Maximising the number of on-time jobs on parallel servers with sequence dependent deteriorating processing times and periodic maintenance

Eduardo Pérez*
Ingram School of Engineering, Texas State University, 601 University Drive San Marcos, TX 78666, USA
Email: eduardopr@txstate.edu
*Corresponding author

Rahul R. Ambati
Department of Computer Science, Texas State University, 601 University Drive San Marcos, TX 78666, USA
Email: rahul@txstate.edu

Alex J. Ruiz-Torres
Facultad de Administración de Empresas, Universidad de Puerto Rico – Rio Piedras, San Juan, PR 00931-3332, USA
Email: alex.ruiztorres@upr.edu

Abstract: This paper considers a parallel-machine scheduling problem with sequence dependent processing times and periodic maintenance. The time to complete jobs increases as the machines deteriorate and the machine deterioration depends on the particular job sequence assigned to a machine. The planned maintenance activity returns the machine to its optimal condition, and all machines undergo this maintenance activity at the same time. The objective is to find the job schedule that maximises the number of on-time jobs given a specified maintenance schedule. The paper presents a mathematical programming formulation, several solution algorithms, and evaluates their performance under various experimental conditions.

Keywords: parallel machines; machine deterioration; late jobs; on-time jobs; scheduling; maintenance.


Biographical notes: Eduardo Pérez is an Assistant Professor at Texas State University, Ingram School of Engineering, San Marcos, Texas, USA. He obtained his PhD in Systems and Industrial Engineering from the Texas A&M University and BS in Industrial Engineering from the University of Puerto Rico.
E. Pérez et al.

Rico, Mayaguez Campus. He is the Director of the Integrated Modeling and Optimisation for Service Systems (iMOSS) research laboratory. His research interests include stochastic optimisation and discrete-event simulation with application to healthcare systems, parallel machines, and renewable energy. He has served as a Session Chair for the IIE and INFORMS conferences and as a reviewer for journals such as *Simulation*, *IIE Transactions*, the *European Operational Research*, and *Omega*.

Rahul R. Ambati is a graduate student in the Department of Computer Science at Texas State University. He received his Bachelor in Computer Science from the JNT University, India in 2011. His research interests include scheduling and computer applications.

Alex J. Ruiz-Torres is a full time faculty member of the Facultad de Administracion de Empresas at the University of Puerto Rico, Rio Piedras Campus. He has received two Fulbright awards including the 2014 Fulbright – Aalto University Distinguished Chair (Helsinki, Finland). He has also been awarded two NASA Faculty Fellowships at the Kennedy Space Center and a NASA Small Business Technology Transfer grant. He has been a Visiting Lecturer at universities in Europe, the Middle East, and the Americas. He has consulted for manufacturing, logistic, and healthcare organisations in the USA, Ecuador, Mexico, and China. He has over 50 journal publications, including articles in the *European Journal of Operations Research*, *Computers and Operations Research*, *International Journal of Production Economics*, *Computers and Industrial Engineering*, *Omega*, and the *International Journal of Production Research*. His research interests include production planning, supply chain models, simulation modelling, and knowledge-based systems.

1 Introduction

This paper studies a parallel-machine scheduling problem with job sequence dependent processing times in which periodic maintenance activities are performed on the machines. Most of the current literature assumes that machines are always available over the scheduling horizon which is not the case in practice given the need for periodic repairs and planned maintenance. Maintenance sustains machine efficiency which is important in production environments. It is assumed that machines are not available during these maintenance periods, which affects the schedule and the ability of the shop to complete jobs. Consequently, it is of great interest to production planners to have models and solution approaches to effectively plan these maintenance activities as to meet desired production metrics related to efficiency and customer service.

The goal of this paper is to derive scheduling algorithms for maximising the number of on-time jobs in an environment of parallel machines with periodic maintenance activities occurring at fixed time intervals, where the job’s processing time increases due to machine deterioration. The problem of deterioration depends upon the job sequence previously processed by the machine. This viewpoint of the production environment is in line with the work published by Yang (2011), Yang et al. (2012), and Ruiz-Torres et al. (2013, 2015). It is a relevant problem observed in the industrial and service settings and that focuses on satisfying customer requirements in terms of due dates.

Research related to the scheduling of deteriorating jobs abounds, but few papers were found by the authors dealing with the minimisation of late jobs, and none in the parallel
Maximising the number of on-time jobs on parallel servers

The problem of scheduling deteriorating jobs has been studied widely since the works published by Gupta and Gupta (1988), and Browne and Yechiali (1990). Two important reviews for deteriorating jobs have been published by Alidaee and Womer (1999) and by Cheng et al. (2004). In addition, the problem of minimising the number of late jobs has received similar attention by many researchers (Lai and Lee, 2011; Steiner and Zhang, 2011; Wang, 2007; Xu et al., 2010b). Reviews for scheduling problems with due date criteria have been done by (Biskup and Herrmann, 2008; Sterna, 2011; Xu et al., 2010a). To the best of the authors knowledge, only one research paper by Wang (2007) exists that deals with deteriorating jobs and the number of late jobs criterion and none exist that also consider maintenance activities.

Table 1 presents three techniques published in the literature to deal with the effect of job/machine deterioration. The key element of the deteriorating jobs’ problem is that the jobs’ processing time is a function of their start times or the number of jobs processed since the beginning of the schedule. In this paper, the job deterioration problem is modelled following the third technique listed in Table 1 where the deterioration of job processing times depends upon the specific jobs that have been processed by the machine previously. Thus, in the models proposed in this paper, the deterioration of the machine and therefore of the job processing times is a function of the sequence of jobs processed by the same machine.

Table 1  Job/machine deterioration modelling techniques

<table>
<thead>
<tr>
<th>Modelling form</th>
<th>Parameters</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>Processing time is a function of the job’s start time</td>
<td>• Deterioration factor&lt;br&gt;• Start time of the job</td>
<td>Ren and Kang (2007), Ji and Cheng (2008), Ji and Cheng (2009), Cheng et al. (2009), Huang and Wang (2011)</td>
</tr>
<tr>
<td>Processing time is a linear function of the job’s start time with baseline processing time</td>
<td>• Baseline processing time&lt;br&gt;• Deterioration factor&lt;br&gt;• Start time of the job</td>
<td>Kang and Ng (2007), Kuo and Yang (2008), Toksari and Güner (2009), Mazdeh et al. (2010)</td>
</tr>
<tr>
<td>Processing time is a function of the job’s position in the machine</td>
<td>• Deterioration effect of the job in the machine&lt;br&gt;• Machine selected&lt;br&gt;• Job position in the machine schedule</td>
<td>Yang (2011), Yang et al. (2012), Ruiz-Torres et al. (2013), Mosheiov (2011), Hsu et al. (2011)</td>
</tr>
</tbody>
</table>

The literature on parallel-machine scheduling with due date criteria is extensive. Ho and Chang (1995) discuss the performance of several heuristics derived from the optimal single machine (SM) algorithm from Moore (1968). Lin and Jeng (2004) developed a dynamic programming methodology and some heuristics to address the batch scheduling problem in parallel-machines. The authors consider two objective functions: the minimisation of the maximum lateness and the minimisation of tardy jobs. M’Hallah and Bulfin (2005) use integer programming and a branch a bound algorithms to minimise the weighted and un-weighted number of tardy jobs for the identical and unrelated parallel-machine settings. Bornstein et al. (2005) formulate the identical parallel-machine problem using a maximum flow network model and develop a polynomial algorithm that solves the problem. Ruiz-Torres et al. (2007) study the dual resource problem on
parallel-machines with the objective of minimising the number of late jobs. In this work, the speed of the machines depends on the allocation of a secondary resource which is limited in capacity and allocated at the beginning of the schedule. The authors propose a set of heuristics for solving the problem. Ravetti et al. (2007) address a scheduling problem with unrelated parallel machines, sequence dependent setups and due dates. The problem objective is to minimise the sum of the makespan and the weighted delays. The authors propose different configuration based on metaheuristic experiments.

