$

- **Earliness of a job** (E_j) - the amount of time that job j finished prior to its due date:
$$E_j = \max\{-(C_j - d_j), 0\}. \tag{3}$$

- **Unit penalty** (U_j) - a flag denoting whether a job is tardy or not:

$$U_j = \begin{cases} 1 : T_j > 0 \\ 0 : T_j = 0 \end{cases}. \tag{4}$$

The following nine scheduling objectives will be used in order to evaluate the quality of the schedules created by the tested DRs (Allahverdi et al., 1999; Leung, 2004; Allahverdi et al., 2008; Pinedo, 2012; Durasević & Jakobović, 2017):

- **Makespan** (C_{max}) - denotes the completion time of the last job that leaves the system:

$$C_{max} = \max_j \{C_j\}. \tag{5}$$

- **Maximum flowtime** (F_{max}) - denotes the maximum flowtime achieved by any of the jobs:

$$F_{max} = \max_j \{F_j\}. \tag{6}$$

- **Maximum tardiness** (T_{max}) - denotes the maximum tardiness achieved by any of the jobs:

$$T_{max} = \max_j \{T_j\}. \tag{7}$$

- **Total weighted completion time** (Cw) - denotes the weighted sum of all completion times:

$$Cw = \sum_j w_{Cj} C_j, \tag{8}$$

- **Total weighted tardiness** (Twt) - denotes the weighted sum of tardiness values of all jobs:

$$Twt = \sum_j w_{Tj} T_j, \tag{9}$$

- **Total flowtime** (Ft) - denotes the sum of flowtimes of all jobs:

$$Ft = \sum_j F_j, \tag{10}$$

- **Weighted number of tardy jobs** (Nwt) - denotes the weighted sum of all tardy jobs:

$$Nwt = \sum_j w_{Tj} U_j. \tag{11}$$

- **Weighted earliness and weighted tardiness** ($Etwt$) - denotes the sum of the total weighted tardiness and the total weighted earliness:

$$Etwt = \sum_j (w_{Ej} E_j + w_{Tj} T_j), \tag{12}$$

- **Machine utilisation** (M_{ut}) - denotes the difference between the maximum utilisation and minimum utilisation of all machines:

$$M_{ut} = \max_i \left(\frac{P_i}{C_{max}} \right) - \min_i \left(\frac{P_i}{C_{max}} \right), \tag{13}$$

  where P_i is defined as the sum of processing times of all jobs which were executed on machine with index i.

By using the standard notation of scheduling problems, the problem studied in this paper can be defined as $Rm|r_j|\gamma$, where γ represents one of the nine previously defined criteria. Additionally, scheduling will be performed under dynamic conditions, which means that during scheduling it will not be known when the next job enters the system, neither which will be the characteristics of that job. Once the job enters the system, all its characteristics become available. Therefore, during the execution of the system, the DRs are applied at each decision point to determine which of the released jobs should be scheduled next.

3. Dispatching rules for the unrelated machines environment

This section will describe various dispatching rules for solving the unrelated machines scheduling problem with release times and under dynamic conditions, which were collected from the literature. The dispatching rules are applied to determine which job should be scheduled next on which machine each time a job enters the system and there is at least one machine free, or a machine becomes free and there is at least one job waiting to be scheduled. The priority values calculated by DRs for scheduling job j on machine i will be denoted as π_{ij}. It should be noted that the priority values for some rules are calculated based only on job properties, and will therefore be the same for all machines. The following 26 DRs will be tested:

- **Minimum completion time** (MCT) (Maheswaran et al., 1999; Braun et al., 2001) - jobs are selected in provisional order and the priorities of the selected job on all machines are calculated as

  $$\pi_{ij} = \frac{1}{\max(mr_i, time) + p_{ij}},$$

  where mr_i represents the time when machine i becomes available, and $time$ represents the current time of the system. In this way jobs will be scheduled on the machine on which they will be completed the soonest.

- **Minimum execution time** (MET) (Maheswaran et al., 1999; Braun et al., 2001) - determines the priorities of jobs as

  $$\pi_{ij} = \frac{1}{p_{ij}}.$$

  Therefore, jobs will be scheduled depending only on their processing times, so that each job is scheduled on the machine on which it achieves its minimum processing time. This can naturally lead to situations in which a great amount of jobs is waiting to be processed on a single machine, while the other machines are free. In order to avoid this, jobs will be selected by their processing time, but executed on a machine on which they achieve their minimum completion time.

- **Earliest release date** (ERD) (Pinedo, 2012) - determines the priorities of jobs as

  $$\pi_{ij} = \frac{1}{r_j}.$$

  This means that jobs will be scheduled in order by which they became available. The job with the highest priority will be scheduled on the machine on which it achieves its minimum completion time.

- **Longest processing time** (LPT) (Pinedo, 2012) - determines the priorities of jobs as

  $$\pi_{ij} = p_{ij}.$$

  Jobs with the longest processing time will therefore be selected first and scheduled on the machine on which they achieve their minimum completion time.

- **Weighted shortest processing time** (WSPT) (Lee et al., 1997) - calculates the priorities as

  $$\pi_{ij} = \frac{w_{C_j}}{p_{ij}}.$$

  This rule functions similarly as the MET rule, however, it additionally considers weights which can be defined for jobs. The job with the largest priority value is selected and scheduled on the machine on which it achieves its minimum completion time.

- **Maximum standard deviation** (Maxstd) (Munir et al., 2008) - calculates the standard deviations of processing times for each job, and schedules the one with the highest standard deviation. The selected job is scheduled on the machine on which it achieves its minimum completion time. The intuition behind this rule is to prioritise those jobs which have a high variation of their processing times on different machines, since they will have a larger influence on the makespan if scheduled on an inappropriate machine.

- **Switching algorithm** (SA) (Maheswaran et al., 1999) - uses both the MET and MCT rules in a cyclic fashion depending on the load distribution of the system. The motivation behind this heuristic lies in the fact that the MET rule can create imbalance in the load of the machines by assigning most of the jobs to only a small subset of machines. The MCT rule, on the other hand, tries to even out the load balance across all the machines. Therefore the SA heuristic uses both rules to keep a good balance across all machines, but also to assign jobs to those machines on which they have the smallest processing times. The heuristic uses the *load balance index* to determine when the algorithm should switch from one rule to the other. The index is calculated as

$$\nabla = \frac{mr_{min}}{mr_{max}},$$

where mr_{min} denotes the earliest machine ready time, and mr_{max} the latest machine ready of all machines in the system. Additionally, two threshold values are also defined: ∇_l and ∇_h. The SA heuristic starts to schedule tasks by using the MCT rule until the load balance index reaches a value of at least ∇_h, when it switches to the MET rule. This will cause the load balance index to decrease over time until it decreases to a value of ∇_l or less, when the SA heuristic switches again back to the MCT rule.

- **k-percent best** (KPB) (Maheswaran et al., 1999) - considers only a certain subset of machines when scheduling a job. The subset of machines is constructed by selecting the $m * (k/100)$ machines on which the job j achieves the smallest processing times. The job is assigned to a machine from the selected subset on which it achieves the minimum completion time. The purpose of this heuristic is to schedule jobs on machines for which they have the smallest processing times. In this way the rules tries to prevent them from being scheduled on other machines which could be more suitable for other jobs which arrive into the system.

- **Ordered minimum completion time** (OMCT) (e Santos & Madureira, 2014) - represents an extension of the MCT rule in which the priorities of the jobs are calculated as

$$\pi_{ij} = \alpha * \sigma + (1 - \alpha) * S,$$

where σ represents the standard deviation of all processing times of job j, $\alpha \in [0, 1]$ a control parameter, and S the *sufferage* value which is defined as the difference between the second smallest completion time and the smallest completion time of job j. The job with the highest priority is scheduled on the machine on which it achieves its smallest completion time. By using the standard deviation and sufferage values, this rule tries to determine for which jobs it would be more damaging if they were not scheduled on their preferred machine, and gives them a larger priority value.

- **Opportunistic load balancing** (OLB) (Braun et al., 2001) - schedules a job on the next available machine, regardless of the expected execution time or completion time of that job. The intuition behind this rule is to evenly distribute the load on all machines. Unfortunately, since this rule does not consider the execution times of jobs, it can create schedules with poor results for the makespan criterion. This can be improved to a certain degree so that if several machines are free at the same time, the job is scheduled on the machine on which it achieves its smallest execution time.

- **Earliest due date** (EDD) (Pfund et al., 2008; Pinedo, 2012) - calculates the priories of jobs as

  $$\pi_{ij} = \frac{1}{d_j}.$$

  The reasoning behind this rule is to schedule the job with the earliest due date, to minimise the tardiness of jobs. The job with the largest priority value is scheduled on the machine on which it achieves its minimum completion time.

- **Minimum slack** (MS) (Pinedo, 2012) - calculates the priorities of jobs as

  $$\pi_{ij} = \max\left(d_j - p_{ij} - time, 0\right)$$

  . In this rule the job with the smallest priority is selected and scheduled on the machine on which it achieves its minimum completion time. The rule tries to first schedule those jobs which are already late or close to being late.

- **Montagne's heuristic** (MON) (Morton & Pentico, 1993) - calculates the priorities of jobs as
$$\pi_{ij} = \frac{w_{T_j}}{p_{ij}} * \left(1 - \frac{d_j}{p_s}\right),$$
where p_s represents the sum of processing times of all available jobs for machine i. The rule then schedules the job which achieved the highest priority value to the machine on which it achieves its minimum completion time. This rule tries to scale the WSPT rule with an additional slack factor prioritise to jobs which have an earlier due date. A disadvantage of this rule is that the slack factor is not dynamic, but rather constant during the system execution.

- **Weigthed critical ratio** (CR) (Morton & Pentico, 1993) - calculates the priorities of jobs as
$$\pi_{ij} = \frac{w_{T_j}}{p_{ij}} \left(\frac{1}{1 + \frac{(d_j - p_{ij} - time)}{\bar{p}}}\right),$$
where $\bar{p}$ represents the average processing time of all jobs waiting to be scheduled. The job with the highest priority is scheduled on the machine on which it achieves its minimum completion time. This rule extends the WSPT rule with a dynamic slack factor, by which it prioritises jobs which are close to their due dates. The disadvantage of this rule is that if the job is late, the priority continues to grow. In this survey the CR rule will be used without the weight, since this variant achieved better results.

