MAKESPAN MINIMIZATION WITH SEQUENCE DEPENDENT MACHINE DETERIORATION AND MAINTENANCE EVENTS

Journal:	*International Journal of Production Research*
Manuscript ID	TPRS-2016-IJPR-0126
Manuscript Type:	Original Manuscript
Date Submitted by the Author:	26-Jan-2016
Complete List of Authors:	Ruiz-Torres, Alex; University of Puerto Rico in Rio Piedras Paletta, Giuseppe; Università della Calabria, Dipartimento di Economia, Statistica e Finanza M'Hallah, Rym; University of Kuwait
Keywords:	SCHEDULING, MAINTENANCE SCHEDULING, PARALLEL MACHINES, SEQUENCE DEPENDENT SYSTEMS, MAKESPAN
Keywords (user):	

MAKESPAN MINIMIZATION WITH SEQUENCE DEPENDENT MACHINE DETERIORATION AND MAINTENANCE EVENTS

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

Giuseppe Paletta
Dipartimento di Economia Statistica e Finanza
Universita della Calabria,
87036 Arcavacata di Rende (CS), Italy
g.paletta@unical.it

Rym M'Hallah
Department of Statistics and Operations Research
Faculty of Science
Kuwait University
P.O. Box 5969
Safat 13060, Kuwait
rym.mhallah@ku.edu.kw

*** Corresponding Author**

Abstract: This paper addresses the minimal makespan parallel machine problem where machines are subject to preventive maintenance events of a known deterministic duration and the processing time of a job depends on its predecessors since the last maintenance of the machine. The paper proposes some dominance criteria for sequences of jobs assigned to a machine, and uses these criteria to design constructive heuristics to this NP-hard problem. The computational investigation determines the parameters that make a hard instance and studies the sensitivity of the heuristics to these parameters.

Keywords: machine maintenance; sequence-dependent processing time; parallel machines; minimal makespan;

1. INTRODUCTION AND SIGNIFICANCE

This paper addresses a complex production planning problem that considers machine deterioration and maintenance events. In many manufacturing set ups involving metal cutting/shredding, machine performance deteriorates as the machine is used. The heat generated by the cutting process alters the toughness, hardness, and wear resistance characteristics of the cutter, resulting in the cutter being worn and the cutting process taking much longer. To avoid such situations, the cutting tool is sometimes changed and the machine is allowed to cool down. Differently stated, the machine is sometimes subject to a preventive maintenance to avoid lengthening the processing time of the work in process and worsening the outgoing quality of the product. This paper studies the importance of planning for preventive maintenance in a multiple machine setting as to reduce the total completion time of its assigned jobs. A common key objective in manufacturing processes is the maximization of shop efficiency, which is assessed by the time required to complete all pending jobs.

When cutting/shredding a variety of metal parts, the machine performance decreases differently depending on the hardness of the material being processed. A softer/easier material would degrade the machine and its tools less than a harder/tougher material. As a machine degrades, the time required to cut/shred a set of parts increases thus it makes sense to run the softer materials first. However, for a particular type of problems and when maintenance is not considered, Ruiz-Torres et al. (2013) show that this is only true if all jobs have equal processing times.

This paper focuses on the minimal makespan parallel machine problem where a set of n jobs is to be scheduled on a set of m machines. The general case assumes jobs have unequal base processing times, and actual processing times that depend on the degree of deterioration of the machine at the starting time of the job. The degree of deterioration of the machine depends on the jobs that have been already processed. Each machine can be scheduled for any number of maintenance events and the maintenance events can be performed simultaneously in any of the machines. This assumes the maintenance is performed by the machine operators and not by a secondary resource.

Research that considers maintenance events in cases of parallel machines with deterioration is relatively scarce. Yang (2011) and Yang et al. (2012) consider the parallel machine problem where the processing time is a linear function of their start time and the objective is to minimize the total machine load taking into consideration the joint decisions of maintenance frequency and timing, and of the assignment and sequencing of the jobs on the machines. Yang (2011) deals with the identical parallel machine case (thus no difference in base processing times or deterioration effects between machines) while Yang et al. (2012) deals with the more general unrelated machine case. In terms of how the maintenance event is modeled, in Yang (2011), each machine must be maintained exactly once, and the duration of the maintenance event is the same for all machines. Contrarily, in Yang et al. (2012), each machine can be maintained multiple times, and the duration of the maintenance event depends on the machine. Hsu et al. (2013) consider the problem of scheduling unrelated parallel machines where the processing times depend on the job's

position in the machine. They minimize the flowtime and the total machine load. They suppose that at most one maintenance event can occur per machine, and that the length of the maintenance activity is a linear function of its starting time. Lee et al. (2013) study the power position version of the processing times and consider a single maintenance event per machine for the unrelated parallel machine problem. Yang et al. (2014) consider an unrelated parallel-machine scheduling problem that includes controllable processing times and maintenance (rate-modifying) activities. The process times of the jobs can be compressed by allocating a greater amount of a common resource to process the job, while multiple maintenance activities can be performed in each machine. The objective is to minimize a total cost function that is based on the total completion time and total job compressions. Wang et al. (2014) address the parallel machine case where a job's processing time depends on the maintenance event, and the time required to complete the maintenance event is a linear function of the time the maintenance activity is started. They show that the problem can be formulated as a linear assignment problem and solved in polynomial time with a known number of machines m. Ma et al. (2014, 2015) address parallel-machine scheduling problems with deteriorating process times and maintenance. They address three versions of the problem: the minimization of the total absolute deviation of job completion times, the minimization of the total load on all the machines, and the minimization of the total completion time, proposing a polynomial-time algorithm to solve each of the three versions.

The research proposed herein differs significantly from that in the literature. The processing time of a jobs does not depend on the time it starts or on its position in the schedule (as previously assumed), but on the particular set of jobs that precede it on the machine. It supposes that each machine can be subject to multiple maintenance activities, as in Yang et al. (2012), and that the constant time required to perform the maintenance event is machine-dependent. This research problem is significant as it contributes to the body of knowledge in operations management by considering a real life situation that is present in industrial production systems; yet previously unaddressed in the literature. In fact, research that considers the effect of sequence dependent deterioration is scarce and no work addresses how to plan maintenance events to alleviate this effect. Filling this gap in the literature is relevant as more complex decision making tools are developed to maximize the efficiency of production systems. To solve this problem, the paper proposes three heuristics that use different problem characteristics to assign jobs to machines and determine the timing of the maintenance events.