The literature on machine scheduling considering maintenance is extensive (Hadidi et al., 2012; Wang, 2013; Wang et al., 2014; Yang et al., 2014). However, only two papers were found dealing with the due date criteria and periodic maintenance. Chen (2009) studies the SM scheduling problem with periodic maintenance to minimise the number of tardy jobs. The author provides an effective heuristic that obtains a near-optimal schedule for the problem and branch-and-bound algorithm to find the optimal schedule. Lee and Kim (2012) also consider the SM case and present a two-phase heuristic which provides close to optimal solutions. Papers studying the due date criteria in parallel machine scheduling without maintenance are also limited. Wang and Xia (2005) develop optimal algorithms for the SM problem with decreasing linear deterioration. Toksarı and Güner (2009) provide a mathematical formulation for the problem that minimises the tardiness of jobs with common due dates and position-based learning with linear and nonlinear deterioration. Lee and Kim (2012) consider the SM setting and formulates the problem of minimising the number of late jobs with deterioration effects and setup times. The paper uses a heuristic to show dominance properties, a lower bound, and an initial upper bound. The given algorithm can solve large instances in reasonable amounts of computational time. Lee and Lu (2012) formulate the problem of minimising the total weighted number of late jobs with deteriorating jobs and setup times for the SM setting. The authors use a branch-and-bound algorithm and give several dominance properties and a lower bound to solve the problem to optimality.

The rest of this paper is organised as follows: the problem description and a mixed-integer programming (MIP) formulation are presented in Section 2. Section 3 provides solution approaches for the problem and in Section 4 a computational study is presented. Section 5 concludes the paper and gives possible suggestions for future research.

2 Problem definition and notation

Consider the following example to illustrate the problem. A set of ten independent, non-divisible/pre-emptable non-sequence dependent major tasks (the jobs) are available at the start of the schedule to be completed by two teams. Each team has a break of 30 units of time after 180 units of labour. Each task has a baseline duration (if for example, performed first or after team rest/lunch break), a due date for a task, and a deterioration effect, considering the ‘wear and tear’ in the team’s performance level.

Figure 1 depicts the information for two possible schedules. The schedule for each team shows the task sequence and after the completion of each task the teams have a performance level and total time. The performance level for each team is computed as follows: performance after a task = performance before task × (1 – deterioration effect of completed task). For instance, schedule 1 for Team 1 has three assigned ordered tasks;
Maximising the number of on-time jobs on parallel servers

6, 1, and 3. Notice that the third task is performed after the teams break. A 100% performance level is assumed at the start of the schedule or after a break, thus after completing task 6, the team is at 96.3% performance level (\(= 100\% \times (1 - \text{deterioration effect of task 6})\)). Likewise, after completing task 1 the team performance level would be 92.5% (\(= 96.3\% \times (1 - \text{deterioration effect of task 1})\)). After the break, denoted by the grey colour in Figure 1, the team’s performance level returns to 100% and by the end of task 3 the team would have a 96.5% performance level (\(= 100\% \times (1 - \text{deterioration effect of task 3})\)).

**Figure 1** Illustrative example of the problem (see online version for colours)

The teams’ performance level has a direct effect on the total time required to complete the assigned tasks. For example, consider schedule 1 for Team 2, the sum of the baseline times for the tasks scheduled before maintenance equal 136 units versus 149 units which is the total time required to complete the tasks sequence given the deterioration effects. The deterioration effect increases the completion time of a sequence of jobs. Consequently, the resource capacity to complete more tasks within the fixed time period before maintenance is reduced. The two schedules presented in the example have different values for the objective function under consideration; schedule 1 results in only six tasks completed on-time (60% on-time) while schedule 2 results in eight tasks completed on-time (80% on-time).

This study focuses on a parallel-machine scheduling problem in which periodic maintenance activities are performed on the machines, and this time is ‘fixed’ at the start of the planning process (in other words is not part of the decision process). There are \(n\) independent non-divisible jobs \(N = \{1, \ldots, j, \ldots, n\}\) available at time zero and pre-emption is not allowed. There are \(m\) parallel-machines \(M = \{1, \ldots, k, \ldots, m\}\) which can only process one job at a time. The time interval between two consecutive maintenance activities is fixed and each maintenance activity should be started exactly at the planned time specified by the interval. Machines may be idle before maintenance activities. The baseline processing time of job \(j\) on machine \(k\) is \(p_{jk}\) and each job has a due
date \( d_j \). Let \( e_{jk} \) be the deteriorating effect of job \( j \) on machine \( k \) and \( 0 \leq e_{jk} < 1 \) for all \( j \in N \) and \( k \in M \). In this research, we consider this general case (unrelated machines) and also the identical machines case, where \( p_{jk} = p_j \) and \( e_{jk} = e_j \) for all machines \( k \).

There are \( g \) possible positions in each machine, \( g = n - m + 1 \) (as each machine must be assigned at least one job), and let \( G \) be the set of positions. Let \( X_k \) be the ordered set of jobs assigned to machine \( k \), and \( x[h,k] \) be the job assigned to position \( h \) of machine \( k \). Let \( q_{kh} \) indicate the performance level of machine \( k \) for the job in position \( h \) and let \( q_{kh} = (1 - e_{jk} / q_{kh}) \) for each machine \( k \in M \). It is assumed that the machines will have no deterioration \( (q_{kh} = 1) \) at the beginning of the scheduling period and after maintenance is performed for all \( k \in M \). The actual processing time of the job \( x[h,k] \) on machine \( k \) is equal to 

\[
P_{d(h,k)|k} = \frac{p_{d(h,k)|k}}{q_{kh}}.
\]

The problem under consideration is the assignment and sequencing of jobs to the machines as to maximise the percentage of on-time jobs. This problem is equivalent to the problem of minimising the number of tardy jobs. Let \( c_j \) be the completion time of job \( j \) and \( u_j \) be a binary variable equal to 1 if \( c_j > d_j \), and 0 otherwise. Let the total number of late jobs be \( U_{sum} = \sum_{j \in N} u_j \), and the percentage of on-time jobs \( O_t = (n - U_{sum}) / n \).

The complexity of this problem with \( m > 1 \) is clearly NP-hard given the problem that assumes identical machines and no deterioration \( (P || U_{sum}) \) is known to be NP-hard (Ho and Chang, 1995).

The mathematical formulation for this problem is presented next. A MIP model is derived to schedule jobs to machines. The decision variable \( x_{jkh} \) is a binary variable that is equal to 1 if job \( j \) is assigned to machine \( k \) in position \( h \), 0 otherwise. The decision variable \( y_{kh} \) is a binary variable that is equal to 1 if maintenance is performed immediately after the job assigned to machine \( k \) in position \( h \). Let \( \ell_{kh} \) be the elapsed time between the completion time of the job placed in position \( h \) of machine \( k \) and the completion time of the last maintenance activity performed in machine \( k \). Define \( \pi_{kh} \) as the total processing time of all jobs in a processing period if the job assigned to position \( h \) is the last job processed in machine \( k \) and \( s_{kh} \) as the time duration between the completion of the job assigned to the \( h \) position and the starting time of the next maintenance activity. Let \( B \) be a big number, \( IM \) be the time interval between two consecutive maintenances, and \( t_{M} \) the time needed to perform a maintenance activity. Model MIP is now stated:

\[
MIP: \text{Min } z = \sum_{h \in G, k \in M} u_{kh} \tag{1}
\]

s.t.: \[
\sum_{j \in N} x_{jkh} = 1, \quad \forall h \in G, k \in M \tag{2}
\]

\[
\sum_{h \in G, k \in M} x_{jkh} = 1, \quad \forall j \in N \tag{3}
\]

\[
\ell_{k1} = \sum_{j \in N} p_{jk} x_{jk1}, \quad \forall k \in M \tag{4}
\]

\[
\ell_{k(h-1)} + \sum_{j \in N} \sum_{v=1}^{h} \left( p_{jv} / q_{kv} \right) x_{jkh} \leq \ell_{kh} + B y_{kh(h-1)}, \quad \forall k \in M, h \in G \setminus \{1\} \tag{5}
\]

\[
\sum_{j \in N} \sum_{v=1}^{h} \left( p_{jv} / q_{kv} \right) \leq \ell_{kh} + B \left( 1 - y_{kh(h-1)} \right), \quad \forall k \in M, h \in G \tag{6}
\]
Maximising the number of on-time jobs on parallel servers

\[ q_{kh} = \sum_{j \in N} (1 - e_{jk}) \times q_{kh(k-1)} \times x_{jk(k-1)} \times (1 - y_{kh(k-1)}) + y_{kh(k-1)}, \quad \forall k \in M, \ h \in G \setminus \{1\} \]