- **Cost over time** (COVERT) (Morton & Rachamadugu, 1982; Morton & Pentico, 1993) - calculates the priorities of jobs as
$$\pi_{ij} = \frac{w_{T_j}}{p_{ij}} \max\left[\left(1 - \frac{\max\left(d_j - p_{ij} - time, 0\right)}{k\bar{p}}\right), 0\right],$$
where k represents a scaling parameter. The job with the highest priority is scheduled on the machine on which it achieves its minimum completion time. This rule is similar to the CR rule, however it does not allow that the priority of jobs increases the more they are late.

- **Apparent tardiness cost** (ATC) (Vepsalainen & Morton, 1987; Lee et al., 1997; Pfund et al., 2008; Yang-Kuei & Chi-Wei, 2013) - calculates the priorities of jobs as

  $$\pi_{ij} = \frac{w_{T_j}}{p_{ij}} \exp\left[-\frac{\max\left(d_j - p_{ij} - time, 0\right)}{k\bar{p}}\right].$$

  The job with the highest priority value is selected and scheduled on the machine on which it achieves its minimum completion time. The rule can be considered a combination of the WSPT and MS rules, and the scaling factor is used to determine which of these rules will have more influence in the ATC rule.

- **Min-min** (Maheswaran et al., 1999; Braun et al., 2001; Tseng et al., 2009) - calculates the completion time of each available job on all the machines. After that, for each job the machine for which the job achieves its minimum completion time is determined. The job with the overall smallest completion time is selected and scheduled on the machine on which it achieves its minimum completion time. Algorithm 1 represents the min-min rule.

Algorithm 1 Min-min rule

```
while unscheduled jobs are available do
    for each unscheduled job $j$ do
        for each machine $i$ do
            Calculate the completion time $c_{ij}$ for job j and machine i
        end for
    end for
    For each job determine the machine on which it achieves its minimum completion time
    Select the job which achieves the overall minimum completion time
    Schedule the selected job to the machine on which it achieves its minimum completion time
end while
```

- **Max-min** (Maheswaran et al., 1999; Braun et al., 2001) - for each job the rule determines the machine for which the corresponding job achieves its minimum completion time. However, unlike the min-min rule, the max-min rule selects the job with the largest minimum completion time. In that way the max-min rule will prioritise jobs with the longer executing times.

- **Min-max** (Izakian et al., 2009) - for each job the rule determines the machine for which the corresponding job achieves its minimum completion time. The job whose minimum processing time divided by the processing time on the selected machine in the previous step has the maximum value will be scheduled on the selected machine. The intuition behind this rule is to schedule the job whose processing time on the selected machine is the closest to the shortest processing time of that job.

- **Sufferage** (Maheswaran et al., 1999) - for each job the rule determines the machine for which the corresponding job achieves its minimum completion time. The rule then determines the sufferage value for each job. The job with the largest sufferage value is scheduled on the machine for which it achieves its minimum completion time. The intuition behind this heuristic is to schedule the job which would "suffer" the most if not scheduled on the machine with its minimum completion time.

- **Sufferage2** (Rafsanjani & Bardsiri, 2012) - for each job the rule determines the machine for which the corresponding job achieves its minimum completion time. The rule calculates the sufferage value for each job, but additionally scales this value with the following factor

  $$\frac{\min_i p_{ij}}{\min_i ct_{ij}},$$

  where ct_{ij} denotes the completion time of job j on machine i. With this scaling factor the rule also incorporates the information about the processing and completion times when selecting the job to be scheduled.

The job with the largest scaled sufferage value is selected and scheduled on the machine for which the job achieves its minimum completion time.

- **Relative cost** (RC) (Xhafa et al., 2007) - for each job this rule determines the machine on which the job achieves its minimum completion time. Then for each job it calculates two parameters, namely the *static relative cost* and *dynamic relative cost*. The static relative cost for job j and machine i is calculated as
$$\gamma_{ij}^{s} = \frac{p_{ij}}{\frac{\sum_{k \in machines} p_{kj}}{m}},$$
while the dynamic relative cost is calculated as
$$\gamma_{ij}^{d} = \frac{ct_{ij}}{\frac{\sum_{k \in machines} ct_{kj}}{m}}.$$
The total priority of a job is calculated as
$$\pi_{ij} = \frac{1}{(\gamma_{ij}^{s})^{(\alpha)} * \gamma_{ij}^{d}},$$
where α represents a user defined scaling factor. The selected job is scheduled on the machine on which it achieves its minimum completion time. This rule tries to balance between the jobs minimum processing time and minimum completion time, and select the one which has smaller values for both.

- **Longest job to shortest resource - shortest job to fastest resource** (LJFR-SJFR) (Izakian et al., 2009) - for each job the rule determines the machine for which the corresponding job achieves its minimum completion time. In the first step this rule schedules m jobs with the longest minimum completion times to the fastest machines. After this first step the rule alternatively schedules the job with the shortest minimum execution time to the fastest machine, and then the job with the longest minimum execution time to the fastest machine.

- **Minimum execution completion time** (MECT) (Du Kim & Kim, 2004) - represents a combination between the MET and MCT dispatching rules. Algorithm 2 represents the outline of MECT. The DR first determines the maximum ready time of all machines mr_{max}. Afterwards, the rule determines the machines on which job j can finish with its execution prior to mr_{max}. If such machines exist, the one for which job j achieves the minimum execution time is selected. However, if such machines do not exist, the machine on which job j achieves its minimum completion time is selected. Out of all unscheduled jobs, the job which achieves the minimum completion time on the selected machine will be scheduled. The intuition behind MECT is to alternatively use the minimum execution and completion times, in order to perform the scheduling decision. The rule will use the minimum execution time to select the machine on which job j should be executed, if this will not lead to the increase of the makespan. However, if there is no decision which does not increase the makespan, then the machine for which job j achieves its minimum completion time is selected.

- **Work queue** (WQ) (Izakian et al., 2009) - is in the literature defined in a quite similar way as the OLB rule. Therefore, in this study a variant of the WQ rule is proposed and used. This variant selects the machine that has the least workload, i.e. the machine which up to now spent the least time processing jobs. After the machine is selected, the job which achieves the minimum completion time on the selected machine is scheduled on it. The motivation behind this rule is to evenly distribute the work over all machines.

- **Just in time** (JIT) - is a DR which is proposed in this study. This rule tries to schedule the jobs as closely to their due dates as possible. To achieve this, the rule calculates the earliness or tardiness of the job and multiplies it with the corresponding job weight. Therefore, the priority value for each job is calculated as

$$\pi_{ij} = \begin{cases} w_{T_j} * (d_j - p_{ij} - time)^2, & \text{if job } j \text{ is late} \\ w_{E_j} * (d_j - p_{ij} - time)^2, & \text{if job } j \text{ is early} \end{cases}.$$

Algorithm 2 MECT rule

```
while unscheduled jobs are available do
    Let $mr_i$ denote the ready time of machine $i$
    $mr_{max} = \max_i(mr_i)$
    Let $ct_{ij}$ represent the completion time of job $j$ on machine $i$
    for each unscheduled job $j$ do
        Let $M'$ represent all machines for which $p_{ij} + mr_i < mr_{max}$
        Let $sm_j$ represent the selected machine for job $j$
        if $|M'| > 0$ then
            $sm_j = \arg\min_{i \in M'} p_{ij}$
        else
            $sm_j = \arg\min_{i \in M} ct_{ij}$
        end if
    end for
    Schedule the job with the smallest value of $ct_{sm_j j}$
end while
```

The job with the smallest priority value is selected and scheduled on the machine on which it achieves its minimum completion time.

4. Experimental design and setup

Since for the evaluation of the selected problem instances it is necessary to have a wide range of different problems, this section will describe the way in which the problem instances, which were used for evaluation purposes, were generated.

For the generation of processing times, two parameters need to be defined, ϕ_j which is a measure that determines the job heterogeneity, while ϕ_m defines the measure of machine heterogeneity. The ϕ_j parameter controls whether the different jobs will have similar or vastly different processing times. On the other hand, the ϕ_m parameter controls whether a single jobs will have similar or vastly different processing times on the different machines. For each job j a random number μ_j is generated by using an uniform distribution from the interval $[1, \phi_j]$. The corresponding processing times for job j are then generated in a way that for each machine i a random number is sampled from the uniform distribution between $[1, \phi_m]$, and is multiplied by μ_j. Based on previous studies (Tseng et al., 2009), in this paper four parameter combinations will be used: $\phi_j = 3000$ and $\phi_m = 100$ for high job and high machine heterogeneity, $\phi_j = 3000$ and $\phi_m = 10$ for high job and low machine heterogeneity, $\phi_j = 100$ and $\phi_m = 100$ for low job and high machine heterogeneity, $\phi_j = 100$ and $\phi_m = 10$ for low job and low machine heterogeneity.

The release times of the jobs were generated by using a uniform distribution from the interval

$$r_j \in \left[0, \frac{\hat{p}}{2}\right],$$

where $\hat{p}$ represents the expected duration of the schedule that is defined as

$$\hat{p} = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m} p_{ij}}{m^2}.$$

This means that all jobs are expected to be released during the first half of the system execution, which results with a system that has a higher load. The reason why such problem instances were used lies in the fact that for problems with lower load most of the DRs achieved a very similar value for the makespan criterion, which would make it impossible to evaluate the DRs for that criterion.

The due dates of jobs were also generated using a uniform distribution from the interval

$$d_j \in \left[r_j + (\hat{p} - r_j) * \left(1 - T - \frac{R}{2}\right), r_j + (\hat{p} - r_j) * \left(1 - T + \frac{R}{2}\right)\right].$$

The parameter T represents the due date tightness which adjusts the percentage of late jobs, while the parameter R represents the due date range which adjusts the dispersion of due dates. Both parameters assumed values of 0.2, 0.4, 0.6, 0.8 and 1 in various combinations.

Finally, the weights of jobs are generated by using the uniform distribution from the interval $< 0, 1]$. Each one of the three defined weights, w_{E_j}, w_{T_j}, and w_{C_j}, are generated independently from each other.

By using the previously defined expressions, four problem instance sets, one for each of the previously mentioned machine and job heterogeneity parameter value combinations, were generated. Each problem set consists of 60 independently generated problem instances, where each instance consists of 10 machines and 1000 jobs. However, each problem instance is generated by using different values for the due date parameters. The generated problem instances with the best until now known solutions can be obtained from the project site [3].