The remaining of the paper is organized as follows. Section 2 defines the problem, models it as a mixed integer program, explains how to compute the completion time of a job, and develops some valid dominance criteria. Section 3 describes three constructive heuristics. Section 4 presents the computational investigation, determines what makes an instance difficult, and compares the sensitivity of the heuristics to these parameters. Section 5 summarizes the paper and gives potential extensions.

2. PROBLEM DESCRIPTION AND SPECIAL CASES

Consider a set $N = \{1,\ldots,j,\ldots n\}$ of n independent jobs to be processed on a set $M=\{1,\ldots,k,\ldots m\}$ of m parallel machines. All the jobs are non-preemptive and available for processing at time zero. Each machine can process only one job at a time. Let t_k be the duration of a maintenance event that returns machine k to its initial performance state. Maintenance activities can only be initiated when a job completes its processing; thus, they cannot interrupt the processing of a job or block the machine. A machine is either busy processing a job or under maintenance (no inserted idle time is allowed). At the start of the schedule all machines are at their baseline state (in other words, at 100% performance level or 0% deterioration). Furthermore, each machine can be maintained multiple times during the schedule and maintenance events can occur simultaneously in any number of machines. Some basic notation is presented next. Let

- p_{jk}, $j\in N$, $k\in M$, be the baseline processing time of job j on machine k; and
- d_{jk}, $j\in N$, $k\in M$, the deteriorating effect of job j on machine k with $0 \le d_{jk} < 1$;

The problem under consideration is the assignment of jobs to the machines, the sequencing of the jobs on the machines, and the schedule of the maintenance activities so that the makespan or maximum completion time C_{max} of all the jobs is minimized, where $C_{max}=max_{k\in M}\{C_k\}$ and C_k is the sum of the actual processing times for the jobs assigned to the machine and the duration of the maintenance activities. This problem is NP-hard. In the absence of machine maintenance and deterioration, it reduces to the $R||C_{max}$ problem, which is NP-Hard (Pinedo, 2012).

The mathematical formulation for this problem is presented next. Let H be the possible number of positions in each machine where $H = \{1, 2, ..., 2(n - m) + 1\}$. The first decision variable x_{jkh}, $j\in N$, $k\in M$, $h\in H$, is a binary variable that is equal to 1 if job j is assigned to machine k in position h, 0 otherwise. The second decision variable s_{kh}, $k\in M$, $h\in H$, is a binary variable that is equal to 1 if there is a maintenance event in machine k in position h, 0 otherwise. The third decision variable is q_{kh}, $k\in M$, $h\in H$, which is the performance rate of machine k for the job in position h. Using these three decision variables, the problem can be modeled as follows.

$$\text{Minimize } z = C_{max} \qquad (1)$$

$$\sum_{j\in N} x_{jkh} + s_{kh} \le 1 \qquad \forall\, h\in H, k\in M \qquad (2)$$

$$\sum_{h\in H,\, k\in M} x_{jkh} = 1 \qquad \forall\, j\in N \qquad (3)$$

$$\sum_{j\in N,\, h\in H} p_{jk}/q_{kh} \times x_{jkh} + \sum_{h\in H} s_{kh} \times t_k \le C_{max} \qquad \forall\, k \in M \qquad (4)$$

$$x_{jkh} \le \sum_{l\in N} x_{lk(h-1)} + s_{k(h-1)} \qquad \forall\, j\in N, k\in M, h\in H\backslash\{1\} \qquad (5)$$

$$\sum_{j\in N} (1 - d_{jk}) \times q_{k(h-1)} \times x_{jk(h-1)} + s_{k(h-1)} = q_{kh} \qquad \forall\, h\in H\backslash\{1\},\ k\in M \qquad (6)$$

$$q_{k1} = 1 \qquad \forall\, k \in M \qquad (7)$$

$$x_{jkh} \in \{0, 1\} \qquad \forall\, j\in N, k\in M, h\in H \qquad (8)$$

$$s_{kh} \in \{0, 1\} \qquad k\in M, h\in H \qquad (9)$$

$$q_{kj} \ge 0 \qquad k\in M, h\in H \qquad (10)$$

Equation (1) is the objective function. Equation (2) states that at most one job or a maintenance event can be assigned to each position in each machine. Equation (3) states that each job must be assigned exactly once to one position in one machine. Equation (4) establishes the total load in each machine must be not greater than C_{max}, while Equation (5) guarantees continuous assignments. Equation (6) defines the performance level of each machine for each job position, where the performance level returns to 1 when there is a maintenance event. Equation (7) defines the initial performance level of each machine. Finally, Equations (8) and (9) establish the job and maintenance assignment variables binary while Equation (10) defines the performance rate variables positive. In this formulation the resulting number of maintenance events in the makespan machine(s) will be optimal; however that is not necessarily the case for the other machines. In other words, an optimal solution may have multiple unnecessary maintenance events in the non-makespan machine(s), for example a maintenance event could be scheduled with no subsequent jobs.

The rest of this section is organized as follows. Section 2.1 computes the actual total processing time of a group of jobs assigned to a machine between two maintenance events. Section 2.2 proves three lemmas. Lemma 1 gives a special case which requires a maintenance event while Lemmas 2 and 3 pinpoint two cases where no maintenance is to be scheduled. Section 2.3 provides two lemmas for the case of identical deterioration rates across the jobs assigned to a machine. Section 2.4 proposes an algorithm that computes the optimal number of maintenance events for the jobs assigned to a single machine in the presence of identical deterioration effects of jobs. These lemmas and algorithms serve as foundation to determine problem bounds and solution approaches described in Section 3.

2.1 Makespan of a group of jobs assigned to a machine with *w* maintenance events

This section explains three basic concepts. First it describes how a maintenance event divides the jobs assigned to a machine k into groups. Second it describes the natural bounds on the number of maintenance events to be scheduled on a machine k, $k \in M$, given the number of jobs assigned to it. Third, it describes how the actual total processing time of each group of jobs is computed.