(7)

\[ q_{kl} = 1, \quad \forall k \in M \]

(8)

\[ \ell_{kh} \leq l_{kh}^M, \quad \forall h \in G, \ k \in M \]

(9)

\[ \pi_{kh} \geq \ell_{kh} - B(1 - y_{kh}), \quad \forall k \in M, \ h \in G \]

(10)

\[ s_{kh} \geq l_{kh}^M - \pi_{kh} - B(1 - y_{kh}), \quad \forall h \in G, \ k \in M \]

(11)

\[ c_{kh} = \sum_{v=1}^{h-1} \pi_{kv} - \sum_{v=1}^{h-1} s_{kv} + t_{kh} - \sum_{v=1}^{h-1} y_{kv} + \ell_{kh}, \quad \forall h \in G, \ k \in M \]

(12)

\[ t_{kh} \geq c_{kh} - \sum_{j \in N} x_{jkh} d_j, \quad \forall k \in M, \ h \in G \]

(13)

\[ t_{kh} \leq B u_{kh}, \quad \forall k \in M, \ h \in G \]

(14)

\[ \ell_{kh}, \pi_{kh}, s_{kh}, \geq 0, \quad \forall k \in M, \ h \in G \]

(15)

\[ x_{jkh}, y_{jkh}, u_{kh} \in \{0, 1\}, \quad \forall j \in N, \ k \in M, \ h \in G \]

(16)

In model MIP, equation (1) is the objective function which is equivalent to the percentage of on-time jobs. Equation (2) specifies that each position in each machine can be assigned at most one job, while equation (3) states that each job must be assigned just once to one position in one machine. Equations (4) to (6) specify the elapsed time of each job in the sequence. Equations (7) and (8) define the performance level of each machine for each job position. Equation (9) ensures that the elapsed time of each job is always less than or equal to the maintenance interval \( I^M \). Equations (10) and (11) are used to compute \( \pi_{kh} \) and \( s_{kh} \), which are needed to compute the completion time and tardiness of the jobs in equations (12) and (13). Finally, equation (14) specifies the value of \( u_{kh} \). The problem has \( nm(m + 2) \) binary variables, making it ‘computationally’ challenging as \( n \) and \( m \) increase.

3 Solution methods

The mixed integer programming formulation can be solved only for small size problems. However, for large practical problems commercial solvers may not give an optimal solution. Note that the problem is NP-hard as discussed in Section 2. In this section, therefore, we derive heuristic algorithms for the problem of maximising the percentage of on-time jobs in parallel-machines with sequence dependent deterioration and periodic maintenance activities. The loading of the jobs in the machines will follow one of two strategies: loading one machine at a time or loading all the machines simultaneously. These strategies are consistent with Ho and Chang (1995) who address the problem of minimising the number of late jobs in an environment with identical machines. The time interval between two consecutive maintenances is fixed and maintenance should be started at specified times.

The level deterioration of a machine depends on the jobs’ processing sequence. Consequently, a job being added to a machine will be considered for all possible positions in the machine processing sequence. From all the sequences that result in jobs
being *on-time*, the assignment that either minimises the load in the machine or the deterioration of the machine will be selected. In the following, we give a detailed description of the heuristics. We define a block as a set of jobs scheduled consecutively in a processing period which is a time interval between two consecutive maintenance activities. We use the following additional notation for the description.

\( Z \) set of jobs not yet assigned to the machines  
\( L \) set of late jobs  
\( Y \) set of temporarily un-assignable jobs  
\( M' \) set of current empty machines  
\( V_k \) set of temporary job sequences for machine \( k \)  
\( P_k \) total processing time of the not yet assigned jobs if they are performed in machine \( k \)  
\( Q_k \) performance level of machine \( k \) if it processes the set of not yet assigned jobs  
\( C_k \) current load of machine \( k \)  
\( q_k \) current performance level of machine \( k \) (at end of the last job in the machine)  
\( n_k \) number of on-time jobs currently assigned to machine \( k \)  
\( \alpha \) job under consideration  
\( \gamma \) machine being currently loaded  
\( b \) index for blocks  
\( b_k \) index for blocks for machine.

### 3.1 Heuristic SM

The SM (loading) heuristic loads one machine at a time with the goal of scheduling as many on-time jobs as possible. The first thing to consider when implementing this heuristic is to decide what machine to select first when they are not identical. We conducted several pilot experiments and among the approaches tested a rule based on deterioration provided the best performance. Therefore, when machines are not identical the machine with the largest cumulative deterioration is selected first for loading. Intuitively, the less efficient machine is loaded so a sequence of jobs is assigned that will maximise the machine performance. The second thing to consider is the selection order or rule for jobs loading. We consider four priority rules \( (\omega) \) to create an ordered list and they are listed next.

- \((\omega = p)\) listing the jobs in non-decreasing order of processing times \( p_{j\gamma} \)
- \((\omega = d)\) listing the jobs in non-decreasing order of due dates \( d_{j\gamma} \)
- \((\omega = e)\) listing the jobs in non-decreasing order of deterioration effects \( e_{j\gamma} \)
- \((\omega = r)\) listing the jobs in non-increasing order of processing time-deterioration effect ratios \( r_{j\gamma} = p_{j\gamma} / e_{j\gamma} \)
where $\gamma$ is the machine selected to be loaded. There are four versions of heuristic $SM$ ($SM-p$, $SM-d$, $SM-e$, $SM-r$) both for the identical and different machine problem. Each version will use one of the rules $\omega$ to order the unscheduled jobs.

Step 0  Let $L = \emptyset$. Let $M' = M$. Let $Z$ the set of all jobs in $N$ in any order. Let $Y$ be the set of pending jobs in $N$ in any order.

Step 1  Let $b = 1$. Let $P_k = \sum_{j \in Z} P_{jk}$ for all $k$ from $M'$ and $Q_k = \prod_{j \in Z} (1 - e_{jk})$ for all $k$ from $M'$.

Step 2  If processing times are identical for all machines, select any $\gamma$ from $M'$, else let $\gamma$ be the machine with $\min_{k \in M'} \{Q_k\}$. Remove $\gamma$ from $M'$ and set $n_\gamma = 0$ and $C_\gamma = 0$. Order $Z$ with respect the priority rule $\omega$.

Step 3  Remove the first job from $Z$. This is job $\alpha$.

Step 4  Let $V_\gamma$ be the set of $n_\gamma + 1$ temporary machine sequences generated by:
   a  inserting job $\alpha$ ahead of each of the $n_\gamma$ jobs in the original machine sequence
   b  adding job $\alpha$ at the end of the original machine sequence.

Step 5  Let $v^*$ be the machine sequence from $V_\gamma$ with all jobs on-time and least $C_\gamma$. If $v^*$ is found go to Step 6, else $L = L \cup \alpha$ and go to Step 7.

Step 6  If $C_\gamma \leq C_L^M$ assign $\alpha$ to machine $\gamma$ and set $n_\gamma = n_\gamma + 1$ and $h = n_\gamma$, else $Y = Y \cup \alpha$.

Step 7  If $Z \neq \emptyset$ go to Step 3, else go to Step 8.

Step 8  If $Y \neq \emptyset$, schedule a periodic maintenance activity after block $b$. Let $b = b + 1$, $C_\gamma = 0$, $Z = Y$, $q_\alpha = 1$ and go to Step 3. Else, go to Step 9.

Step 9  If $M' \neq \emptyset$ and $L \neq \emptyset$ then $Z = L$ and go to Step 1, else End.

3.2 Heuristic $AM$

The $AM$ (all machines loading) heuristic considers the performance of all machines when loading each individual job to find the ‘best’ position and machine to load each job. The first element to consider for this approach is to decide the order in which jobs are compared and loaded. With $m$ different machines, there are $m$ possible values (one per machine) for each job processing time, deterioration, and processing time to deterioration ratio. As explained in Section 3.1, the four job parameters $p_{jk}$, $d_j$, $e_{jk}$, and $r_{jk}$ can be used to generate a priority list. Based on pilot experiments, we use four $\omega$ rules to create the following four priority lists:

- $(\omega = p)$ listing the jobs in non-decreasing order by smallest processing times, i.e., by $p_j = \min_{k \in M} \{p_{jk}\}$
- $(\omega = d)$ listing the jobs in non-decreasing order of due dates $d_j$
- $(\omega = e)$ listing the jobs in non-decreasing order by smallest deterioration effects, i.e., by $e_j = \min_{k \in M} \{e_{jk}\}$
(\omega = r) listing the jobs in non-increasing order of processing time-deterioration effect ratios, i.e., by \( r_j = \min_{\omega \in M} \{ p_{j\omega} / e_{j\omega} \} \).