To test the performance of all the individual DRs a simple simulator was designed. Based on the defined problem instance and DR the simulator will simulate how the DR would be used to construct the entire schedule for the given problem. In each discrete time moment the simulator checks whether it needs to invoke the DR to update the schedule, or if it will simulate that some work is performed on the machines and will move to the next moment in time. If at the current time moment a job is released into the system and there is at least one available machine, or if a machine becomes available and there is at least

---

[3] `http://gp.zemris.fer.hr/scheduling/problemsets.7z`

one job waiting to be scheduled, then the simulator will invoke the DR. Such a moment in time will be denoted as a *decision point*. In all other time moments, the simulator will simply move to the next time moment to simulate that work is being performed on the machines. When a DR is invoked by the simulator, it will consider only those jobs which are released but yet unscheduled, and will posses no knowledge about any of the future jobs that will arrive in the system. Additionally, it is possible that at a certain decision point the DR determines that a job should be scheduled on a machine which is already executing a job. This situation occurs since the priorities of jobs are calculated for all machines, whether they are free or not. If it happens that a job should be scheduled on a machine which is already taken, the scheduling of this job is postponed to a later moment in time. This allows the DRs to insert idle times into the schedule, and not to schedule a job as soon as a machine becomes available. Furthermore, if at any moment a tie between two machines occurs during the execution of the DR, then the machine with a smaller ID is selected (all machines have an ID associated to them, and are always evaluated in the order of their IDs). It should also be mentioned that at a single decision point more than one job can be scheduled, if there are enough available machines.

In addition to the performance of the individual DRs, the *estimated lower bound* (ELB) for each criterion will also be denoted for the four problem sets. The ELB values for a problem set are calculated by summing up the best solution obtained by any of the DRs and several genetic algorithm executions for each problem instance. Although these values do not represent the optimal solutions which can be obtained, they still provide a general idea about the absolute performance of the DRs.

Another important thing which has to be outlined are the execution times of the individual DRs. The time required to calculate the priority value for a single job can be considered almost negligible. However, since at a single decision point the number of jobs and machines based on which the updated schedule needs to be determined can be vast, the time required to determine the updated schedule can also increase substantially. Therefore, the time required to update the schedule depends heavily on the number of unscheduled jobs and available machines at the current decision point. However, to provide a general overview of the execution times for the individual DRs, the time required to update the schedule in a single decision point was measured for all DRs. The decision point was modelled in a way that there are 1000 unscheduled jobs and 10 available machines. In order to calculate the updated schedule in such a situation with a relatively large amount of jobs, all DRs required between 0.02 and 0.7 seconds, except for OLB which executed for only 0.005 seconds due to its simplicity. Based on the measured execution times it is evident that the DRs can calculate the updated schedule in a relatively small amount of time even in decision points with a large number of unscheduled jobs and available machines. This allows for the DRs to be used in dynamic environments in which it is required to quickly perform the scheduling decision.

Finally, it needs to be mentioned that the parameters for each of the DRs were fine tuned on an independent problem instance set, and that the values for which the best results were achieved were selected. The ATC rule was executed with $k = 0.05$, the RC rule with $\alpha = 0.2$, the SA rule with $\nabla_l = 0.1$ and $\nabla_h = 0.8$, the OMCT rule with $\alpha = 0.9$, the COVERT rule with $k = 0.2$, while the other rules do not use any parameters.

5. Results

This section will outline the results which were obtained by the selected DRs for the nine scheduling objectives. Each result of the DRs denoted in the tables represents the sum of the results for the 60 problem instances used to evaluate the DRs. In each table the best result for each criterion will be denoted in bold, while the best five results for each criterion will be denoted with a grey cell. The table includes three additional columns which denote the average rank of each DR on several sets of criteria. The column denoted as $Rank_t$ represents the average rank of the DR on the set consisting of three due date related criteria (Nwt, T_{max}, and Twt). On the other hand, the column denoted as $Rank_{cf}$ represents the average rank of the rules on the set consisting of completion time and flowtime related criteria (C_{max}, C_w, F_{max}, and Fw). These two groups of criteria were selected since the performance of the DRs seems to be relatively correlated for the criteria within each group. Finally, the column denoted as *Rank* represents the average rank for each DR across all the optimised scheduling criteria.

Table 1 represents the results achieved by the selected DRs for the problem instance set which was generated by using high job and high machine heterogeneity. The results demonstrate that DRs which achieve a good performance on the C_{max} criterion also achieve a good performance on the F_{max} criterion as well. This is well evident since the top five DRs are the same for both criteria. The best overall results for both criteria were achieved by the Sufferage2 DR. For the Cw and Ft criteria it can also be observed that if a DR achieves a good result on one criterion, it also achieves a good value on the other. Although WSPT achieved the best result for the Cw criterion, it was unable to achieve a good performance for the Ft criterion, for which the min-min DR achieved the best result, followed closely by the KPB and MECT DRs. It is interesting to note that for the $Etwt$ and M_{ut} the best results are mostly achieved by DRs which do not perform well on other scheduling criteria. For $Etwt$ the best result was achieved by the JIT DR, whereas for the M_{ut} criterion the best result is achieved by the LPT DR. For the remaining three due date related criteria (Nwt, T_{max}, and Twt), the ATC, MON, and COVERT DRs achieve a good performance on all three criteria. However, neither DR achieves the best result for all three criteria. What is surprising is that for the T_{max} criterion the best result is achieved by the proposed JIT DR, almost two times better than the second best result achieved by the ATC rule.

The average rank for the due date related criteria shows that the ATC rule achieves the overall best performance on the aforementioned set of criteria. The rule achieved the best result only for the Twt criterion, and performed well for the other two criteria. The COVERT rule obtained only a slightly lower average rank, but nevertheless performed well across all the due date related criteria.

Table 1: Results for the test set generated with a high job and a high machine heterogeneity

	C_{max}	Cw	F_{max}	Ft	$Etwt$	M_{ut}	Nwt	T_{max}	Twt	$Rank_t$	$Rank_{cf}$	$Rank$
ELB	11.73	2268	11.67	2565	970.3	$4.15*10^{-10}$	15.61	2.189	380.3	-	-	-
MCT	15.39	6980	15.27	6858	13402	$4.84*10^{-7}$	22.88	10.092	2346.1	20	20	18.2
MET	13.30	2990	13.18	2862	14482	$7.75*10^{-7}$	18.42	7.527	889.24	10	8.3	11.6
ERD	14.76	7177	14.50	7045	13327	$3.50*10^{-7}$	23.08	9.696	2408.7	20	20.5	17.9
LPT	13.80	9194	13.77	9061	12947	**$1.023*10^{-8}$**	25.53	9.749	3221.5	23.7	21	17.7
WSPT	13.74	**2610**	13.61	3603	14281	$5.44*10^{-7}$	19.21	8.498	1158.6	13.3	10.3	13
Maxstd	14.07	8532	14.03	8398	13106	$1.98*10^{-8}$	24.72	9.945	2968.3	23	18.2	18.4
SA	13.36	3534	13.23	3405	14259	$7.58*10^{-7}$	19.06	7.506	1049.7	10.3	9.3	11.8
KPB	12.35	2740	12.23	2611	14555	$8.80*10^{-7}$	18.14	6.862	800.72	5	4.8	9.1
OMCT	14.02	8505	13.98	8371	13105	$2.01*10^{-8}$	24.73	9.886	2953.9	22.3	21	17.6
OLB	12.82	2856	12.71	2726	14517	$8.67*10^{-7}$	18.28	6.964	839.56	8	7.3	10.9
EDD	13.82	3860	13.70	3728	14240	$7.31*10^{-7}$	19.36	8.077	1205.1	12.7	14.5	14.7
MS	14.79	7136	14.55	7004	13366	$4.04*10^{-7}$	23.07	9.645	2407.9	19	20.5	17.8
MON	13.71	3739	13.58	3608	13873	$5.10*10^{-7}$	17.99	5.325	754.99	3.3	12	9.3
CR	14.01	4851	13.88	4714	13884	$6.97*10^{-7}$	20.47	8.300	1517.1	15	16.5	15.3
COVERT	13.67	3725	13.54	3593	13880	$5.14*10^{-7}$	**17.98**	5.315	752.17	2	10.3	**8.4**
ATC	13.72	3727	13.61	3595	13878	$5.16*10^{-7}$	17.99	5.306	**751.60**	**1.7**	11.8	9
Min-min	12.35	2739	12.22	**2610**	14556	$8.63*10^{-7}$	18.14	6.833	800.85	5.3	**3.8**	8.7
Max-min	15.88	7784	15.63	7649	13221	$2.76*10^{-7}$	23.82	10.621	2659.7	22.3	23	19.2
Min-max	11.87	6751	11.77	6624	13293	$2.25*10^{-7}$	22.54	7.797	2182.8	15	8.5	10.4
Sufferage	12.14	7007	12.03	6874	13278	$1.46*10^{-7}$	22.83	8.148	2294.9	16.7	10.5	11.6
Sufferage2	**11.74**	8787	**11.70**	8658	12825	$6.76*10^{-8}$	24.86	8.078	2964.1	20.7	12.5	13.1
RC	12.13	4038	11.99	3907	14096	$5.57*10^{-7}$	19.58	7.097	1219.8	12	8	11.3
LJFR-SJFR	12.56	2790	12.44	2661	14541	$7.80*10^{-7}$	18.18	7.054	818.84	7.7	6.3	10.3
MECT	12.34	2740	12.21	2612	14556	$8.82*10^{-7}$	18.15	6.815	801.66	5.7	4.3	9.4
WQ	60.73	29994	60.60	29852	22060	$1.73*10^{-7}$	44.33	51.021	18188	26	26	23.8
JIT	13.81	7225	13.70	7092	**12301**	$3.19*10^{-7}$	20.90	**2.904**	1559.6	10.3	18.5	12.8

On the other hand, the min-min rule obtained the best average rank for the completion time and flowtime related criteria. This rule is followed closely by the MECT and KPB rules which obtained a slightly lower average rank than the min-min rule. By considering the average rank on all the criteria, the best rank is achieved by the COVERT rule, meaning that it performed relatively well on a wide range of criteria. Although the KPB rule did not achieve the best result for any of the criteria, it still belongs to the five best DRs based on their rank, since it also performs well for most of the criteria. The MECT rule also performs well for all criteria, which can be seen from the fact that for five criteria the rule achieves results which are among the top five results. Unfortunately, for the $Etwt$ and M_{ut} criterion the rule achieved among the worst results, which consequentially led to the deterioration of the rule's rank. From the results in the table it is evident that the DRs with the best ranks can be divided into two groups. The first group consists of rules which perform well on all criteria except for the $Etwt$ and M_{ut} criteria (like KPB, MECT, and min-min). The second group consists of those rules which perform well only on two or three criteria, while on the others they achieve moderate results (like COVERT, MON, and ATC).