Let n_k, $k \in M$, be the number of jobs assigned to machine k. As it is not useful to schedule a maintenance event before the machine starts or after it completes processing all the jobs, there can be at most $n_k - 1$ maintenance activities per machine k, $k \in M$ (after the job in position 1, 2,…, $n_k - 1$). It follows that there are at most $n_k - 1$ occurrences when a maintenance activity can be initiated on machine k, $k \in M$. Suppose machine k, $k \in M$, is maintained w times in a schedule, $1 \leq w \leq n_k - 1$. Then, the n_k jobs are divided in $(w+1)$ groups, where group $G_{k,i}$, $i=1,\ldots,w+1$, has $\eta_{k,i}$ jobs and $\eta_{k,1}+\eta_{k,2}+ \ldots\ldots+ \eta_{k,w}+ \eta_{k,w+1}=n_k$.

Let $x[h,i,k]$ be the job assigned to position h ($1 \leq h \leq n_{k,i}$) of group $G_{k,i}$ of machine k. Let $p_{x[h,i,k]}$ be the baseline processing time of job $x[h,i,k]$, $d_{x[h,i,k]}$ be the deteriorating effect of job $x[h,i,k]$, and $q_{x[h,i,k]}$ be the performance level of machine k for the job $x[h,i,k]$, where $q_{x[h,i,k]}=(1- d_{x[h-1,i,k]})\, q_{x[h-1,i,k]}$ for each position h, $h=2,\ldots, +\eta_{k,i}$, and $q_{x[1,i,k]}=1$ (which assumes

that the machines start with no deterioration). The actual processing time of the job $x[h,i,k]$ is equal to $p'_{x[h,i,k]} = p_{x[h,i,k]}/q_{x[h,i,k]}$.

The completion time, $C_k(w)$, of all the jobs assigned to machine k when w maintenance events are scheduled is equal to

$$\begin{aligned} C_k(w) &= w\, t_k + \sum_{i=1}^{w+1} \sum_{h=1}^{\eta_{k,i}} p'_{x[h,i,k]} \\ &= w\, t_k + \sum_{i=1}^{w+1} \sum_{h=1}^{\eta_{k,i}} p_{x[h,i,k]} \frac{1}{\prod_{l=1}^{h-1}(1-d_{x[l,i,k]})} \\ &= w\, t_k + \sum_{i=1}^{w+1} C_{k,i}(w), \end{aligned}$$

where $w\, t_k$ is the total duration of the w maintenance events and

$$C_{k,i}(w) = \sum_{h=1}^{\eta_{k,i}} p_{x[h,i,k]} \frac{1}{\prod_{l=1}^{h-1}(1-d_{x[l,i,k]})}$$

is the actual total processing time of the $\eta_{k,i}$ jobs of group $G_{k,i}$.

2.2 Special cases when a maintenance is (not) required

This section proposes three lemmas that reduce the search space for maintenance events. Lemma 1 indicates a trivial case where maintenance must be scheduled to reduce the processing time of a job. Lemmas 2 and 3 indicate two special cases where no maintenance should be scheduled.

Lemma 1. Consider a schedule for all the jobs assigned to machine k when w maintenance events are realized, then if there exists a job $x[h, i, k]$ (for $h > 1$ and $i = 1,\ldots,w + 1$) such that

$$p'_{x[h,i,k]} > p_{x[h,i,k]} + t_k,$$

then an additional maintenance event in the schedule immediately before this job decreases the objective function.

Proof. If $p'_{x[h,i,k]} > p_{x[h,i,k]} + t_k$ then by considering a new maintenance event between the jobs $x[h\text{-}1,i,k]$ and $x[h,i,k]$ on the ordered group $G_{k,i}$, we decrease the objective function. □

Lemma 1 is valid when we consider m machines and k is the makespan machine.

Lemma 2. Let $C_k^*(0)$ be the makespan of the optimal schedule on machine k when no maintenance event is realized, then if

$$C_k^*(0) - \sum_{j=1}^{n_k} p_{jk} \leq t_k,$$

it is not necessary consider a maintenance event in the schedule.

Proof. See proof of Corollary 1 (page 417) in Kuo and Yang (2008b). □

Lemma 3. Let $C_{k,i}(w)$ be the optimal actual total processing time of all the jobs assigned to group $G_{k,i}$. If

$$C_{k,i}(w) - \sum_{j \in G_{ki}} p_{jk} \leq t_k,$$

Then it is not necessary to consider a maintenance event in the schedule.

Proof. The extension of Lemma 2 to each group $G_{k,i}$ proves Lemma 3. □

2.3 Special case of identical deteriorating effect

This section considers the special case where all jobs have identical deteriorating effects regardless of their assigned machine; i.e., $d_{jk} = d$ for all jobs j. It proves two Lemmas. Lemma 4 proves that a schedule that satisfies the group balance principle of Kuo and Yang (2008a) is necessarily optimal to the problem at hand, while Lemma 5 gives an optimal sequencing of the jobs assigned to a machine for a given number of maintenance events.

The group balance principle of Kuo and Yang (2008a) stipulates that the number of jobs per group should be as equal as possible. That is, if there are n_k jobs to be assigned to (w+1) groups, the number of jobs in each group i, i=1,…,w+1, is either $\eta_{k,i}=\alpha$ or $\eta_{k,i}=\alpha+1$, where $\alpha = \lfloor n_k/(w+1) \rfloor$ be the largest integer smaller than or equal to $n_k/(w+1)$. The group balance principle is satisfied if $\eta_{k,i}-\eta_{k,o} \leq 1$ for i=1,.., w+1 and o= 1,..,w+1.

Lemma 4. When $d_{jk} = d$ for all jobs j, and w maintenance events are considered, it suffices to consider the group balance principle to obtain an optimal schedule on machine k.

Proof. When $d_{jk} = d$ for all jobs j, the machine load

$$\begin{aligned} C_k(w) &= w\, t_k + \sum_{i=1}^{w+1} \sum_{h=1}^{\eta_{k,i}} p'_{x[h,i,k]} \\ &= w\, t_k + \sum_{i=1}^{w+1} \sum_{h=1}^{\eta_{k,i}} p_{x[h,i,k]} \frac{1}{(1-d)^{h-1}} \\ &= w\, t_k + \sum_{i=1}^{w+1} C_{k,i}(w), \end{aligned}$$

where

$$C_{k,i}(w) = \sum_{h=1}^{\eta_{k,i}} p_{x[h,i,k]} \frac{1}{(1-d)^{h-1}}$$

is the actual processing time of all the jobs assigned to group $G_{k,i}$.