The same rules are used when machines are considered to be identical. Under those circumstances for each machine \( k \in M \), \( p_j = p_{j\omega} \), \( e_j = e_{j\omega} \), and \( r_j = p_{j\omega} / e_{j\omega} \). The second element to consider is the machine selection and job position when multiple machines are capable of completing the job on hand on-time. Pilot experiments showed that the position minimising the increase in machine load should be selected from those that result in the job being on-time. Four versions of the AM(AM-p, AM-d, AM-e, and AM-r) heuristic are considered for both unrelated and identical machines. The only difference between them is the rule \( \omega \) used to order the jobs. The steps of the AM heuristic are presented next.

**Step 0** Let \( L = \emptyset \). Let \( b_k = 1 \) for each \( k \in M \). Let \( n_k = 0 \) for each \( k \in M \). Let \( Z \) be an ordered list of jobs by rule \( \omega \). Let \( Y \) be the set of pending jobs in \( N \) in any order.

**Step 1** Remove the first job from \( Z \). This is job \( \alpha \).

**Step 2** For all \( k \in M \) let \( V_k \) be the set of \( n_k + 1 \) temporary machine sequences generated by
a) inserting job \( \alpha \) ahead of each of the \( n_k \) jobs in the original machine sequence and
b) adding job \( \alpha \) at the end of the original machine sequence.

Let \( \Sigma \) be the combination of all machine sequences sets \( V_k \), \( \forall k \in M \).

**Step 3** Let \( v^* \) be the machine sequence in machine \( k^* \) from \( \Sigma \) with all jobs on-time and least \( C_{k^*} \). Solve ties by minimum machine performance level \( q_k \). If \( v^* \) is found go to Step 4, else \( L = L \cup \alpha \) and go to Step 5.

**Step 4** If \( C_{k^*} \leq \frac{1}{n} \) assign \( \alpha \) to machine \( k^* \) and set \( n_{k^*} = n_{k^*} + 1 \) and \( b_k = n_{k^*} \), else \( Y = Y \cup \alpha \).

**Step 5** If \( Z = \emptyset \) go to Step 1, else go to Step 6.

**Step 6** If \( Y = \emptyset \), schedule a periodic maintenance activity after block \( b_k \). Let \( b_k = b_k + 1 \), \( C_k = 0 \), \( Z = Y \), \( q_{lk} = 1 \) and go to Step 1. Else, End.

The two heuristic approaches are polynomial in nature with a complexity of \( O(n^2 + nm) \).

### 4 Computational experiments

We now report computational results to evaluate the relative performance and robustness of the heuristics proposed in this paper. The experiments include four major factors [congestion ratios (CRs), range of the machine deterioration effects (e_range), number of machines (m), and heuristic (heu)] and two responses (average number of on-time jobs and the heuristics average error versus the best solution). We assume that the maintenance duration and maintenance intervals are fixed to 30 units of time and
180 units of time, respectively. We discuss the effect of different maintenance durations and maintenance intervals only for those heuristics showing the best performance. The experiments were performed on two sets of problem instances, each considering a different machine setting; identical and unrelated. The heuristics were coded in C++ using a personal computer with a 3.4 GHz processor and 4 GB RAM.

The CR provides the overall due date tightness. A higher CR will result in a higher number of late jobs. Consider \( D_{\text{max}} = \frac{p_{\text{ave}} \times n}{(m \times CR)} \) and let the due date of a job be determined by \( p_{j}^{\text{min}} + U(0, D_{\text{max}}) \) where \( p_{j}^{\text{min}} = \min_{k \in M} \{ p_{jk} \} \). The previous method guarantees that each job can be on-time if at least placed in one of the machines at the start of the schedule, therefore there are no inherent late jobs. We consider the CR at three levels which are independent to the machine setting. For identical machines, the CR levels are 0.5, 1, and 2 while for unrelated machines the CR levels are 2, 3.5, and 5. The CR levels were chosen based on results obtained from preliminary experiments with the goal of obtaining similar overall performance in terms of the percentage of late jobs.

We assume that machine processing times and deterioration effects follow a uniform distribution. Processing times range from 1 to 100 while for the deterioration effect two ranges are considered, 1% to 5% and 5% to 10%. We named the deterioration effect range variable \( e_{\text{range}} \). We consider the number of machines at three different levels, 5, 10, and 20. Finally, we consider eight heuristics for each machine setting, \( SM-p, SM-d, SM-e, SM-r, AM-p, AM-d, AM-e, \) and \( AM-r \). There are \( 3 \times 2 \times 3 \times 8 \) experimental levels for each machine setting (identical and unrelated). We consider 25 replications by level which produces a total of 1,800 instances. Instances are available at (to be included in the final manuscript).

Subsections 4.1 and 4.2 explain the findings of the computational results for the identical and unrelated machine settings respectively. For each machine setting, we present a sensitivity analysis on the impact of increasing maintenance duration and maintenance intervals in the heuristics results for both the identical and unrelated machine settings. We consider three levels for maintenance durations (30, 60, and 90) and two levels for maintenance intervals (180 and 300).

### 4.1 Results for the identical machine case

Table 2 presents the average percentage on-time results for the identical machine experiments and Table 3 presents the average error versus the best solution for the case in which maintenance duration equals 30 units and maintenance intervals equals 180 units. The error of a heuristic is computed using the following equation: \( \frac{\text{best replication value-heuristic value}}{\text{best replication value}} \). The error shows, for each experimental instance, the result of each heuristic against the best heuristic result.

Table 2 shows that, when \( CR = 0.5 \), the only heuristic that generates a schedule with all jobs on-time for all the instances is \( SM-d \) (i.e., 100% on Table 2 and 0.00 on Table 3). Although, \( SM-d \) is the heuristic with the best performance when \( CR = 0.5 \), the rest of the SM heuristic provide very good results as well (in terms of on-time percentages). This result is indicative of a significant slack on the due dates, therefore several scheduling strategies can result in good/optimal schedules.
Table 2  On time percentages results for the identical machine setting