Table 2 represents the results achieved for the problem set generated with a high job and a low machine heterogeneity. By examining the table it is evident that the DRs perform quite similar as they did for the problem set with high machine and job heterogeneity. For example, for the C_{max}, Cw, and F_{max} criteria the top five rules are the same for both problem sets. For the C_{max} criterion the best result was achieved by the Sufferage2 rule, while for Cw the best result was obtained by WSPT, which is the same as for the previous problem set. However, for the F_{max} criterion the best result was achieved by the RC rule, followed closely by the Sufferage2 rule. The MECT rule achieved the best overall result for the Ft criterion. For the $Etwt$ criterion the best result was achieved by WQ, which did not achieve good results for $Etwt$ on the previous set. On the other hand, for the M_{ut} criterion the LPT DR once again achieved the best result. The remaining three due date related criteria are solved best

Table 2: Results for the test set generated with a high job and a low machine heterogeneity

	C_{max}	Cw	F_{max}	Ft	$Etwt$	M_{ut}	Nwt	T_{max}	Twt	$Rank_t$	$Rank_{cf}$	$Rank$
ELB	16.37	4468	16.08	3854	1326	$4.36*10^{-9}$	2.988	1.118	23.98	-	-	-
MCT	20.26	9229	19.69	8034.4	22840	$3.53*10^{-6}$	8.213	7.498	855.2	18.7	19.8	18.2
MET	17.94	5230	17.28	4041.4	25818	$5.19*10^{-6}$	4.935	5.587	343.0	10	8.8	12.1
ERD	19.23	9472	16.86	8285.0	22528	$2.40*10^{-6}$	8.253	6.674	824.0	17.3	15.5	15.0
LPT	18.21	12132	18.11	10927	20818	$\mathbf{1.63*10^{-7}}$	10.89	7.509	1282	24.3	20.5	17.6
WSPT	18.64	**4535**	18.08	5261.7	24915	$3.25*10^{-6}$	5.933	6.426	497.7	12.7	10.5	12.4
Maxstd	18.53	11603	18.41	10408	21230	$2.09*10^{-7}$	10.50	7.682	1229	23.3	20.8	17.9
SA	17.96	5531	17.21	4344.3	25544	5.0¸5*10^{-6}	5.137	5.526	354.0	10.3	9.8	12.2
KPB	17.11	5065	16.53	3875.7	25936	$5.39*10^{-6}$	4.767	5.186	319.0	6	5.3	9.7
OMCT	18.52	11623	18.40	10425	21223	$2.05*10^{-7}$	10.54	7.683	1234	24.3	20.8	18.0
OLB	17.72	5287	17.04	4097.4	25758	$5.474*10^{-6}$	4.927	5.320	341.3	8.7	8.3	11.7
EDD	19.18	7935	17.14	6749.1	23746	$4.16*10^{-6}$	7.247	6.180	666.5	14	14.5	14.8
MS	19.28	9475	16.94	8288.0	22528	$2.52*10^{-6}$	8.270	6.688	825.6	18.3	16.5	15.8
MON	18.70	6460	18.10	5268.2	24696	$3.52*10^{-6}$	**4.391**	3.338	269.1	2	14.3	10.7
CR	19.34	8466	17.41	7287.3	23365	$3.90*10^{-6}$	7.671	6.478	742.4	16	16.5	15.4
COVERT	18.70	6459	18.10	5269.6	24696	$3.50*10^{-6}$	4.406	3.342	269.5	3.33	14.3	10.9
ATC	18.64	6447	18.09	5257.6	24702	$3.49*10^{-6}$	4.401	3.304	**266.9**	**1.7**	12.3	9.6
Min-min	17.10	5065	16.50	3875.4	25936	$5.25*10^{-6}$	4.771	5.132	318.9	5.3	4.3	9.0
Max-min	20.58	10188	18.21	9001.1	22072	$1.93*10^{-6}$	9.045	7.484	954.2	21	22	18.4
Min-max	16.76	9972	16.44	8762.9	22213	$1.34*10^{-6}$	8.758	6.323	901.2	18.3	11.3	12.6
Sufferage	16.72	9843	16.30	8634.4	22312	$1.55*10^{-6}$	8.650	6.171	876.8	17	10.3	12.0
Sufferage2	**16.37**	11353	16.26	10142	21201	$2.12*10^{-7}$	9.857	6.442	1081	20.3	11.8	13.2
RC	16.87	7210	**16.25**	6017.5	24187	$3.77*10^{-6}$	6.351	5.297	510.3	11	7.3	10.6
LJFR-SJFR	17.29	5097	16.67	3907.0	25913	$5.14*10^{-6}$	4.804	5.295	323.5	7.3	6.3	10.1
MECT	17.05	5048	16.45	**3859.1**	25949	$5.56*10^{-6}$	4.760	5.162	317.1	4.7	**3.3**	**8.8**
WQ	60.69	30110	59.95	28879	**20290**	$1.47*10^{-6}$	30.22	40.08	9996	26	26	21.0
JIT	19.18	10108	18.75	8911.4	21716	$2.23*10^{-6}$	6.481	**2.875**	528.9	9	20.8	13.9

by the ATC, COVERT, and MON rules, with MON achieving the best result for the Nwt criterion, and ATC for the Twt criterion. As for the previous set, the best result for the T_{max} criterion was achieved by the JIT rule.

The ATC rule achieved the best average rank for the set of due date related criteria for this problem set as well. The MON and COVERT rules again obtained the second best and third best ranks, with the MON rule obtaining a lower average rank for the due date related criteria. On the other hand, for the set of completion time and flowtime related criteria the best average rank was obtained by the MECT rule, whose results were among the top five for each of the criteria in this set. The min-min and KPB rules came second and third with somewhat larger average ranks. Regarding the average ranks on all the criteria, there are certain changes in the ranks compared to the previous problem set. For example, the best average rank was achieved by the MECT rule, which is not surprising considering that for six criteria it was among the top five rules. However, for the $Etwt$ and M_{ut} criteria, this rule achieved the worst results among all the tested DRs. Min-min, KBP, and LJFR-SJFR were among the top five rules even though they did not achieved the best result for any of the criteria. Nevertheless, they achieved good performance on most of the criteria but, similarly as the MECT rule, they performed quite bad for the $Etwt$ and M_{ut} criteria. The ATC rule performs well for the due date related criteria, while for the other criteria it performs worse to a certain extent. Nevertheless, this rule was still able to achieve the third best rank among all the DRs.

Table 3 represents the results achieved by the DRs when applied on problem instances generated with a low job heterogeneity and a high machine heterogeneity. Smaller changes in the performance of rules for some criteria are again noticeable when compared to the previous two test sets, but most of the rules retain a very similar performance. For the C_{max} and Cw criteria the same five rules once again achieve the best results. However, for the C_{max} criterion the min-max rule achieved the best result this time, followed closely by the Sufferage2 rule. For the Cw criterion the best result was once again achieved by the WSPT rule. For the F_{max} criterion, the best results are now achieved by rules which did not achieve the best results in the previous cases, such as MS, CR, EDD, and ERD which achieves the overall best result for this criterion. The MECT rule achieved once again the best result for the Ft criterion. For the $Etwt$ criterion the best result was achieved by the WQ rule, with the other rules performing similar as for the previous two problem sets. Once again for the M_{ut} criterion the LPT DR achieved the best result. For the due date related criteria the ATC, COVERT, and MON rules achieve the top results for all the three criteria. This time the MON and COVERT rules achieved better values for the due date related criteria than ATC, which is probably due to the choice of the parameter value for ATC.

Table 3: Results for the test set generated with a low job and a high machine heterogeneity

	C_{max}	Cw	F_{max}	Ft	$Etwt$	M_{ut}	Nwt	T_{max}	Twt	$Rank_T$	$Rank_{cf}$	$Rank$
ELB	11.99	4583	7.176	773.5	1093	$1.56*10^{-8}$	2.330	0.719	6.422	-	-	-
MCT	15.45	7194.0	13.67	3314	25042	$1.34*10^{-5}$	4.386	3.764	225.4	19	19.5	17.6
MET	13.73	4797.1	12.28	938.8	27088	$2.20*10^{-5}$	2.673	2.837	59.39	10.3	9	12.1
ERD	14.77	7238.9	**7.191**	3366	24942	$9.78*10^{-6}$	4.317	2.163	199.3	11.7	14	12.2
LPT	14.08	9176.1	13.69	5317	23449	**$3.06*10^{-6}$**	5.745	4.445	423.8	25	21	18.1
WSPT	14.04	**4583.3**	13.00	1417	26693	$1.58*10^{-5}$	3.0368	3.368	103.9	14.3	10.5	13.3
Maxstd	14.19	8607.7	13.76	4748	23936	$3.20*10^{-6}$	5.387	4.410	378.1	23	20.3	17.7
SA	13.69	4836.9	11.97	975.6	27056	$2.24*10^{-5}$	2.696	2.719	60.97	10	9	12.0
KPB	12.94	4677.4	11.48	820.2	27186	$2.43*10^{-5}$	2.558	2.542	49.60	6	5	9.8
OMCT	14.21	8634.1	13.77	4769	23922	$3.15*10^{-6}$	5.403	4.431	382.5	24	21.3	18.2
OLB	14.48	4997.4	13.00	1140	26925	$2.36*10^{-5}$	2.798	3.160	76.96	12.3	13	14.7
EDD	14.88	6762.8	7.900	2891	25368	$1.54*10^{-5}$	4.096	2.182	175.0	11.3	13.3	12.8
MS	14.81	7253.7	7.227	3381	24931	$1.01*10^{-5}$	4.348	2.181	201.3	12.7	15	13.0
MON	14.12	5304.2	12.92	1445	26606	$1.60*10^{-5}$	**2.493**	1.769	**41.97**	**1.3**	13.3	10.2
CR	14.86	6982.3	9.270	3119	25267	$1.22*10^{-5}$	4.440	2.700	240.7	16.3	14	14.3
COVERT	14.02	5334.2	12.82	1470	26580	$1.51*10^{-5}$	2.518	**1.726**	42.79	1.7	12.3	9.3
ATC	14.06	5360.4	12.89	1495	26556	$1.58*10^{-5}$	2.526	1.783	43.47	3	13.5	10.4
Min-min	12.96	4677.5	11.51	822.7	27186	$2.37*10^{-5}$	2.571	2.558	49.73	7	6	10.3
Max-min	15.94	7841.8	8.337	3968	24453	$7.96*10^{-6}$	4.794	2.632	255.4	17	17	14.9
Min-max	**12.01**	6585.5	11.03	2728	25518	$7.63*10^{-6}$	3.922	2.952	170.1	14	8.3	10.7
Sufferage	12.43	7021.4	11.68	3165	25163	$6.50*10^{-6}$	4.206	3.307	211.5	16.7	11.8	12.7
Sufferage2	12.15	8698.1	11.76	4828	23779	$4.19*10^{-6}$	5.372	3.602	349.7	21.3	15.3	14.8
RC	12.58	5161.8	11.23	1299	26776	$1.73*10^{-5}$	2.892	2.760	80.45	12	7.3	11.4
LJFR-SJFR	13.09	4692.7	11.62	833.7	27177	$2.27*10^{-5}$	2.599	2.724	51.65	9	7	11.1
MECT	12.73	4634.6	11.11	**773.5**	27223	$2.56*10^{-5}$	2.545	2.404	44.60	5	**3.8**	**9.1**
WQ	60.96	30025	59.34	26234	**19630**	$4.99*10^{-6}$	28.62	38.32	8959	26	26	20.9
JIT	17.33	9811.1	16.62	5952	22779	$8.89*10^{-5}$	4.698	3.870	303.8	21	25	19.3