The actual processing time of a job on a machine k in the presence of w maintenances depends on its position in its assigned group and not in the group itself. Subsequently, its increase to $C_k(w)$ depends on the position of the inserted job. For any schedule that does not satisfy the group balance principle, there exists an alternative schedule that satisfies this principle. Let $G_{k,i}$ and $G_{k,o}$ be two groups that do not satisfy this principle, and suppose that $n_{k,i}-n_{k,o}>1$. Then $C_k(w)$ can be decreased by

$$\frac{p_{x[\eta_{k,i},i,k]}}{(1-d)^{\eta_{k,i}-1}} - \frac{p_{x[\eta_{k,i},i,k]}}{(1-d)^{\eta_{k,o}}} = \frac{p_{x[\eta_{k,i},i,k]}(1-(1-d)^{(\eta_{k,i}-\eta_{k,0}-1)})}{(1-d)^{\eta_{k,i}-1}} > 0$$

if the job in position $\eta_{k,i}$ of $G_{k,i}$ is removed and inserted in position $\eta_{k,o}$+1 of $G_{k,o}$. Continuing iteratively in this way will improve $C_k(w)$ until the difference in the number of jobs of each couple of groups is less than or equal to one. Thus the proof of the Lemma.

□

Lemma 4 does not apply to the general case of d_{jk} because the actual processing time of a job inserted at position h depends on the group in which it is inserted and therefore by the jobs that precede it. When the job in position $\eta_{k,i}$ of $G_{k,i}$ is removed and inserted in position $\eta_{k,o}$+1 ($n_{k,i}-n_{k,o}>1$) of $G_{k,o}$, $C_k(w)$ changes of

$$\frac{p_{x[\eta_{k,i},i,k]}}{\prod_{l=1}^{\eta_{k,i}-1}(1-d_{x[l,i,k]})} - \frac{p_{x[\eta_{k,i},i,k]}}{\prod_{l=1}^{\eta_{k,o}}(1-d_{x[l,o,k]})}.$$

This changes is advantageous only if

$$\prod_{l=1}^{\eta_{k,o}}(1-d_{x[l,o,k]}) > \prod_{l=1}^{\eta_{k,i}-1}(1-d_{x[l,i,k]}).$$

Lemma 5. Let $d_{jk} = d$ for all jobs assigned to machine k, and machine k has w maintenance events. Jobs in machine k are divided into $w+1$ groups $G_{k,1}$, …, $G_{k,w+1}$ according to the group balance principle. In this case there exists an optimal schedule in which the jobs are sequenced in non-increasing order of their baseline processing times (p_{jk}) and then arranged one by one to each group in turn. That is, the jobs of the sequence are assigned to a schedule from the first position of the first group to the first position of the last group, and then from the second position of the first group to the second position of the last group, and so on.

Proof. Let $r \equiv n_k \bmod(w+1)$ be the remainder of the division of n_k by $(w+1)$, and $\alpha = \lfloor n_k/(w+1) \rfloor$. Without any loss of generality, assume that there are $(\alpha+1)$ jobs in each of the first r groups, and α jobs in each of the other groups (i.e. the group balance principle is respected).

Hardy et al. (1934) stipulate that for two sequences of numbers x_i and y_i, the sum $\sum_i x_i y_i$ of products of the corresponding elements of the sequences is minimal if the sequences are monotonic in the opposite sense. Using this result to sequence the jobs in a non-increasing order of their baseline processing times and to assign them sequentially (from the first position of the first group to the first position of the last group, then from the second position of the first group to the second position of the last group, and so on) yields a machine load

$$
\begin{array}{lllllllll}
C_k(w) = & w\,t_k + & & & & & & & \\
 & p_{x[1,1,k]} & & +p_{x[1,2,k]} & & +\ldots+ & p_{x[1,w+1,k]} + & & + \\
 & p_{x[2,1,k]} & /(1-d) & +p_{x[2,2,k]} & /(1-d) & +\ldots+ & p_{x[2,w+1,k]} & /(1-d) & + \\
 & p_{x[3,1,k]} & /(1-d)^2 & +p_{x[3,2,k]} & /(1-d)^2 & +\ldots+ & p_{x[3,w+1,k]} & /(1-d)^2 & + \\
 & \vdots & & \vdots & & \vdots & \vdots & & \vdots \\
 & p_{x[\alpha,1,k]} & /(1-d)^{\alpha-1} & +p_{x[\alpha,2,k]} & /(1-d)^{\alpha-1} & +\ldots+ & p_{x[\alpha,w+1,k]} & /(1-d)^{\alpha-1} & + \\
 & p_{x[\alpha+1,1,k]} & /(1-d)^{\alpha} & +p_{x[\alpha+1,2,k]} & /(1-d)^{\alpha} & +\ldots+ & p_{x[\alpha+1,r,k]} & /(1-d)^{\alpha}. &
\end{array}
$$

$C_k(w)$ equals the sum of the constant $w\,t_k$ and of the products of each element of the following sequence of n_k numbers corresponding to the baseline processing times:

$p_{x[1,1,k]} \geq p_{x[1,2,k]} \geq \ldots \geq p_{x[1,w+1,k]} \geq p_{x[2,1,k]} \geq p_{x[2,2,k]} \geq \ldots \geq p_{x[2,w+1,k]} \geq \ldots \geq p_{x[\alpha,1,k]} \geq p_{x[\alpha,2,k]} \geq \ldots \geq p_{x[\alpha,w+1,k]} \geq p_{x[\alpha+1,1,k]} \geq p_{x[\alpha+1,2,k]} \geq \ldots \geq p_{x[\alpha+1,h,k]}$, with the corresponding element of the following sequence of n_k numbers corresponding to the deteriorating rates of the h-th position ($h = 1\ldots,\alpha+1$) of each of the $(w+1)$ groups

$$
\begin{array}{ll}
 & 1=1=\ldots=1 \\
\leq & 1/(1-d)=1/(1-d)=\ldots..=1/(1-d) \\
\leq & 1/(1-d)^2=1/(1-d)^2=\ldots..=1/(1-d)^2 \\
\leq & \ldots.\leq \\
\leq & 1/(1-d)^{\alpha-1}=1/(1-d)^{\alpha-1}=\ldots=1/(1-d)^{\alpha-1} \\
\leq & 1/(1-d)^{\alpha}=\ldots..=1/(1-d)^{\alpha}
\end{array}
$$

where

- the first set of w+1 numbers equaling 1 represents the deteriorating rates at the first position of each of the w+1 groups,
- the second set of w+1 numbers equaling 1/(1-d) represents the deteriorating rates at the second position of each of the w+1 groups,
- the third set of w+1 numbers equaling $1/(1-d)^2$ represents the deteriorating rates at the third position of each of the w+1 groups, …..,
- the α-th set of w+1 numbers equaling $1/(1-d)^{\alpha-1}$ represents the deteriorating rates at the α-th position of each of the w+1 groups, and
- the (α+1)-th set of r number equal to $1/(1-d)^{\alpha}$ represents the deteriorating rates at the (α+1)-th position of each of the r groups.