<table>
<thead>
<tr>
<th>CR</th>
<th>e_range</th>
<th>m</th>
<th>SM-d</th>
<th>SM-p</th>
<th>SM-e</th>
<th>SM-r</th>
<th>AM-d</th>
<th>AM-p</th>
<th>AM-e</th>
<th>AM-r</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>100.0%</td>
<td>99.8%</td>
<td>100.0%</td>
<td>99.9%</td>
<td>98.2%</td>
<td>94.6%</td>
<td>92.7%</td>
<td>94.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>98.2%</td>
<td>94.6%</td>
<td>92.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>98.2%</td>
<td>94.6%</td>
<td>92.7%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>5</td>
<td>100.0%</td>
<td>98.8%</td>
<td>99.4%</td>
<td>99.1%</td>
<td>95.3%</td>
<td>90.7%</td>
<td>86.2%</td>
<td>90.9%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>100.0%</td>
<td>100.0%</td>
<td>99.9%</td>
<td>97.7%</td>
<td>91.9%</td>
<td>88.8%</td>
<td>92.0%</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>100.0%</td>
<td>97.7%</td>
<td>93.0%</td>
<td>89.7%</td>
<td>92.8%</td>
</tr>
<tr>
<td>1</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>88.9%</td>
<td>86.7%</td>
<td>85.1%</td>
<td>86.0%</td>
<td>89.7%</td>
<td>84.7%</td>
<td>75.5%</td>
<td>84.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>94.0%</td>
<td>88.4%</td>
<td>90.2%</td>
<td>89.2%</td>
<td>87.8%</td>
<td>84.2%</td>
<td>83.0%</td>
<td>83.4%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>99.0%</td>
<td>91.2%</td>
<td>90.0%</td>
<td>94.6%</td>
<td>95.2%</td>
<td>88.2%</td>
<td>90.8%</td>
<td>88.9%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>5</td>
<td>84.4%</td>
<td>77.6%</td>
<td>82.0%</td>
<td>78.1%</td>
<td>84.4%</td>
<td>81.4%</td>
<td>67.2%</td>
<td>80.1%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>84.2%</td>
<td>80.4%</td>
<td>82.0%</td>
<td>80.5%</td>
<td>81.7%</td>
<td>82.2%</td>
<td>71.2%</td>
<td>82.4%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>93.9%</td>
<td>85.6%</td>
<td>85.0%</td>
<td>87.4%</td>
<td>80.2%</td>
<td>83.2%</td>
<td>72.8%</td>
<td>83.7%</td>
</tr>
<tr>
<td>2</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>59.9%</td>
<td>60.0%</td>
<td>48.8%</td>
<td>57.6%</td>
<td>62.7%</td>
<td>61.2%</td>
<td>51.4%</td>
<td>61.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>65.5%</td>
<td>62.6%</td>
<td>57.3%</td>
<td>61.4%</td>
<td>66.3%</td>
<td>65.5%</td>
<td>59.5%</td>
<td>64.4%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>70.2%</td>
<td>63.9%</td>
<td>63.0%</td>
<td>66.4%</td>
<td>73.4%</td>
<td>71.5%</td>
<td>68.7%</td>
<td>71.2%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>5</td>
<td>51.4%</td>
<td>52.2%</td>
<td>49.8%</td>
<td>51.3%</td>
<td>60.2%</td>
<td>62.0%</td>
<td>45.4%</td>
<td>61.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>55.2%</td>
<td>54.4%</td>
<td>46.8%</td>
<td>54.7%</td>
<td>56.4%</td>
<td>61.3%</td>
<td>44.8%</td>
<td>62.4%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>65.6%</td>
<td>59.4%</td>
<td>60.0%</td>
<td>59.6%</td>
<td>60.6%</td>
<td>66.7%</td>
<td>61.3%</td>
<td>67.2%</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td>84.01%</td>
<td>81.16%</td>
<td>79.95%</td>
<td>81.42%</td>
<td>82.64%</td>
<td>80.96%</td>
<td>74.60%</td>
<td>80.95%</td>
</tr>
</tbody>
</table>

For $CR = 1$, the results show that the $SM-d$ heuristic is the best performer with an average percentage on-time of 90.7% with $AM-d$ as the second best performer with 86.5%. However, with $CR = 1$ the $AM-d$ heuristic provided the best performance for the experimental point $e_range = (1\%, 5\%)$ with five machines. In addition, the $AM-d$ heuristic tied with $SM-d$ providing the best performance when $e_range = (5\%, 10\%)$ with five machines. These two experimental results indicate that when job due dates are tighter and the number of machines are limited, loading the jobs considering all the machines is a better option that loading the jobs considering one machine at a time.

For $CR = 2$, the computational results show that the $AM-d$, $AM-p$, and $AM-r$ heuristics provide the best performance with an average percentage of on-time jobs of 63.3%, 65.0% and 64.9%, respectively. As noted in the experiments for $CR = 1$, the results show that as due dates become ‘tight’, considering all machines ($AM$) for jobs loading instead of the $SM$ strategy becomes a better alternative. For the group of experiments where $CR = 2$, heuristic $SM-d$ is no longer one of the best alternatives on average. In contrast most of the heuristics considering all machines loading provide better results in terms of the total number of jobs completed on-time.

Table 4 presents the analysis of variance (ANOVA) of the base case experiments for the identical machine setting. From the results, it can be noted that $CR$, $e_range$, $m$, $heu$, $CR \times e_range$, and $CR \times heu$ significantly affect the number of scheduled on-time jobs. As noted, all main variables are significant. $CR$, $e_range$, and $m$ are significant because...
of the increase in problem difficulty in terms of on-time percentage. As $CR$ and $e_{range}$ increased, the problem difficulty also increased; a higher $CR$ provides tighter due dates limiting the number of jobs to be schedule on-time, while at a higher $e_{range}$ represent a higher rate of job deterioration and therefore longer processing times, making more difficult to schedule jobs on-time. Similarly, fewer machines reduces the capacity to schedule jobs. In general, higher problem difficulty typically results in a larger difference in performance between the heuristics. The opposite being that for the easy problems, when the number of machines is large, the $CR$ is low, and the $e_{range}$ is small, all the heuristics perform well (small difference between them) thus the relative error would be close to 0 (for example, at $CR = 1$, $e_{range} = 1\%$ to $5\%$ and $m = 20$ the average error for all the heuristics is 0.3%). The fact that the $heu$ main effect is significant confirms the fact that not all heuristics perform the same. The $CR \times e_{range}$ and $CR \times heu$ interactions are key to solving the identical machine problem.

Table 3  Error results for the identical machine setting

<table>
<thead>
<tr>
<th>CR</th>
<th>$e_{range}$</th>
<th>m</th>
<th>SM-d</th>
<th>SM-p</th>
<th>SM-e</th>
<th>SM-r</th>
<th>AM-d</th>
<th>AM-p</th>
<th>AM-e</th>
<th>AM-r</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.5</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>0.0%</td>
<td>0.2%</td>
<td>0.0%</td>
<td>0.1%</td>
<td>0.1%</td>
<td>3.8%</td>
<td>5.8%</td>
<td>3.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.1%</td>
<td>0.1%</td>
<td>3.0%</td>
<td>5.1%</td>
<td>4.0%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>20</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>1.2%</td>
<td>0.5%</td>
<td>0.7%</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>0.0%</td>
<td>1.2%</td>
<td>0.6%</td>
<td>0.9%</td>
<td>0.1%</td>
<td>4.9%</td>
<td>9.6%</td>
<td>4.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.1%</td>
<td>0.0%</td>
<td>2.0%</td>
<td>6.0%</td>
<td>9.1%</td>
<td>5.9%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>20</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>5.8%</td>
<td>6.0%</td>
<td>5.9%</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>0.0%</td>
<td>10.5%</td>
<td>9.0%</td>
<td>9.8%</td>
<td>6.0%</td>
<td>7.0%</td>
<td>11.0%</td>
<td>5.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>0.0%</td>
<td>7.9%</td>
<td>7.0%</td>
<td>7.8%</td>
<td>3.1%</td>
<td>4.2%</td>
<td>7.0%</td>
<td>7.0%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>20</td>
<td>0.0%</td>
<td>9.0%</td>
<td>2.6%</td>
<td>8.1%</td>
<td>2.0%</td>
<td>6.0%</td>
<td>5.1%</td>
<td>5.0%</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td></td>
<td>3.0%</td>
<td>5.0%</td>
<td>6.0%</td>
<td>5.7%</td>
<td>3.8%</td>
<td>3.0%</td>
<td>8.0%</td>
<td>3.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>2.0%</td>
<td>4.0%</td>
<td>6.0%</td>
<td>5.0%</td>
<td>1.4%</td>
<td>2.6%</td>
<td>5.0%</td>
<td>4.2%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>20</td>
<td>0.0%</td>
<td>2.0%</td>
<td>1.8%</td>
<td>4.0%</td>
<td>0.2%</td>
<td>2.0%</td>
<td>2.9%</td>
<td>3.1%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>0.0%</td>
<td>4.2%</td>
<td>6.0%</td>
<td>8.1%</td>
<td>7.0%</td>
<td>5.9%</td>
<td>3.1%</td>
<td>9.0%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>10</td>
<td>0.0%</td>
<td>5.0%</td>
<td>17.1%</td>
<td>6.0%</td>
<td>2.0%</td>
<td>3.3%</td>
<td>6.0%</td>
<td>5.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>0.0%</td>
<td>5.0%</td>
<td>2.9%</td>
<td>4.0%</td>
<td>2.0%</td>
<td>3.0%</td>
<td>3.0%</td>
<td>4.0%</td>
</tr>
</tbody>
</table>

The results for the $CR \times e_{range}$ interaction showed that the $e_{range}$ effect is very small when the $CR$ is at low level and very large when $CR$ is at high level. The $CR \times heu$ interaction indicates that the $CR$ has a higher effect in the heuristic performance (i.e., maximise the number of on-time jobs). As discussed earlier in this section, as the $CR$ increases the heuristics considering all machines loading become a better alternative.
Table 4  ANOVA for the identical machine setting