For this problem set, the best rank for the due date related criteria was achieved by the MON rule, followed closely by the COVERT rule. The ATC rule, which achieved the best ranks for the previous two problem sets, achieved this time only the third best rank. For the set of completion time and flowtime related criteria the overall best rank was obtained by the MECT rule. The second best and third best average ranks were obtained by the KPB and min-min rules, respectively. This is mostly consistent with the performance of the rules on the previous problem set, except for the fact that the KPB rule now obtains a slightly better overall performance than the min-min rule. Considering the average ranks on all the criteria, the MECT rule achieved the best overall rank. Once again it performs well for most criteria, except for the $Etwt$ and M_{ut} criteria. The COVERT and MON rules also achieved a good rank, although they achieved good results only for the due date related criteria, while for the other criteria they achieved mostly mediocre results. The KPB and min-min rules also belong to the top five DRs by their ranks. Neither of those two rules achieved the best results for either one of the criteria, but managed to perform well for most of the criteria.

Finally, Table 4 represents the results achieved by the rules for the problem set with a low job and machine heterogeneity. From the results it is evident that for this problem instance the behaviour is much more different than for any of the previous three problem sets. For the C_{max} criterion, the best result was achieved by the Sufferage2 rule. However, rules like maxstd, max-min and MCT also obtained good results for this criterion, although on the previous three problem sets they were unable to do so. For the Cw criterion the situation is similar as for the previous problem sets, with WSPT achieving the best result once again. The max-min rule achieved the best result for the F_{max} criterion, while the MS and ERD rules also achieved very similar results. For the Ft criterion, the LJFR-SJFR rule achieved the best result by a small margin over the min-min, KPB and RC rules. The WQ rule again achieves the best result for the $Etwt$ criterion, with no other rule achieving even remotely good results. Additionally, the WQ rule achieves also the best result for the M_{ut} criterion as

Table 4: Results for the test set generated with a low job and a low machine heterogeneity

	C_{max}	Cw	F_{max}	Ft	$Etwt$	M_{ut}	Nwt	T_{max}	Twt	$Rank_T$	$Rank_{cf}$	$Rank$
ELB	30.33	13763	1.069	218.5	603	$1.19*10^{-7}$	2.220	0.164	7.972	-	-	-
MCT	30.3471	13780	1.599	232.6	22834	$3.39*10^{-4}$	2.254	0.236	8.970	16.3	14.9	14.9
MET	30.3534	13770	1.830	223.5	22842	$3.53*10^{-4}$	2.254	0.263	8.724	16	13.3	16.0
ERD	30.3377	13777	1.153	230.1	22836	$3.33*10^{-4}$	2.256	0.196	8.841	12	11.5	10.9
LPT	30.3331	13786	1.583	238.8	22828	$3.35*10^{-4}$	2.256	0.202	9.075	15.3	15	13.3
WSPT	30.3528	**13765**	1.738	225.0	22841	$3.48*10^{-4}$	2.261	0.252	8.939	19.7	11.5	15.6
Maxstd	30.3347	13783	1.540	236.5	22830	$3.36*10^{-4}$	2.252	0.227	9.207	15.7	13.3	12.8
SA	30.3534	13770	1.854	223.6	22842	$3.49*10^{-4}$	2.255	0.242	8.679	14.7	14.3	15.6
KPB	30.3534	13767	1.643	220.8	22844	$3.51*10^{-4}$	2.247	0.250	8.481	10.7	10.8	13.1
OMCT	30.3339	13782	1.569	235.6	22831	$3.38*10^{-4}$	2.256	0.218	9.017	16.3	13	13.6
OLB	31.203	14014	6.267	466.9	22619	$1.01*10^{-4}$	2.467	0.837	19.744	24	24	19.2
EDD	30.3526	13772	1.432	225.1	22841	$3.44*10^{-4}$	2.252	0.220	8.700	12.3	10.8	12.6
MS	30.3362	13776	1.151	229.0	22837	$3.38*10^{-4}$	2.251	0.194	8.889	8.7	10.5	10.1
MON	30.3534	13772	1.816	225.6	22840	$3.51*10^{-4}$	**2.242**	0.208	8.396	4.3	15	12.1
CR	30.3526	13773	1.641	226.7	22839	$3.52*10^{-4}$	2.255	0.275	8.935	18.3	14.8	16.8
COVERT	30.3516	13773	1.570	226.3	22839	$3.54*10^{-4}$	2.242	0.202	8.405	**4**	12.8	11.1
ATC	30.3516	13773	1.573	225.9	22840	$3.51*10^{-4}$	2.245	0.207	8.461	5.7	12.5	11.6
Min-min	30.3534	13767	1.735	220.7	22844	$3.50*10^{-4}$	2.252	0.251	8.442	10.7	10	12.6
Max-min	30.3353	13784	**1.150**	237.5	22830	$3.29*10^{-4}$	2.257	0.204	9.374	17	12	12.1
Min-max	30.3361	13770	1.159	223.2	22843	$3.33*10^{-4}$	2.246	0.197	8.567	6	6	**7.6**
Sufferage	30.3360	13768	1.205	221.1	22844	$3.35*10^{-4}$	2.252	**0.190**	8.474	5.7	**5.3**	**7.6**
Sufferage2	**30.3319**	13774	1.257	227.0	22839	$3.42*10^{-4}$	2.254	0.192	8.631	8.3	9.3	10
RC	30.3473	13768	1.410	220.8	22845	$3.37*10^{-4}$	2.246	0.218	8.519	8.7	7	10
LJFR-SJFR	30.344	13767	1.564	**220.6**	22844	$3.37*10^{-4}$	2.246	0.226	**8.396**	6.7	5.8	8.8
MECT	30.3600	13789	1.855	242.1	22825	$5.44*10^{-4}$	2.269	0.312	9.555	23	23	21.2
WQ	62.1436	31019	57.95	17450	**17331**	$\mathbf{0.43*10^{-4}}$	24.238	34.55	5867	26	26	20.4
JIT	31.7544	15792	19.61	2245	20977	$1.31*10^{-4}$	2.598	2.212	36.21	25	25	20.0

well. For this problem set, no single rule performs well for all three due date related. The ATC, MON, COVERT, and LJFR-SJFR rules perform well for the Twt and Nwt criteria, but usually not for the T_{max} criterion, for which the Sufferage rule performs the best.

The best rank for the due date related criteria was obtained by the COVERT rule, followed closely by the MON rule. The ATC rule achieved the third best average rank, the same as the sufferage rule. However, it is interesting to note that the average values of these rules for the set of due date related criteria are larger than for any of the previous three problem sets. The reason for this is due to the fact that no single rule obtains good results across all the due date related criteria, which then leads to a larger average rank. For the set consisting out of the completion time and flowtime related criteria the best average rank was obtained by the sufferage rule. The second and third best ranks were obtained by the LJFR-SJFR and min-max rules. However, even for this criteria set it is evident that the best average rank value is larger than it was in the previous three problem sets. Therefore, for the problem set with a low job and machine heterogeneity the DRs do not perform equally well on all criteria within these two groups as they did for the previous three problem sets. The overall best rank across all the criteria was achieved by both the Sufferage and min-max rules. These two rules demonstrated good performance across all criteria, except the $Etwt$ criterion for which they achieved among the worst results from all the rules. Surprisingly, neither of the rules which were among the best for solving due date related criteria for the previous three problem sets are now among the top five rules when considering their ranks. For this problem set the MS rule achieved the best rank among the DRs for optimising the due date related criteria. Although this rule did not perform well for the Twt criterion, it achieved relatively good results for several other criteria. Additionally, the LJFR-SJFR and Sufferage2 rules also achieved a good rank which placed them among the five best rules for this problem set.

The comparison of the DRs with the ELB values leads to some interesting observations. First of all, the DRs have shown to be least effective for the $Etwt$ and M_{ut} criteria. However, this is to be expected since DRs were usually not designed to optimise such criteria. For the due date related criteria the results obtained by the DRs when compared to the ELB values are still significantly worse. This is especially evident for the Twt criterion, while for the other two criteria the difference is not as prominent. Furthermore, for the problem set with low job and machine heterogeneity the differences between the results obtained by the DRs and the ELB values diminish for the due date related criteria. On the other hand, for the completion time and flowtime criteria the differences are quite small. This means that the DRs are most appropriate and well designed for dealing with such types of criteria.