These two sequences are monotonic in the opposite sense. Thus, the Lemma is proved. □

Lemma 5 reduces the problem of scheduling n_k jobs on machine k to finding the optimal number w of maintenance events. Finding the optimal number w of maintenance events in the case of identical deterioration effects is achievable using Algorithm 1.

Algorithm 1

Step 1: Sequence the n_k jobs in a non-increasing order of their baseline processing times. Set w = 0, where w is the number of maintenance events, which divide the n_k jobs into (w+1) groups.

Step 2: If $\sum_{j=1}^{n_k} p_j \frac{1}{(1-d)^{j-1}} - \sum_{j=1}^{n_k} p_j \leq t_k$, then stop; the sequence is optimal.

Step 3: Set $w = w + 1$.

Step 4: For j=1,…,n_k, Set h=⌈j/(w+1)⌉

If $p_j \frac{1}{(1-d)^{h-1}} > p_j + t_k$ then go to Step 3.

Else

assign job j to group $G_{k,i}$ at position h, where $i = j$ mod(w+1) if the remainder of the division of j by (w+1) is different from 0, $i = w + 1$ otherwise; thus, satisfying the group balance principle.

End If

Step 5: Stop; the schedule is optimal.

Step 1 initializes the algorithm. Step 2 compute the makespan when w = 0 and applies Lemma 2, which indicates whether a given machine can have its makespan reduced if subject to a maintenance. Step 3 increases the number of maintenances if this is an improvement. Step 4 applies Lemma 5 to sequence the jobs and divide them into w +1 groups, and then checks whether Lemma 1 holds.

3. SOLUTION METHODS FOR IDENTICAL MACHINES

This section proposes three building heuristics that use some of the group concepts described in the previous sections. The solution methods will address the case of identical parallel machines, where $t_k=t$, $p_{jk} = p_j$ and $d_{jk} = d_j$ for any machine k, $k \in M$. An illustrative example is included in Appendix 1.

Additional Notation

ω_k	Number of maintenance events in machine k.
N_k	Set of jobs assigned to machine k.
r_j	Effect ratio of job j. $= p_j(1 - d_j)/d_j$.
z_j	Threshold deterioration for job j. $= t / (p_j + t)$
S	The current job and maintenance schedule.
G^*_i	Set of jobs sequenced by non-increasing order of r_j assigned to group i.
$G_{k,i}$	Set of jobs sequenced by non-increasing order of r_j assigned to group i of machine k.
G'_i	Groups i of jobs.
$C'(G)$	Actual total processing time of the jobs in group G.
$D'(G)$	Deterioration level at the end of processing all the jobs in ordered group G.

3.1 Building Heuristic 1(*BH1*)

This heuristic has two stages; the first stage loads the jobs in the machines without any maintenance, while the second stage schedules the maintenance events. The overall strategy is to iteratively improve on the sequence that determines the makespan by adding maintenance events. There are two sorting rules to order the jobs for the initial machine assignment (Step 2). The heuristic breaks ties arbitrarily.

Stage 1. This stage loads the jobs to the machine based on an ordered list. Jobs are loaded into the machine where the resulting completion time is smallest. This builds a schedule S with no maintenance events, thus $\omega_k = 0$, for all k, $k \in M$.

Step 1. Let $N^* = N$ and $I = \{1 \ldots m\}$.
Step 2. Let $G^*_i = \varnothing \ \forall i$, $i \in I$, and order the jobs in N^* by non-increasing order of p_j (second iteration by non-increasing order of r_j).
Step 3. Remove the first job from N^*: let this be job φ.
Step 4. Add job φ to each set $G^*_i \ \forall i$, $i \in I$.
Step 5. Let u^* be the group with ***min.*** $C'(G^*_i) \ \forall i$, $i \in I$.
Step 6. Remove job φ from each set $G^*_i \ \forall i$, $i \in I \backslash \{u^*\}$.
Step 7. If $N^* \neq \varnothing$ then Step 3.
Step 8. Let $N_i = G^*_i$ and $G_{i,1} = G^*_i \ \forall i$, $i \in I$. The schedule $S = \{G^*_{i,1} \ \forall i, i \in I\}$.

Steps 1 and 2 initialize the sets and order the jobs. Steps 3 and 4 select the first job in the list and load it in each of the groups. Step 5 determines which group has the smallest completion time when job φ has been added. Step 6 removes job φ from all other groups. The process continues until all jobs have been assigned to a group. Finally, Step 8 loads one group into each machine.

Stage 2. This stage iteratively adds maintenance events to the machine with the largest completion time until no improvement is achieved.

Step 9. Let $S_{best} = S$, k be the makespan machine of S, N_k the set of jobs assigned to k, and $C^* = C_k$.
Step 10. If Lemma 2 applies or $\omega_k = n_k - 1$ or $C^* - \omega_k t_k - \sum_{j \in N_k} p_j \leq t_k$ then End.
Step 11. Set $\omega_k = \omega_k + 1$ and let $I = \{1, \ldots, \omega_k + 1\}$, $N^* = N_k$.
Step 12. Let $G_{k,i} = \varnothing \ \forall i$, $i \in I$, and order the jobs in N^* by non-increasing r_j.
Step 13. Remove the first job from N^*: let this be job φ.
Step 14. Let $C'i = C'(G_{k,i}) \ \forall i$, $i \in I$.
Step 15. Add job φ to each set $G_{k,i} \ \forall i$, $i \in I$.
Step 16. Let u^* be the group with ***min.*** $[C'(G_{k,i}) - C'i\,] \ \forall i$, $i \in I$.
Step 17. Remove job φ from each set $G_{k,i} \ \forall i$, $i \in I \setminus \{u^*\}$.
Step 18. If $N^* \neq \varnothing$ then Step 13.
Step 19. Let $i = 1$, and empty the machine sequence.
Step 20. Schedule the jobs in $G_{k,i}$ at the end of the current sequence of machine k.
Step 21. If $i \leq \omega_k$ then $i = i + 1$, schedule a maintenance event at the end of the current schedule and return to Step 20.
Step 22. If $C_k < C^*$ then return to Step 9.
Step 23. $S = S_{best}$ and End.