<table>
<thead>
<tr>
<th>Source</th>
<th>Sum of squares</th>
<th>df</th>
<th>Mean square</th>
<th>F value</th>
<th>p-value</th>
<th>Prob &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>0.690</td>
<td>31</td>
<td>0.0220</td>
<td>23.13</td>
<td>&lt; 0.0001</td>
<td></td>
</tr>
<tr>
<td>A (CR)</td>
<td>0.470</td>
<td>2</td>
<td>0.2300</td>
<td>242.37</td>
<td>&lt; 0.0001</td>
<td></td>
</tr>
<tr>
<td>B (e_range)</td>
<td>0.032</td>
<td>1</td>
<td>0.0320</td>
<td>33.55</td>
<td>&lt; 0.0001</td>
<td></td>
</tr>
<tr>
<td>C (m)</td>
<td>0.061</td>
<td>2</td>
<td>0.0300</td>
<td>31.54</td>
<td>&lt; 0.0001</td>
<td></td>
</tr>
<tr>
<td>D (heu)</td>
<td>0.071</td>
<td>3</td>
<td>0.0240</td>
<td>24.42</td>
<td>&lt; 0.0001</td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td>0.010</td>
<td>2</td>
<td>0.0050</td>
<td>5.15</td>
<td>0.0102</td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>0.005</td>
<td>4</td>
<td>0.0014</td>
<td>1.42</td>
<td>0.2466</td>
<td></td>
</tr>
<tr>
<td>AD</td>
<td>0.034</td>
<td>6</td>
<td>0.0056</td>
<td>5.84</td>
<td>0.0002</td>
<td></td>
</tr>
<tr>
<td>BC</td>
<td>0.001</td>
<td>2</td>
<td>0.0006</td>
<td>0.63</td>
<td>0.5368</td>
<td></td>
</tr>
<tr>
<td>BD</td>
<td>0.003</td>
<td>3</td>
<td>0.0010</td>
<td>1.05</td>
<td>0.3822</td>
<td></td>
</tr>
<tr>
<td>CD</td>
<td>0.007</td>
<td>6</td>
<td>0.0011</td>
<td>1.15</td>
<td>0.3513</td>
<td></td>
</tr>
<tr>
<td>Residual</td>
<td>0.039</td>
<td>40</td>
<td>0.0010</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cor total</td>
<td>0.73</td>
<td>71</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.1.1 Effect of maintenance durations and intervals

This section presents an analysis for the performance of the algorithms under different maintenance durations and maintenance intervals. We compare the four algorithms that exhibited the best performance among the experiments which are: SM-d, SM-p, SM-r, and AM-d. We consider three levels for maintenance durations (30, 60, and 90) and two levels for maintenance intervals (180 and 300). The results are presented based on the three CR levels, as heuristic dominance is mostly related to this experimental factor. Figure 2 reports the on-time average percentages when CR = 0.5 for the considered maintenance durations and intervals. The two values in parenthesis on the x-axis represent the maintenance duration and the maintenance interval respectively. For instance, (30, 180) means 30 units of time for maintenance duration and 180 units of time for maintenance interval. The graph shows that when CR = 0.5 the heuristic with the best performance is SM-d. The results show that sorting the jobs by due date and loading the machines one at a time provides the best results for the identical machine setting when CR = 0.5.

Figure 3 depicts the on-time average percentages for CR = 1. The graph demonstrates that, on average SM-d, provides the schedules with the higher number of on-time jobs for most of the experiments. Therefore we can conclude that in the identical machine scenario, increasing the CR factor from 0.5 to 1 does not changes the recommendation in terms of the algorithm with the best performance.

Figure 4 depicts the performance of the algorithms when CR = 2. Heuristic SM-d no longer provides the highest on-time average percentages at this level of variable CR. In this case AM-d displays the best performance for most of the experiments, therefore we can conclude that sorting the jobs by due date and considering all the machines when loading each job provides better results than individual loading when due dates are very tight. Figure 4 also shows that as maintenance duration increases, the performance of all the algorithms decrease. In addition, the results show that having a longer maintenance interval (300 units of time) allows for larger number of jobs to be scheduled on-time since less time is spent on maintenance and more time available to process jobs.
Maximising the number of on-time jobs on parallel servers

Figure 2  Average on-time results (%) for the identical machines setting with $CR = 0.5$

Figure 3  Average on-time results (%) for the identical machines setting with $CR = 1$

Figure 4  Average on-time results (%) for the identical machines setting with $CR = 2$
4.2 Results for the unrelated machine case

Table 5 provides the results for the average percentage of on-time jobs and Table 6 provides the average error versus the best solution for the unrelated parallel machine problem experiments. When \( CR = 2 \) the results show that heuristic AM-d provides the best performance with an on-time average of 93.7% followed by heuristic SM-r with 87.9%. Heuristic AM-d provided the best results for most of the experimental instances with \( CR = 2 \). However, even though SM-d came in third place in terms of overall performance, there are four experimental points where this algorithm provided the best results. SM-d provided the best performance for experimental points with \( e_{range} = (1\%, 5\%) \) with 10 and 20 machines. The heuristic also provided the best results when \( e_{range} = (5\%, 10\%) \) with 20 machines. These results show that when due dates are ‘more relaxed’, and the number of machines is 20 (higher capacity), SM-d provides better results than AM-d.

<table>
<thead>
<tr>
<th>CR</th>
<th>( e_{range} )</th>
<th>m</th>
<th>SM-d</th>
<th>SM-p</th>
<th>SM-e</th>
<th>SM-r</th>
<th>AM-d</th>
<th>AM-p</th>
<th>AM-e</th>
<th>AM-r</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>73.8%</td>
<td>81.2%</td>
<td>60.0%</td>
<td>81.6%</td>
<td>93.0%</td>
<td>83.0%</td>
<td>78.0%</td>
<td>87.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>97.4%</td>
<td>90.6%</td>
<td>88.0%</td>
<td>89.7%</td>
<td>95.0%</td>
<td>85.5%</td>
<td>78.7%</td>
<td>88.2%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>100.0%</td>
<td>98.4%</td>
<td>93.7%</td>
<td>98.7%</td>
<td>96.0%</td>
<td>86.0%</td>
<td>83.3%</td>
<td>89.0%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>5</td>
<td>61.9%</td>
<td>71.0%</td>
<td>59.0%</td>
<td>73.3%</td>
<td>90.0%</td>
<td>80.0%</td>
<td>76.0%</td>
<td>85.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>77.1%</td>
<td>87.0%</td>
<td>86.0%</td>
<td>93.0%</td>
<td>98.0%</td>
<td>91.0%</td>
<td>77.0%</td>
<td>85.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>98.6%</td>
<td>97.0%</td>
<td>92.0%</td>
<td>98.0%</td>
<td>95.0%</td>
<td>81.5%</td>
<td>77.5%</td>
<td>84.2%</td>
</tr>
<tr>
<td>3.5</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>64.5%</td>
<td>66.3%</td>
<td>59.0%</td>
<td>64.8%</td>
<td>82.0%</td>
<td>77.3%</td>
<td>73.0%</td>
<td>77.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>81.9%</td>
<td>84.2%</td>
<td>81.0%</td>
<td>84.5%</td>
<td>79.0%</td>
<td>74.0%</td>
<td>79.0%</td>
<td>79.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>92.0%</td>
<td>95.2%</td>
<td>91.0%</td>
<td>94.2%</td>
<td>89.0%</td>
<td>81.0%</td>
<td>77.0%</td>
<td>80.0%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>5</td>
<td>52.5%</td>
<td>60.6%</td>
<td>57.6%</td>
<td>59.9%</td>
<td>79.0%</td>
<td>76.0%</td>
<td>71.0%</td>
<td>75.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>79.0%</td>
<td>83.0%</td>
<td>75.5%</td>
<td>78.9%</td>
<td>81.0%</td>
<td>62.0%</td>
<td>59.0%</td>
<td>63.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>90.0%</td>
<td>93.0%</td>
<td>90.3%</td>
<td>92.0%</td>
<td>87.0%</td>
<td>59.2%</td>
<td>58.4%</td>
<td>61.8%</td>
</tr>
<tr>
<td>5</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>58.3%</td>
<td>59.0%</td>
<td>56.0%</td>
<td>60.0%</td>
<td>65.0%</td>
<td>60.4%</td>
<td>61.0%</td>
<td>60.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>74.0%</td>
<td>80.2%</td>
<td>71.0%</td>
<td>73.8%</td>
<td>79.0%</td>
<td>63.0%</td>
<td>64.0%</td>
<td>62.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>90.8%</td>
<td>89.6%</td>
<td>84.9%</td>
<td>89.0%</td>
<td>81.0%</td>
<td>64.0%</td>
<td>65.0%</td>
<td>63.0%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>5</td>
<td>45.3%</td>
<td>50.4%</td>
<td>44.0%</td>
<td>51.2%</td>
<td>62.0%</td>
<td>58.0%</td>
<td>58.0%</td>
<td>59.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>70.6%</td>
<td>77.5%</td>
<td>61.6%</td>
<td>72.0%</td>
<td>69.0%</td>
<td>56.9%</td>
<td>50.8%</td>
<td>51.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>88.0%</td>
<td>89.0%</td>
<td>83.0%</td>
<td>87.0%</td>
<td>72.0%</td>
<td>53.7%</td>
<td>50.6%</td>
<td>54.4%</td>
</tr>
</tbody>
</table>