Aside from considering the four problem sets individually, it is also interesting to denote the best ranking rules on all four problem sets together. The MON and COVERT rules obtain the best average rank of 2.8, while the ATC rule obtains the average rank of 3 for the set of due date related criteria. Based on the average rank values it is evident that these three rules perform well for the due date related criteria across all the problem sets. For the completion time and flowtime criteria the best rules across all four problem sets are the min-min, LJFR-SJFR, and KPB rules with average ranks equalling to 6, 6.3, and 6.4 respectively. The ranks denote that for this set of criteria it is harder for a single rule to perform well for all the problem sets. If all nine criteria and all four problem sets are considered at the same time, then the best average rank is obtained by the COVERT rule, with an average rank equalling to around 9.4. Aside from this rule, the LJFR-SJFR, ATC, and min-min rules also obtained quite good average rank values of 10, 10.1, and 10.1 respectively. The aforementioned rules can be considered the most versatile rules, since they achieve the best average rank value across all the tested problems and all the optimised criteria. Therefore, if the heterogeneity of the jobs and machines in the system is not known in advance and it is required for the rule to perform well over most of the criteria, then one of the aforementioned DRs should be used.

6. Discussion and analysis

In this section a short analysis based on the results obtained in the previous section will be performed. The analysis will be divided into two parts. The first part will focus on DRs and tries to analyse for what situations they are most well suited. In the second part for each criterion it will be analysed which are the best rules for that criterion.

6.1. Analysis of DRs

Although MET is a quite simple DR, it managed to achieve good results in some occasions. Except for the set with low machine and job heterogeneity, this rule performed well for the Cw and Ft criteria, usually being ranked between the fifth and seventh place for those two criteria. However, for the other criteria it did not achieve as good results. Therefore, it is evident that in cases of high heterogeneity it can be beneficial to schedule jobs by their processing times, since there will be large differences between the processing times of a single job. Thus, jobs should usually be scheduled on those machines for which they have smaller processing times. For the case when the heterogeneity is low, the performance of the rule deteriorates significantly for most criteria, since there is no large difference between the processing times on the different machines. MCT is also a simple rule, however, unlike the MET rule, it did not achieve as good results, especially for problems with high job heterogeneity. This behaviour is expected, since the jobs which should be scheduled are selected randomly. Therefore the rule does not select the jobs in any ingenious way, but just determines on which machine to schedule the selected job. In order to avoid the random selection of jobs, the OMCT rule orders the jobs by using a priority based on the standard deviation of processing times. However, using these priorities is only partially successful. For the problems with high heterogeneity the rule performed rather well for the $Etwt$ and M_{ut} criteria, being between the fourth and sixth best rule for the aforementioned criteria. However, for all the other criteria the rule obtained results which were among the worst. For the problems with low machine and job heterogeneity the rule performed well for the C_{max} criterion, while for the other criteria it again performed quite poorly. The MECT rule, which represents a combination of the MET and MCT rules, performs much better than the previous three rules for all problems except those with low machine and job heterogeneity. The rule achieved a good performance for all of the criteria except the $Etwt$ and M_{ut} criteria. The reason for such a good performance comes from the fact that if a job would not increase the makespan, the DR schedules the job on the machine with the shortest processing time. On the other hand, if it would increase the makespan the DR schedules the job on the machine with the earliest completion time. Therefore, the rule tries to simultaneously reduce the flowtime and makespan related criteria. However, the rule also reduces the due date related criteria as well, since it tries to execute jobs as fast as possible, thus reducing the possibility of them being late. This makes MECT one of the most versatile rules. The rule achieved the best results for the Ft and Cw criteria. However, on problems with low machine and job heterogeneity the rule achieved bad results. This is probably due to the fact that the processing times over the different machines do not have large variations and thus the adaptive part of the rule is not useful. These observations confirm those from the original paper where the MECT rule was proposed.

The WSPT rule represents an extension of MET, however, it also takes into account the weights of jobs when calculating priorities. This enables the rule to achieve the best result for the Cw criterion across all four problem sets, making it the main choice when this criterion needs to be optimised. For the other criteria the rule does not perform well, however, it performs better on problems with high heterogeneity. The LPT rule has shown variable behaviour depending on the heterogeneity of the problems. For problem sets with high heterogeneity, this rule achieved good performance for the $Etwt$ and M_{ut} criteria. By first scheduling jobs with the longest processing times, the rule is able to evenly distribute the load across several machines. When used for problem instances with low job and machine heterogeneity, this rule is also able to obtain a very good result for the makespan criterion. However, for most of the other criteria the rule did not obtain good results.

The goal of the ERD rule is to schedule those jobs which entered the system sooner. When the job heterogeneity is low, this rule obtains good results for the F_{max} and T_{max} criteria, since the rule will reduce the time which the job spends in the system. However, if the job heterogeneity is high the rule does not perform well for most of the criteria. The reason why the rule does not perform well under high job heterogeneity is because jobs will have largely different processing times. Therefore, if the processing times of jobs are not considered when calculating the priories, it is possible that jobs with large processing times will be scheduled. This will delay the execution of other jobs and increase the value of the scheduling objectives. The maxstd rule generally achieved bad results across all the criteria. For problems with high heterogeneity the rule achieved good results for the M_{ut} and $Etwt$ criteria. The performance of the rule did improve slightly when the heterogeneity is low for jobs and machines, even achieving a relatively good result for the C_{max} criterion. Even so, on the other criteria the results are not competitive with other rules. Therefore, the information about the standard deviation of the processing times does not provide any significant information during the scheduling process.

The results have demonstrated that the min-min rule was one of the best performing rules. This is especially true for the problems with high job heterogeneity, for which the rule performs well for the Cw and Ft criteria, being among the top three rules for those criteria. For the other criteria, except $Etwt$ and M_{ut}, the rule also achieves good results, being either the fifth or sixth best rule. This shows that min-min is an appropriate rule for cases where job heterogeneity is high. The reason for such a good performance comes from the fact that it tries to complete the jobs as soon as possible, which optimises the Ft criterion. However, for problem instances with low job heterogeneity, the performance of the rule deteriorates. The rule still achieves excellent results for the Ft and Cw criteria, but the performance on the other criteria deteriorates. The max-min heuristic is not as successful as the min-min heuristic. It performs well only for the problems with the low heterogeneity, and only for the F_{max} and M_{ut} criteria. This is again the consequence of smaller differences between processing times, since this rule selects those jobs with higher processing times, which leads to bad schedules on problems with high heterogeneity.

The min-max rule is much more successful. For example, for problems with high heterogeneity it achieved results which are among the best for the C_{max} and F_{max} criteria. Thus, the modification of the rule which also takes into account that jobs are schedules on machines on which they have shorter processing times is beneficial for optimising the makespan criterion. The reason for this is that by scheduling the rules on machines on which they have shorter processing times will effectively lead to their faster execution, but also keep certain machines free for more appropriate jobs which could be released in the future. For problems with low heterogeneity this rule is unable to achieve such good results for those two criteria. However, the rule performs well across most of the criteria, usually ranking between the fourth and seventh place. Such behaviour is expected, since for low heterogeneity conditions the differences between the processing times will be much less prominent. Therefore, scheduling jobs on machines on which they have smaller processing times will not have an equally strong effect as in conditions with higher heterogeneity.

The sufferage rule achieved good results for the C_{max} criterion, especially for the problems with high heterogeneity, where it was ranked between the second and fourth place. For problems with high job heterogeneity, it also achieved good results for the F_{max} and M_{ut} criteria. For the other criteria it achieved quite poor results. However, for problems with low machine and job heterogeneity, it performs rather well across most of the criteria, usually ranking either fifth or sixth for most criteria. Thus, the rule exhibits a similar behaviour as the min-max rule, being highly specialised under high heterogeneity for only a few criteria, but performing well for most criteria under low heterogeneity. The extended sufferage rule, denoted as sufferage2, has shown to be the best rule for the C_{max} criterion. It achieved the best results for all problem sets except for the one with low job and high machine heterogeneity, on which it achieved the second best result. For problems with high job heterogeneity the rule also achieved good results for the F_{max} criterion, ranking second or first, and the $Etwt$ criterion, for which it ranks second and third. However, for problems with high heterogeneity the rule achieves poor performance on most other criteria. On the problem instances with low heterogeneity the performance of the rule improves for other criteria, but still the performance for most of them was mediocre. Therefore, this rule highly specialised for optimising the C_{max} criterion, while neglecting other criteria.

The SA rule achieved mediocre results across most of the criteria, regardless of the heterogeneity of the problem instances. Therefore, this is one of the rare rules which did not perform well on at least one criterion. The LJFR-SJFR has shown to perform well across most criteria, except the $Etwt$ and M_{ut} criteria, when considering problem instances with high heterogeneity. For most criteria it achieved a rank between seven and nine, except for the Ft and Cw criteria, for which it achieved the forth and fifth best result, respectively. However, for the problems with low heterogeneity the rule achieves the best result for the Ft and Twt criteria, and second best for the Cw criterion. Although it is surprising that this rule achieved the best result for the Twt criterion, it can be explained by the fact that by optimising the Ft criterion the rule reduced the amount of time the jobs spent in the system, and therefore indirectly also reduced the tardiness of the jobs. KPB is another rule which achieved a good performance for the Cw and Ft criteria across all problem sets, always achieving values which are among the top four results. For problems with high job heterogeneity the rule achieved good results for other criteria, ranking between sixth and ninth place for all other criteria, except for the $Etwt$ and M_{ut} criteria. However, as the heterogeneity is reduced, the results of this rule also deteriorate. Therefore, limiting the the machines on which jobs can be scheduled is more beneficial under high heterogeneity conditions, since in those conditions the processing times of jobs will have very different values for the different machines. The behaviour of the RC rule is interesting since it varies with regards to the heterogeneity of the problems. For high job heterogeneity this rule achieves good results for the C_{max} and F_{max} criteria, while performing mediocre for other criteria. However, for problem instances with low job and high machine heterogeneity, the rule performs well for most criteria, but does not excel in either. On the other hand, for problems with low heterogeneity the rule performs the best for the Ft criterion and well for most other criteria. Therefore, RC proves to be a quite versatile rule, across all the heterogeneity conditions.