Step 9 is used to determine the makespan machine and keep the current schedule as the best schedule found. Step 10 ends the process when one of three conditions prevails: the case of Lemma 2 where no single maintenance event reduces the completion time, the case where the maximum number of maintenance events has been scheduled, and the case where the time of an additional maintenance is equal to or greater than the maximum available reduction in processing time, a similar concept to that of Lemma 3. Step 12 initializes the sets and orders the jobs assigned to the makespan machine. Steps 13 to 18 iteratively assign the next "available" job in N^* to the group where its addition results in the smallest increase in the groups' completion time until the set N^* is empty. Step 19 empties the machine sequence and Steps 20 and 21 loads the groups and maintenance events into the machines. If the completion time of the makespan machine is reduced the process returns to Step 9, else it continues to Step 12 where the solution of the process is set to the best found.

3.2 Building Heuristic 2 (*BH2*)

This heuristic iteratively builds groups and then assigns these groups to the machines using the LPT rule. The strategy is to iteratively increase the number of groups, and therefore maintenance events in the schedule. The heuristic follows two iterations based on the rule used to sort the job list (Step 3). Furthermore there are two versions of this heuristic based on the rule used to select the job to group assignment (Step 6), *BH2-C* and *BH2-D*. The steps are described next (break any ties arbitrarily).

Step 1. Let $w^* = m$ and $C_{max\text{-}best} = \infty$.
Step 2. Let $I = \{1,\ldots,\ w^*\}$ and $N^* = N$.
Step 3. Let $G^*_i = \varnothing\ \forall i,\ i \in I$, and order the jobs in N^* by non-increasing p_j (second iteration: r_j).
Step 4. Remove the first job from N^*: let this be job φ.
Step 5. Add job φ to each set $G^*_i\ \forall i,\ i \in I$. Order the jobs in each set $G^*_i\ \forall i,\ i \in I$ by r_j.
Step 6. Let u^* be the group with
  a) ***min.*** $C'(G^*_i)\ \forall i,\ i \in I$
  b) ***min.*** $D'(G^*_i)\ \forall i,\ i \in I$.
Step 7. Remove job φ from each set $G^*_i\ \forall i,\ i \in I \backslash \{u^*\}$.
Step 8. If $N^* \neq \varnothing$ then Step 4.
Step 9. Order the groups by non-increasing $C'(G^*_i)$ and let Γ be the ordered set of groups.
Step 10. Remove the first m groups from Γ and assign each to one of the m machines.
Step 11. If $\Gamma = \varnothing$ then Step 15.
Step 12. Let k be the machine with the least load. Schedule a maintenance event at the end of the current schedule of machine k.
Step 13. Remove the next group in Γ and assign it to machine k.
Step 14. If $\Gamma \neq \varnothing$ then return to Step 12.
Step 15. If $C_{max}(S) < C_{max\text{-}best}$ then $C_{max\text{-}best} = C_{max}(S)$ and $S_{best} = S$.
Step 16. If $w^* < n - m$ then $w^* = w^* + 1$ and return to Step 2.
Step 17. $S = S_{best}$ and End.

Step 1 sets the number of unassigned groups to the number of machines and initializes the best solution result. Step 2 initializes the index for the unassigned groups and the set of jobs to be assigned (N^*). Step 3 initializes the unassigned group sets and sorts the jobs in N^*. Steps 4 to 8 iteratively assign the next "available" job in N^* to the group where its addition results in the smallest increase in the groups' actual total processing time until the set N^* is empty. Step 9 orders the groups by their actual total processing time and forms a set of groups. Steps 10 to 11 assign the first m groups from Γ to a machine each, and move to Step 15 if the list Γ is empty to process. When the number of groups in Γ is greater

than m, these groups are loaded into the machines by least loaded in Steps 12 to 14 until the list Γ is empty. Step 15 is used to determine whether a schedule with an improved makespan was generated and to keep that schedule. Step 16 determines if a new iteration will be performed where an additional group (and therefore an additional maintenance) should be considered.

3.3 Building Heuristic 3 (*BH3*)

This heuristic forms groups of jobs that will be processed successively on a machine without a maintenance event among them. That is, the sum of their processing times without a maintenance event is less than its counterpart if a maintenance event is scheduled between any pair of successive jobs of the group. This heuristic assigns each group to the least loaded machine, with the objective of balancing the workload of the machines.

Stage 1. This stage builds the schedule based on two ordered sets of all jobs.

Step 1. Let L be the jobs in N ordered by non-decreasing value of d_j and let Z be the jobs in N ordered by non-decreasing value of z_j. Let $i = 0$.

Step 2. Let $j1$ be the first job from L and $j2$ the first job from Z.

Step 3. If $d_{j1} < z_{j2}$ and $j1 \neq j2$, then let $i = i + 1$, assign $j1$ and $j2$ into group G'_i (with deterioration level $d_{G'i}$) and remove them from L and Z. If $d_{Gi} < \max_{j \in Z} \{z_j\}$, then insert G'_i in L and reorder L (considering G'_i as a job whose processing time is the sum of the actual processing times of the jobs assigned to G'_i and whose deterioration rate is $d_{G'i}$). Else, schedule G'_i on the least loaded machine. Go to Step 6.

Step 4 If $j1 = j2$, remove $j2$ from L. Go to Step 6.

Step 5 Remove $j2$ from L and Z, and schedule it on the least loaded machine.

Step 6. If $Z \neq \varnothing$, return to Step 2.

Step 7. Schedule the first job from L on the least loaded machine preceding it by a maintenance event, and remove it from L. If $L \neq \varnothing$, return to Step 7.

Stage 2. This stage tries to reduce the completion time of each machine by changing the positions of the maintenance events if need be.

Step 8. For each machine k, $k \in M$, and job j, $j \in N_k$, schedule a maintenance event before j if $p'_j > p_j + t$. Let w_k be the resulting number of maintenance events.

Step 9. For each group $G_{k,i}$, $i=1,...,w_k+1$, $k \in M$, insert a maintenance event if Lemma 3 does not hold.