For \( CR = 3.5 \), the results show that the AM-d heuristic is the best performer with an average percentage on-time of 83.8% with SM-p as the second best performer with 80.4%. Heuristic AM-d provided the best performance for most of the experimental points; however, SM-p provided a better performance for half of the experiments. For example, heuristic SM-p provided the best performance when the \( e_{range} = (5\%, 10\%) \) with 10 and 20 machines. This result shows that sorting the jobs by processing time while loading one machine at a time, in some cases, can be a better option.
At \( CR = 5 \), the results show that the \( SM-p \) heuristic is the best performer with an average percentage on-time of 74.1% with \( AM-d \) as the second best performer with 71.3%. Heuristic \( SM-p \) provided the best performance for half of the experimental points; generating the best schedule for most of the experiments with 10 and 20 machines. For instance, \( SM-p \) provided the best performance when the \( e_{range} = (5\%,\ 10\%) \) with 10 and 20 machines. This result shows that the number of machines has a significant effect on the unrelated parallel machine setting.

### Table 6  Error results for the unrelated machine experiments

<table>
<thead>
<tr>
<th>( CR )</th>
<th>( e_{range} )</th>
<th>( m )</th>
<th>( SM-d )</th>
<th>( SM-p )</th>
<th>( SM-e )</th>
<th>( SM-r )</th>
<th>( AM-d )</th>
<th>( AM-p )</th>
<th>( AM-e )</th>
<th>( AM-r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>11.4%</td>
<td>10.0%</td>
<td>15.0%</td>
<td>12.0%</td>
<td>5.0%</td>
<td>13.0%</td>
<td>14.0%</td>
<td>15.2%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>0.7%</td>
<td>7.6%</td>
<td>12.0%</td>
<td>8.5%</td>
<td>3.0%</td>
<td>11.4%</td>
<td>13.0%</td>
<td>8.7%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>0.0%</td>
<td>1.6%</td>
<td>3.0%</td>
<td>1.3%</td>
<td>2.0%</td>
<td>1.0%</td>
<td>3.0%</td>
<td>5.0%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>5</td>
<td>13.0%</td>
<td>10.9%</td>
<td>17.0%</td>
<td>13.0%</td>
<td>6.0%</td>
<td>14.0%</td>
<td>18.4%</td>
<td>19.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>11.0%</td>
<td>8.4%</td>
<td>13.0%</td>
<td>9.5%</td>
<td>9.0%</td>
<td>13.0%</td>
<td>17.0%</td>
<td>11.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>1.4%</td>
<td>2.1%</td>
<td>3.6%</td>
<td>2.0%</td>
<td>4.3%</td>
<td>1.8%</td>
<td>4.1%</td>
<td>7.0%</td>
</tr>
<tr>
<td>3.5</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>6.9%</td>
<td>4.3%</td>
<td>9.0%</td>
<td>6.3%</td>
<td>10.0%</td>
<td>26.0%</td>
<td>25.8%</td>
<td>25.9%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>6.5%</td>
<td>4.6%</td>
<td>7.0%</td>
<td>6.0%</td>
<td>10.0%</td>
<td>26.0%</td>
<td>24.0%</td>
<td>25.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>4.6%</td>
<td>1.3%</td>
<td>6.0%</td>
<td>2.3%</td>
<td>10.0%</td>
<td>11.7%</td>
<td>18.0%</td>
<td>11.3%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>5</td>
<td>29.2%</td>
<td>19.1%</td>
<td>25.3%</td>
<td>19.7%</td>
<td>11.0%</td>
<td>31.5%</td>
<td>32.6%</td>
<td>28.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>19.0%</td>
<td>9.7%</td>
<td>17.8%</td>
<td>14.3%</td>
<td>15.0%</td>
<td>27.0%</td>
<td>24.1%</td>
<td>26.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>6.0%</td>
<td>0.8%</td>
<td>9.2%</td>
<td>3.0%</td>
<td>11.0%</td>
<td>13.5%</td>
<td>19.0%</td>
<td>16.0%</td>
</tr>
<tr>
<td>5</td>
<td>(1%, 5%)</td>
<td>5</td>
<td>14.7%</td>
<td>14.8%</td>
<td>17.0%</td>
<td>13.4%</td>
<td>12.0%</td>
<td>30.9%</td>
<td>30.7%</td>
<td>31.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>12.0%</td>
<td>6.0%</td>
<td>13.0%</td>
<td>9.8%</td>
<td>5.0%</td>
<td>29.8%</td>
<td>30.0%</td>
<td>29.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>2.2%</td>
<td>3.4%</td>
<td>8.5%</td>
<td>4.0%</td>
<td>4.0%</td>
<td>19.7%</td>
<td>19.0%</td>
<td>19.2%</td>
</tr>
<tr>
<td></td>
<td>(5%, 10%)</td>
<td>5</td>
<td>35.0%</td>
<td>27.9%</td>
<td>30.0%</td>
<td>27.1%</td>
<td>15.0%</td>
<td>31.2%</td>
<td>35.1%</td>
<td>30.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10</td>
<td>16.9%</td>
<td>9.7%</td>
<td>18.0%</td>
<td>15.9%</td>
<td>7.0%</td>
<td>30.0%</td>
<td>32.0%</td>
<td>31.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20</td>
<td>5.6%</td>
<td>5.0%</td>
<td>14.5%</td>
<td>5.0%</td>
<td>6.0%</td>
<td>22.0%</td>
<td>18.9%</td>
<td>20.0%</td>
</tr>
<tr>
<td>Overall</td>
<td></td>
<td>10.89%</td>
<td>8.18%</td>
<td>13.27%</td>
<td>9.60%</td>
<td>8.07%</td>
<td>19.64%</td>
<td>21.03%</td>
<td>19.97%</td>
<td></td>
</tr>
</tbody>
</table>

In Table 7, the ANOVA results for the unrelated machine setting case. The ANOVA table lists the significant factors which include \( CR, e_{range}, m, heu \), and \( CR \times heu \). The main effect of \( m \) is clear since as the number of machines increases, the higher the capacity and therefore more options are available to load jobs, and the number of on-time jobs increases. In the unrelated machine setting, the range of the machine deterioration effects (\( e_{range} \)) becomes a main factor because the deterioration is not the same for all machines. If the range of the machine deterioration effect increases, machines are going to deteriorate faster and the number of on-time jobs decreases. The main effect of \( CR \) and \( heu \) do not have much meaning by themselves because they are involved in significant interactions. The \( CR \times heu \) interaction indicates that the \( CR \) has a large effect in finding the heuristic that will maximise the number of on-time jobs. As discussed earlier in this section, as the \( CR \) increases the \( SM-p \) heuristic, considering SM loading, becomes a good alternative for some of the experimental cases.
Table 7  ANOVA for the unrelated machine setting