The OLB rule has generally achieved mediocre or bad performance regardless of the heterogeneity of the problem instances. However, this is expected since the rule randomly selects the job which should be scheduled next. The WQ rule performs poorly for most of the rules. The only criteria for which it performs well are the $Etwt$ and M_{ut} criteria. However, it does not consistently perform well on both of the aforementioned criteria. For M_{ut} it performs well consistently across all the problem types, but obtains the best result for problems with low heterogeneity. On the other hand, for all problem sets, except the one with high machine and job heterogeneity, this rule performed well for the $Etwt$ criterion. The reason for this is that by dispatching jobs across machines to evenly distribute the load will lead to an increase in tardiness, but also to a significant decrease in the earliness of the jobs, which then also leads to a better value of the $Etwt$ criterion. The JIT rule was proposed for optimising the $Etwt$ criterion, which it does since it usually achieved either the best or second best result for the $Etwt$ criterion. However, apart from optimising this criterion, the rule also achieved the best result for the T_{max} criterion when applied on problem instances with high job heterogeneity. Thus, by trying to schedule the job as close to its due date, many jobs end up being late, however, none of them ends up being completed long after their due date.

The EDD rule is the simplest rule designed for optimising due date related criteria. Due to its simplicity it was unable to achieve good results on any of the tested criteria, except for F_{max} in only one occasion. The MON rule, although also being a rather simple rule, achieved much better results than EDD. The rule performed well for all the due date related criteria. It performed especially well for the Nwt criterion for which it achieved the best result for all problem sets, except the one with high machine and job heterogeneity, where it achieved the third best result. The MS rule was shown to perform poorly on problem instances with high job heterogeneity. On the problem sets with low job heterogeneity the rule achieves good results, mostly for the F_{max} criterion, but also for the T_{max} criterion. In the case of low heterogeneity, the rule does not perform poorly for any of the tested criteria. Therefore, scheduling jobs closer to their due date is mostly useful for problems with low job heterogeneity. The CR rule represents a simple extension of the MS rule. Unfortunately, this rule did not achieve good results on most of the problem sets. Therefore, the rule did not demonstrate any advantages with regards to the other tested rules.

The COVERT rule is similar to the CR rule, however, it does not allow for the priorities to increase with the increase of the tardiness of the job. Such a modification has proven to be useful, since the rule achieved extremely good results. Naturally, the rule achieved the best results for the due date related criteria. The best performance of the rule can be noted on the problem sets with high machine heterogeneity, on which the rule always achieved results which were among the top three for the due date related criteria. Additionally, for these two problem sets the rule did not achieve poor results for any of the tested criteria. This allowed the rule to achieve a high rank when all the criteria are considered, meaning that the rule is well rounded. However, for the problem sets with low machine heterogeneity, the performance of the rule deteriorated. The rule still achieved good results for the due date related criteria, but usually it did not achieve the best results for any of the criteria. Therefore, the rule seems to struggle in choosing the right machine on which it should schedule the job, since it achieves inferior performance when the execution time of jobs is similar across all machines. The ATC rule represents a further extension of the COVERT rule, which uses an exponential function to model the priorities. This rule achieves the best performance when applied on problem sets with high job heterogeneity. On these sets it achieves the best result for the Twt criterion, and second best results for the Nwt and T_{max} criteria. For the problem sets with high job heterogeneity this rule even obtained the best average rank when only the due date related criteria are considered. However, the rule achieved mediocre results for most of the other criteria, but still for neither of the criteria it achieved the worst results. For problems with low job heterogeneity the performance of the rule deteriorates, but it still manages to perform well for most due date related criteria, obtaining the third best rank for them.

6.2. Analysis of scheduling criteria

For the C_{max} criterion the best results are achieved by the sufferage2, min-max, sufferage, and MECT rules. All these rules are similar in the fact that they use the minimum completion time to determine on which machine the current job should be scheduled. The differences arise mostly in the way that they select the job which should be scheduled. For example, min-max tries to execute the job on the machine for which it has the minimal processing time. Sufferage2 and sufferage take into account the difference between the shortest and second shortest minimum completion time to determine which job would "suffer" most if not executed on the machine with the minimum completion time. MECT, on the other hand, schedules a great deal of jobs on machines on which they achieve their minimum processing time. Therefore, to optimise the C_{max} criterion, it is not enough for the rules to take into account the minimum processing times of jobs. They have to additionally ensure that either the jobs are executed on machines on which they have a short processing time, or that the job can not be scheduled efficiently on any other machine. With these strategies the rule ensures that it will not schedule jobs on just any machine, but rather that it will keep certain machines free if it determines that the current jobs can not be efficiently executed on them.

The Cw criterion is the only criterion for which one rule achieved the best result across all the test sets. For this criterion the WSPT rule achieved the best results, which is expected since it directly uses the information about the weights of jobs, in addition to their processing times. Other rules which perform well for this criterion are the min-min, KPB, and LJFR-SJFR rules. However, neither of these three rules takes into account the weights of jobs. Min-min schedules the jobs with the smallest minimum execution time, thus trying to complete the jobs as soon as possible. KPB works similarly, however, it additionally limits the number of machines on which the job can be scheduled. The LJFR-SJFR also uses the minimum completion time, however, it interchangeably schedules jobs with the longest and smallest completion time. Therefore, the best results for this criterion are achieved by rules which take into account directly the job weight for the completion times, or which schedule jobs by their minimum completion times. However, no rule combines both of these, therefore it could be possible to extend the min-min rules with job weights to obtain better results for this criterion.

For the $Etwt$ criterion it is not simple to find a single DR which performs well on all problem sets. The proposed JIT rule most consistently achieved the best results for this criterion. The LPT rule also achieved good results for the *Etwt* criterion across all the problem sets. It is interesting that this rule performs well for this criterion, although not using information about earliness or tardiness at all. The explanation for such a result lies in the fact that by prioritising rules with the longest processing times, the jobs are completed at a later moment in time, thus increasing tardiness, but reducing earliness. The WQ rule achieves an interesting result for this criterion, since it performs well on all problem sets, except on the set with high job and machine heterogeneity. Therefore, balancing the load across all the machine also has a good effect on the *Etwt* criterion. Unfortunately, it is hard to find a common behaviour between the three aforementioned rules. The only thing they have in common is that they do necessarily schedule the jobs with the minimum completion time, and that jobs usually have a large completion time. Only the proposed JIT rule directly uses the tardiness and earliness information to schedule the jobs, and therefore should be more reliable than the other two methods. Additionally, in most cases the JIT rule achieves a better value for the *Twt* criterion, which means that it still tried to reduce the tardiness of jobs.

In the case of the F_{max} criterion, no single rule achieved the best performance on all problem sets. The best results were obtained by the RC, sufferage, sufferage2, min-max, and ERD rules. The first four rules try to schedule jobs on most appropriate machines. Usually, they achieve this by trying to schedule a job with the minimum completion time, but also taking into account that the job is not scheduled on a machine on which it has a large processing time. With this the rules can decide not to schedule jobs on certain machines, in order to keep them free until a more suitable job is released. Consequentially, the rules will try to execute the jobs as soon as possible, but will rather try to schedule them on machines which are most appropriate. This will allow for jobs to be executed as fast as possible, and will thus reduce the time which the jobs spend in the system. The ERD rule works in a different way, since it prioritises jobs which were released earlier. With this the rule implicitly tries to reduce the flowtime of each job, since it will try to schedule it as soon as a machine becomes free. However, this rule usually did not perform equally well as the aforementioned rules. Thus, for this criterion, it is beneficial to schedule jobs by using the minimum completion time and some additional criteria which ensure that the job is scheduled on the most appropriate machine.

The results for the Ft criterion are quite similar to those obtained for the Cw criterion, which in itself is expected since both criteria have a similar definition. The best results for this criterion are achieved by the min-min, KPB, and LJFR-SJFR rules. The reason why the aforementioned three rules perform well is due to the fact that all three rules try to schedule jobs by their minimum completion times. Therefore, they try to minimise the amount of time that the rules spend in the system. On the other hand, the WSPT rule which performed best for the Cw criterion, does not perform well for this criterion since it uses job weights which are not used in the definition of the Ft criterion. Therefore, the best rules for this criterion are those which try to schedule jobs so that they finish executing as soon as possible, such as the min-min rule.

The situation for the M_{ut} criterion is not as simple as for the previous criteria, since no single rule performs well for this criterion across all four problem sets. This is likely the result of the fact that the heterogeneity of the problem instances has a large influence on this criterion, and how the schedule which optimises it should be constructed. For problem instances with high heterogeneity the LPT rule achieved the best results. This is probably due to the fact that there is a high variability between processing times of jobs. Therefore, by executing those which have the longest processing time first the rule can more evenly distribute the load across all machines. On the other hand, for problems with low heterogeneity, the best results were achieved by the WQ rule. This is probably due to the fact that all the processing times are now more or less similar, and therefore this rule can more effectively distribute the balance across all the machines. Other rules which performed well for this criterion are OMCT and maxstd rules for problems with high heterogeneity. On the other hand, the OLB and JIT rules performed best under low job and machine heterogeneity. This demonstrates that to optimise this criterion under different conditions rules which have a completely different behaviour are required.

For the Nwt, T_{max}, and Twt criteria the best results are achieved by rules which are designed for optimising the due date related criteria. The three best rules for the aforementioned criteria were MON, COVERT, and ATC. For the Nwt and Twt criteria the best results are mostly achieved by the MON rule, while for the T_{max} criterion the COVERT rule achieved the best results in most cases. All three rules perform well for the three tested due date related criteria. It is interesting to note how the MON rule, which uses only a static slack factor, performs better in certain occasions than the two rules which use a dynamic slack factor. The obvious reason for this is that the other two rules use an additional scaling parameter which influences the performance of the rules. Therefore, it is likely possible that if other values for those scaling parameters were used, the ATC and COVERT rules would achieve better performance. When rules are executed under dynamic conditions, it is not known in advance which parameter value would lead to the best results. For the T_{max} criterion the best results for problem sets with high job heterogeneity were achieved by the JIT rule. Although the rule results in large tardiness values, it manages to schedule jobs in a way that no job is extensively late, at least when the processing times of jobs are highly different between jobs. In the end, it is evident that for the due date related criteria the rules need to use the information about the slack of the jobs in order to obtain the best results.