Step 10. Compute the resulting C_k. If C_k is less than its counterpart in Step 7, adopt this schedule for machine k. If k is the makespan machine, update C_{max}.

Stage 3. This stage balances the load across the machines.

Step 11. Let m_1 denote the makespan machine and m_2 the least loaded machine. Set position $j_2 = n_{m2}$.

Step 12. Remove the last job scheduled on m_1. Denote this job j_1. Compute the completion time of m_1.

Step 13. Insert j_1 before the job, of N_{m2}, in position $j_2 \in N_{m2}$. Compute C_{m2} while applying Lemmas 1-3. If $C_{m2} < C_{max}$, adopt the new schedules for m_1 and m_2, update C_{max}, and return to Step 11. Else, set $j_2 = j_2 - 1$. If $j_2 \geq 1$, return to Step 13.

Step 1 sorts N^* according to d_j and according to z_j. Step 2 selects $j1$, the first element of L, and $j2$, the first element of Z. Step 3 makes $j1$ an immediate predecessor of $j2$ when $d_{j1} < z_{j2}$, and removes $j1$ and $j2$ from both L and Z. $G = (j1, j2)$ may be appended with additional jobs of Z when $d_G < max_{j \in Z}\{z_j\}$. When this is the case, Step 3 inserts G into L. Otherwise, it schedules G on the least loaded machine ensuring that it is preceded by a maintenance event if it is not the first group on the machine. Step 4 addresses the case $j2$ has to be preceded by a maintenance; it removes $j2$ from Z. Step 5 schedules $j2$ whose $z_{j2} \leq d_{j1}$ on the least loaded machine because it cannot be preceded by any other job, and removes it from both L and Z. Step 7 schedules, iteratively, each job of L on the least loaded machine and removes it from L.

Stage 2 tries to reduce the completion time of each machine by applying Lemmas 1 and 3. Step 8 computes the completion time of each machine when a job scheduled on machine k is preceded by a maintenance if its actual processing time is greater than or equal to the sum of the maintenance and baseline processing times. Step 9 checks if Lemma 3 holds for each group $G_{k,i}$ assigned to a machine k. Step 10 computes the completion time of k, and checks whether it improves the completion time obtained in Stage 1. When this is the case, it adopts the new schedule and its corresponding C_k.

Stage 3 tries to reduce the makespan by redistributing part of the load of the makespan machine m_1 to the least loaded machine m_2. Specifically, Step 11 identifies m_1 and m_2, *and sets* $j2 = n_{m2}$. Step 12 removes $j1$, the job $j \in N_{m1}$ with the largest p'_j from N_{m1}, and computes the new C_{m1}. Step 13 tries inserting $j1$ before $j2 \in N_{m2}$ and computes the resulting C_{m2}. If the maximum of C_{m1} and C_{m2} is less than C_{max}, then the swap of $j1$ and $j2$ is adopted, and the algorithm returns to Step 11. Otherwise, it considers the next position of $j1$ on $m2$, and repeats Step 13.

5. COMPUTATIONAL EXPERIMENTS

This section presents a set of experiments used to evaluate the performance of the heuristics and to understand the relationship between the production environment variables in the case of identical machines. Sections 5.1-5.3 present, respectively, the experimental framework, the assessment method, and the analysis of the results.

5.1 Experimental Framework

The experiments consider the case of identical machines; thus $p_{jk} = p_j$, $d_{jk} = d_j$, and $t_k = t$ for all machines k. The processing time for each job j is randomly generated using $U(1,100)$, a uniform distribution with range 1 to 100. We consider four experimental parameters: the

number of machines (m), the ratio of jobs to machines (n/m), the range of the deteriorating effects (d_{gen}), and the time required to perform a maintenance (t). The deteriorating effect d_j is randomly generated using a uniform distribution U (d_{min}, d_{max}) with range d_{gen} =(d_{min}, d_{max}). The levels considered for factor m are 2, 5, 10, and 20. The levels for the ratio n/m are 10, 15, and 20 (thus the values of n range between 20 and 400). The d_{gen} factor is considered at two levels (1%, 6%) and (5%, 10%), and the maintenance time is considered at two levels: 3 and 9 time units. For each set of parameter combination of m, n, and d_{gen}, 25 instances are generated; resulting in 600 problem instances. Four heuristics (*BH1*, *BH2-C*, *BH2-D*, and *BH3*) are used to generate schedules for each of these instances at the two levels of t.

5.2 Measures of Error

For each instance, let $C_{max:max}$ and $C_{max:min}$ be respectively the maximum and minimum makespan values from the schedules generated by the four heuristics; therefore, $C_{max:min}$ is the best solution. For a problem instance, the schedule generated by heuristic h with $C_{max:h}$ has a measure of *error1*=$C_{max:h}/C_{max:min} -1$ and a measure of *error2, set to* $(C_{max:max}-C_{max:h})/(C_{max:max}-C_{max:min})$ when $C_{max:max} - C_{max:min} > 0$, to 0 otherwise. For both error measures, at least one heuristic yields a 0% result (i.e., the heuristic obtaining the best solution). While *error1* is not bounded by a maximal value, *error2* is bounded by 100%, the error of the heuristic with the largest makespan for the instance.

5.3 Results

Tables 1 and 2 present the overall heuristics' results. Table 3 presents by experimental point the average makespan and the percentage of times each heuristic generated the best solution. Table 4 presents by experimental point the two error measures. The difference in makespan across the heuristic for a particular combination of problem parameters is about 2 time units, although the difference increases as m increases. For example, the set n/m = 10, *dgen* = (1%, 6%), and t = 3 has $C_{max:max}$ = 514.1 and $C_{max:min}$ = 512.5 when m = 2; which translates into a difference of 1.6. On the other hand, when m = 20, $C_{max:max}$ = 522 and $C_{max:min}$ = 518.4, corresponding to a difference of 3.6. The results regarding the percentage of best solutions found (%best) indicate all of the four heuristics generate a percentage of the best solutions across the experiments. However, it can be noted that *BH1* has a large number of 0%, indicating it did not generate any of the best solutions for the 25 replications of that experimental point (in particular at m = 10 and 20). The resulting values for *error1* are less than 1% across all heuristics, with *BH1* typically having the largest values. The results for *error2* range from 0 to 100% and it is corroborated that *BH1* is outperformed. There are several parameter combinations where *BH1* has an *error2* value of 100% indicating it always generated the worst solution for those 25 replications.