<table>
<thead>
<tr>
<th>Source</th>
<th>Sum of squares</th>
<th>df</th>
<th>Mean square</th>
<th>F value</th>
<th>p-value</th>
<th>Prob &gt; F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Model</td>
<td>0.800</td>
<td>25</td>
<td>0.0320</td>
<td>9.21</td>
<td>&lt; 0.0001</td>
<td></td>
</tr>
<tr>
<td>A (CR)</td>
<td>0.087</td>
<td>2</td>
<td>0.0440</td>
<td>12.53</td>
<td>&lt; 0.0001</td>
<td></td>
</tr>
<tr>
<td>B (e_range)</td>
<td>0.130</td>
<td>1</td>
<td>0.1300</td>
<td>37.14</td>
<td>&lt; 0.0001</td>
<td></td>
</tr>
<tr>
<td>C (m)</td>
<td>0.440</td>
<td>2</td>
<td>0.2200</td>
<td>63.08</td>
<td>&lt; 0.0001</td>
<td></td>
</tr>
<tr>
<td>D (heu)</td>
<td>0.014</td>
<td>3</td>
<td>0.0048</td>
<td>1.38</td>
<td>0.0261</td>
<td></td>
</tr>
<tr>
<td>AB</td>
<td>0.004</td>
<td>2</td>
<td>0.0019</td>
<td>0.54</td>
<td>0.5858</td>
<td></td>
</tr>
<tr>
<td>AC</td>
<td>0.026</td>
<td>4</td>
<td>0.0064</td>
<td>1.85</td>
<td>0.1361</td>
<td></td>
</tr>
<tr>
<td>AD</td>
<td>0.056</td>
<td>6</td>
<td>0.0093</td>
<td>2.67</td>
<td>0.0261</td>
<td></td>
</tr>
<tr>
<td>BC</td>
<td>0.019</td>
<td>2</td>
<td>0.0097</td>
<td>2.8</td>
<td>0.071</td>
<td></td>
</tr>
<tr>
<td>BD</td>
<td>0.027</td>
<td>3</td>
<td>0.0089</td>
<td>2.56</td>
<td>0.0666</td>
<td></td>
</tr>
<tr>
<td>Residual</td>
<td>0.160</td>
<td>46</td>
<td>0.0035</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cor total</td>
<td>0.96</td>
<td>71</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

4.2.1 Effect of maintenance durations and intervals

In this section, we study the effect of maintenance durations and intervals on the performance of the scheduling algorithms in the unrelated machine setting. As in Section 4.1, we consider the four algorithms that exhibited the best performance among the unrelated machine experiments which are: SM-d, SM-p, SM-r, and AM-d (same set of heuristics as for the identical machine case). As stated in Section 4.1, three levels for maintenance durations (30, 60, and 90) and two levels for maintenance intervals (180, 300) were considered. We report the average on-time percentages per CR across all experiments for each algorithm.

Figure 5  Average on-time results (%) for the unrelated machines setting with CR = 2

Figure 5 reports the on-time average percentages when CR = 2. The graph shows that at this level of CR the heuristic with the best performance is AM-d. The results show that sorting the jobs by due date and considering all the machines when loading each job
provide the best results for the unrelated machine setting when $CR = 2$ regardless of the maintenance interval and duration times. The results show that as maintenance duration increases the performance of the algorithms decrease. In addition, as in the identical parallel machine case, the results show that having a shorter maintenance interval (180 units of time) allows for larger number of jobs scheduled on-time when maintenance durations are 30 and 60. The results show that job degradation is significant for the unrelated machine setting since under shorter maintenance intervals the performance of the algorithms is better. Shorter maintenance intervals allow for quick repairs which result in having high performance machines most of the time.

Figure 6 reports the on-time average percentages for the unrelated machine setting when $CR = 3.5$. The graph shows a similar pattern to Figure 5. From Figure 6, it is easily observable that $AM-d$ provides the best performance for all the experiments. Therefore, we can conclude that increasing $CR$ from 2 to 3.5 does not affect the recommendation in terms of the algorithm with the best performance.

Figure 6  Average on-time results (%) for the unrelated machines setting with $CR = 3.5$

![Figure 6](image)

Figure 7  Average on-time results (%) for the unrelated machines setting with $CR = 5$

![Figure 7](image)
Figure 7 depicts the performance of the algorithms when $CR = 5$. At this level of $CR$, heuristic $AM-d$ no longer provides the highest on-time average percentages for all the experiments. Heuristic $AM-d$ provides a better performance when the maintenance interval equals 300, while $SM-p$ provides a better performance when the maintenance interval is 180. This graph shows that the maintenance interval is a significant factor when $CR = 5$ for the unrelated machine setting. For the particular experimental scenario of large maintenance intervals, large durations and tight due dates heuristic dominance may change, placing more importance on the selection of the ‘right’ scheduling approach to maximise on-time performance.

5 Conclusions

This paper investigates identical and unrelated parallel machine scheduling problems with sequence dependent machine deterioration – where the degradation of the machines, which in turn determines job process times, depends on the sequence of the jobs that it processes. Moreover, the paper considers maintenance intervals and durations, important elements in production planning when machines deteriorate. The paper focuses on maximising the percentage of on-time jobs, a critical customer service measure of performance.

Two heuristic approaches are proposed to solve the problems; the first procedure is based on loading one machine at a time, while the second considers all machines simultaneously. Several job ordering approaches are considered including sorting by due date, process time, and deterioration ratio. An extensive computational experiment is conducted to analyse the effect of several experimental factors including the number of machines, due date tightness, and deterioration effect. These experiments demonstrate that the heuristic and due date tightness factors are the most relevant. The experimental results indicate that for the identical machines case the approach of loading one machine at a time generates the best performing schedule under most conditions. On the other hand, the approach of loading all machines simultaneously with jobs ordered by due date generates the best schedules when machines are not identical. The results also demonstrate that due date tightness is an important factor in heuristic dominance, thus as the amount of slack shrinks it becomes more important to evaluate the performance of multiple heuristics. For example, in the case of unrelated machines as tightness increases, the $SM$ loading a time approach that loads by the job’s process times become ‘competitive’ and generates a larger percentage of the best schedules than loading all machines simultaneously.

The effect of the maintenance duration and intervals was also studied as part of the computational experiments. For both scenarios, identical and unrelated machines, we compared the performance of the four algorithms providing the best results (i.e., the higher number of jobs scheduled on time). We considered three levels of maintenance durations (30, 60, and 90 units of time) and two levels of maintenance intervals (180 and 300 units of time). The results showed that for the identical machine scenario and for $CR = 0.5$ and $CR = 1$, ordering the jobs by due dates and loading them one machine at the time provide the best performance. However, heuristic $SM-d$ does not provides the highest on-time average percentages when $CR = 2$. For $CR = 2$ $AM-d$ displays the best performance for most of the experiments which leads us to conclude that sorting the jobs
by due date while considering all the machines when loading jobs work better when due
dates are very tight.

For the unrelated machine scenario, the results show that sorting the jobs by due date
and considering all the machines when loading each job provide the best results for the
unrelated machine setting when \( CR = 2 \) and \( CR = 3.5 \), regardless of the maintenance
interval and duration times. The results show that as maintenance duration increases the
performance of the algorithms decrease. In addition, as in the identical parallel machine
case, the results show that having a shorter maintenance interval (180 units of time)
allows for larger number of jobs scheduled on-time when maintenance durations are 30
and 60. The results show that job degradation is significant for the unrelated machine
setting since under shorter maintenance intervals the performance of the algorithms is
better. Shorter maintenance intervals allow for quick repairs which result in having high
performance machines most of the time.

There are multiple research directions that can build on this work. While maintenance
activities can take the form used in this paper of a single simultaneous event for all
machines, other systems have a single resource that performs maintenance on the
machines at different intervals. Research that would consider scheduling of maintenance
on multiple machines with a ‘secondary’ resource would be of significant relevance and
complexity given the decision process must determine the order and timing of when the
maintenance will occur. Also, problems that consider maintenance ‘types’ would be
relevant, for example, as the maintenance duration increase, the performance level of the
machine increases up to being back to its ‘optimal level’. Thus, the decision process for
the secondary maintenance resource includes the order of maintenance operations and the
duration of these activities.

References


past-sequence-dependent setup times’, European Journal of Operational Research, Vol. 191,
No. 2, pp.387–592.

minimization of the number of late jobs for the parallel machines scheduling problem’,


Chen, W-J. (2009) ‘Minimizing number of tardy jobs on a single machine subject to periodic

time-dependent processing times’, European Journal of Operational Research, Vol. 152,

sum-of-logarithm-processing-times-based learning considerations’, Information Sciences,
Vol. 179, No. 18, pp.3127–3135.


Maximising the number of on-time jobs on parallel servers