7. Conclusion

In this paper a review of existing DRs which can be applied for scheduling in the unrelated machines environment with release times and under dynamic conditions was given. Additionally, all the collected DRs were evaluated on several problem sets and by using nine scheduling criteria. The results demonstrate that there is no single DR which would perform well for all of the nine tested criteria, but rather that DRs usually achieve the best performance for only one or two criteria, or perform well across several criteria but do not excel in any of the criteria. For most criteria it was possible to determine several DRs which performed well across all the test sets, regardless of the heterogeneity conditions. However, for only a few criteria a single DRs was able to achieve the best result for all four problem sets. This shows that DRs are quite sensitive to the heterogeneity conditions of the problem instances, and that changing those conditions can have an influence on the performance of DRs. For some criteria, like $Etwt$ and M_{ut}, the performance of DRs depends on a much greater extent to the heterogeneity conditions. The results obtained in this paper should, however, give a good notion of which DRs are appropriate for optimising which scheduling criteria, and under which heterogeneity conditions.

Although this paper gives an overview of the different DRs, it is still possible to make other reviews which would focus only on specific criteria. For example, it would be interesting to investigate how the due date range and tightness influence the performance of different DRs for the due date related criteria. With such an investigation it would be possible to determine which rules are appropriate for problems with specific tardiness conditions. Since this study gave an overview of rules which performed best for each of the criteria, and also outlined certain similarities between those rules, it could be possible to use that knowledge to design novel DRs, which could perform better than the existing rules. Finally, it could also be useful to analyse all the good DRs in more detail, in order to identify the useful parts of these manually designed DRs, and to try to use these parts when automatically designing new DRs by using different machine learning and evolutionary computation methods. This could very likely lead to simpler, but more efficient automatically designed DRs.

Allahverdi, A., Gupta, J. N., & Aldowaisan, T. (1999). A review of scheduling research involving setup considerations. *Omega*, *27*, 219–239. doi:`10.1016/S0305-0483(98)00042-5`.

Allahverdi, A., Ng, C., Cheng, T., & Kovalyov, M. Y. (2008). A survey of scheduling problems with setup times or costs. *European Journal of Operational Research*, *187*, 985–1032. doi:`10.1016/j.ejor.2006.06.060`.

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`.

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. doi:`10.1006/jpdc.2000.1714`.

de C. M. Nogueira, J. P., Arroyo, J. E. C., Villadiego, H. M. M., & Goncalves, L. B. (2014). Hybrid GRASP Heuristics to Solve an Unrelated Parallel Machine Scheduling Problem with Earliness and Tardiness Penalties. *Electronic Notes in Theoretical Computer Science*, *302*, 53–72. doi:`10.1016/j.entcs.2014.01.020`.

Cheng, V., Crawford, L., & Menon, P. (1999). Air traffic control using genetic search techniques. In *Proceedings of the 1999 IEEE International Conference on Control Applications (Cat. No.99CH36328)* (pp. 249–254). IEEE volume 1. doi:`10.1109/CCA.1999.806209`.

Corman, F., & Quaglietta, E. (2015). Closing the loop in real-time railway control: Framework design and impacts on operations. *Transportation Research Part C: Emerging Technologies*, *54*, 15 – 39. URL: `http://www.sciencedirect.com/science/article/pii/S0968090X15000169`. doi:`https://doi.org/10.1016/j.trc.2015.01.014`.

Cota, L. P., Haddad, M. N., Souza, M. J. F., & Coelho, V. N. (2014). AIRP: A heuristic algorithm for solving the unrelated parallel machine scheduling problem. In *2014 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1855–1862). IEEE. doi:`10.1109/CEC.2014.6900245`.

Dimopoulos, C., & Zalzala, A. (2000). Recent developments in evolutionary computation for manufacturing optimization: problems, solutions, and comparisons. *IEEE Transactions on Evolutionary Computation*, *4*, 93–113. doi:`10.1109/4235.850651`.

Du Kim, H., & Kim, J. S. (2004). An online scheduling algorithm for grid computing systems. In M. Li, X.-H. Sun, Q. Deng, & J. Ni (Eds.), *Grid and Cooperative Computing: Second International Workshop, GCC 2003, Shanghai, China, December 7-10, 2003, Revised Papers, Part II* (pp. 34–39). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:`10.1007/978-3-540-24680-0_5`.

Durasević, M., & Jakobović, D. (2017). 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`.

Fanjul-Peyro, L., & Ruiz, R. (2010). Iterated greedy local search methods for unrelated parallel machine scheduling. *European Journal of Operational Research*, *207*, 55–69. doi:`10.1016/j.ejor.2010.03.030`.

Fanjul-Peyro, L., & Ruiz, R. (2011). Size-reduction heuristics for the unrelated parallel machines scheduling problem. *Computers & Operations Research*, *38*, 301–309. doi:`10.1016/j.cor.2010.05.005`.

Fanjul-Peyro, L., & Ruiz, R. (2012). Scheduling unrelated parallel machines with optional machines and jobs selection. *Computers & Operations Research*, *39*, 1745–1753. doi:`10.1016/j.cor.2011.10.012`.

Hansen, J. V. (2004). Genetic search methods in air traffic control. *Computers & Operations Research*, *31*, 445–459. doi:`10.1016/S0305-0548(02)00228-9`.

Hart, E., Ross, P., & Corne, D. (2005). Evolutionary Scheduling: A Review. *Genetic Programming and Evolvable Machines*, *6*, 191–220. doi:`10.1007/s10710-005-7580-7`.

Izakian, H., Abraham, A., & Snasel, V. (2009). Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Environments.

In *2009 International Joint Conference on Computational Sciences and Optimization* (pp. 8–12). IEEE. doi:`10.1109/CSO.2009.487`.

Kofler, M., Wagner, S., Beham, A., Kronberger, G., & Affenzeller, M. (2009). Priority Rule Generation with a Genetic Algorithm to Minimize Sequence Dependent Setup Costs. In R. Moreno-Díaz, F. Pichler, & A. Quesada-Arencibia (Eds.), *Computer Aided Systems Theory - EUROCAST 2009: 12th International Conference, Las Palmas de Gran Canaria, Spain, February 15-20, 2009, Revised Selected Papers* (pp. 817–824). Berlin, Heidelberg: Springer Berlin Heidelberg. doi:`10.1007/978-3-642-04772-5\_105`.

Lee, J.-H., Yu, J.-M., & Lee, D.-H. (2013). A tabu search algorithm for unrelated parallel machine scheduling with sequence- and machine-dependent setups: minimizing total tardiness. *The International Journal of Advanced Manufacturing Technology*, *69*, 2081–2089. doi:`10.1007/s00170-013-5192-6`.

Lee, Y. H., Bhaskaran, K., & Pinedo, M. (1997). A heuristic to minimize the total weighted tardiness with sequence-dependent setups. *IIE Transactions*, *29*, 45–52. doi:`10.1080/07408179708966311`.

Leung, J. Y.-T. (2004). *Handbook of scheduling : algorithms, models, and performance analysis*. Boca Raton, Fla.: Chapman & Hall/CRC.

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. doi:`10.1006/jpdc.1999.1581`.

Morton, T. E., & Pentico, D. W. (1993). *Heuristic Scheduling Systems*. John Wiley And Sons, Inc.

Morton, T. E., & Rachamadugu, R. M. V. (1982). *Myopic Heuristics for the Single Machine Weighted Tardiness Problem.*. Technical Report DTIC Document.

Munir, E. U., Li, J., Shi, S., Zou, Z., & Yang, D. (2008). Maxstd: A task scheduling heuristic for heterogeneous computing environment. *Information Technology Journal*, *7*, 679–683.

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

Petrovic, S., & Castro, E. (2011). A genetic algorithm for radiotherapy pretreatment scheduling. In C. Di Chio, A. Brabazon, G. A. Di Caro, R. Drechsler, M. Farooq, J. Grahl, G. Greenfield, C. Prins, J. Romero, G. Squillero, E. Tarantino, A. G. B. Tettamanzi, N. Urquhart, & A. Ş. Uyar (Eds.), *Applications of Evolutionary Computation* (pp. 454–463). Berlin, Heidelberg: Springer Berlin Heidelberg.

Pfund, M., Fowler, J. W., Gadkari, A., & Chen, Y. (2008). Scheduling jobs on parallel machines with setup times and ready times. *Computers & Industrial Engineering*, *54*, 764–782. doi:`10.1016/j.cie.2007.08.011`.

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

Quaglietta, E., Pellegrini, P., Goverde, R. M., Albrecht, T., Jaekel, B., Marlière, G., Rodriguez, J., Dollevoet, T., Ambrogio, B., Carcasole, D., Giaroli, M., & Nicholson, G. (2016). The on-time real-time railway traffic management framework: A proof-of-concept using a scalable standardised data communication architecture. *Transportation Research Part C: Emerging Technologies*, *63*, 23 – 50. URL: `http://www.sciencedirect.com/science/article/pii/S0968090X15004143`. doi:`https://doi.org/10.1016/j.trc.2015.11.014`.

Rafsanjani, M. K., & Bardsiri, A. K. (2012). A new heuristic approach for scheduling independent tasks on heterogeneous computing systems. *International Journal of Machine Learning and Computing*, *2*, 371.

e Santos, A. S., & Madureira, A. M. (2014). Ordered minimum completion time heuristic for unrelated parallel-machines problems. In *2014 9th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1–6). IEEE. doi:10.1109/CISTI.2014.6876939.

Tseng, L.-Y., Chin, Y.-H., & Wang, S.-C. (2009). A minimized makespan scheduler with multiple factors for Grid computing systems. *Expert Systems with Applications*, *36*, 11118–11130. doi:10.1016/j.eswa.2009.02.071.

Đurasević, 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. doi:10.1109/MIPRO.2016.7522347.

Vepsalainen, A. P. J., & Morton, T. E. (1987). Priority Rules for Job Shops with Weighted Tardiness Costs. *Management Science*, *33*, 1035–1047. doi:10.1287/mnsc.33.8.1035.

Wang, I.-L., Wang, Y.-C., & Chen, C.-W. (2013). Scheduling unrelated parallel machines in semiconductor manufacturing by problem reduction and local search heuristics. *Flexible Services and Manufacturing Journal*, *25*, 343–366. URL: http://link.springer.com/10.1007/s10696-012-9150-7. doi:10.1007/s10696-012-9150-7.

Xhafa, F., Barolli, L., & Durresi, A. (2007). Batch mode scheduling in grid systems. *International Journal of Web and Grid Services*, *3*, 19–37. doi:10.1504/IJWGS.2007.012635.

Yang-Kuei, L., & Chi-Wei, L. (2013). Dispatching rules for unrelated parallel machine scheduling with release dates. *The International Journal of Advanced Manufacturing Technology*, *67*, 269–279. doi:10.1007/s00170-013-4773-8.