< Tables 1 & 2 >

Tables 3 and 4 present the results by experimental variable, Table 3 for C_{max} and *%best* results and Table 4 for the *error* measures. The lowest average makespan is generated by *BH2C*, although this result is related to some of the experimental variables. At $m = 2$, the performances of *BH2C* and *BH3* are similar in terms of makespan, but as m increases the average makespan generated by *BH2C* is lower than that generated by *BH3*. *BH2C* dominates in terms of makespan across all values of *n/m* and *dgen*. The variable t does seem to have an effect on the average makespan for the different heuristics. At $t = 3$, *BH3* outperforms the other heuristics, but at $t = 9$, *BH2C* outperforms all others.

< Tables 3 & 4 >

The results for *%best* show that while outperformed in terms of average makespan for $m = 2$ and $t = 9$, *BH1* generates a relatively significant percentage of the best solutions. The results for *%best* present a clearer picture of heuristic dominance across factors. For example, *BH2C* generates more than half of the best solutions for all instances with $m > 2$, *dgen* = (5%, 10%) and for $t = 9$. Similarly, *BH3* generates more than half the best solutions when $t = 3$. Thus, it can be claimed that there is a significant interaction between heuristic performance and the experimental variables.

The error measures corroborate some of the previous results. Overall, *BH2C* outperforms all other heuristics when considering both error measures. As m increases, the errors for *BH2C* decrease while for the runner up, *BH3*, both error measures increase. As *n/m* increases, *error1* decreases for all heuristics, while *error2* increases for three of them (conflicting result). This behavior is explained by the relationship between *n/m* and the overall makespan of the problem. As *n/m* increases, the makespan generally increases (at $n/m = 10$ the average makespan is about 532 while at $n/m = 20$ the average makespan is about 1070). Given *error1* has as denominator the makespan of the problem, as *n/m* increases, and assuming the numerator stays relatively the same, the errors will naturally decrease, even when in reality the performance has not improved. Therefore it is proposed that *error2* is a better indicator of the relative performance of the heuristics. Therefore when observing the results of *error2*, the only heuristic whose performance improves as *n/m* increases is *BH2D*, which forms groups based on the performance level of the group and not on the size of the load.

An Analysis of Variance was conducted using *error2* as the indicator variable (given that it is not affected by the size of the makespan). Four of the five main effects were significant at the 0.05 level: *heuristic*, m, t, and *n/m*. All two level interactions with the heuristic factor were highly significant (as documented and discussed previously). The interaction $t \times m$ and $n/m \times d_{gen}$ were significant at the 0.05 level as well. The main effect interactions are presented in Figure 1 (*BH1* is not included as it is outperformed at all levels of all variables). Clearly as m increases the relative performance of *BH2C* improves, while the other two remain flat. As *n/m* increases, the *error* of *BH2C* and *BH3* increase slightly, and the *error* of *BH2D* decreases significantly, thus it is believed that at higher levels of *n/m BH2D* could outperform the other approaches. As the maintenance time increased, the relative *error* of *BH3* increased significantly, thus this approach would not work well for problems with large values of t. Finally, considering the deterioration effects, the *error* of

BH3 does not seem to be affected by an increase in the deteriorations, whereas with a higher deterioration, *BH2C*'s *error* level increases and *BH2D*'s *error* decreases.

<Figure 1>

The presented heuristics construct a feasible schedule that assigns jobs to machines and plans maintenance events. These schedules may further be improved if fed to metaheuristics. Pilot experiments using basic neighborhood search methods such as pairwise interchanges of jobs between the makespan machine and the least loaded machine resulted in makespan reductions of several time units for the test problems. This is an area of future research.

6. CONCLUSIONS

This research addresses the scheduling of a set of jobs on parallel machines where machines are subject to deterioration caused by the jobs but can be restored to a 100% performance level if subject to a maintenance event whose duration is known. This scheduling problem is an important manufacturing problem that occurs in a variety of settings as for example when cutting / shredding metal parts. The paper models the problem as a mixed integer program, and presents some conditions that an optimal solution must satisfy. It then offers three constructive heuristics, and compares their relative performance as a function of the problem parameters. The heuristics can be coupled with search heuristics or applied to the non-identical machine case. Future research problems include the development of solution methods for the case of non-identical machines, and the consideration of independent maintenance resources, therefore only a limited number of maintenance events can happen simultaneously.

REFERENCES

Gupta, J.N.D., Gupta, S.K. 1988. Single facility scheduling with nonlinear processing times. Computers and Industrial Engineering 14: 387–394.

Hardy G.H., Littlewood J.E., Polya G. 1934. Inequalities. Cambridge University Press: London.

Hsu, C.J., Ji, M., Guo, J.Y., Yang, D.L. 2013. Unrelated parallel-machine scheduling problems with aging effects and deteriorating maintenance activities, Information Sciences, 253, 163–169.

Kang, L., Ng, C.T. 2007. A note on a fully polynomial-time approximation scheme for parallel-machine scheduling with deteriorating jobs. International Journal of Production Economics 109(1): 180-184.

Kuo, W.H., Yang, D.L. 2008. Parallel-machine scheduling with time dependent processing times, Theoretical Computer Science 393(1): 204-210.

Kuo W.H., Yang, D.L. 2008. Minimizing the makespan in a single machine scheduling problem with the cyclic process of an aging effect. Journal of the Operational Research Society 59: 416–420.

Luo, W., Ji M. 2015. Scheduling a variable maintenance and linear deteriorating jobs on a single machine. Inf. Process. Lett. 115(1): 33-39.

Luo, W., Cheng, T.C.E., Ji M. 2015. Single-machine scheduling with a variable maintenance activity. Computers & Industrial Engineering 79: 168-174.

Ma, W.M., Sun, L., Zeng, X.Q., Ning, L. 2014. Parallel-Machine Scheduling Problems with Past-Sequence-Dependent Delivery Times and Aging Maintenance. Mathematical Problems in Engineering, Article ID 865978, in press.

Ma, W.M., Sun, L., Liu, S.C., Wu, T.H. 2015. Parallel-Machine Scheduling with Delivery Times and Deteriorating Maintenance. Asia-Pacific Journal of Operational Research, Article ID 1550029.

Pinedo, M.L. 2012. Scheduling Theory, Algorithms, and Systems. Springer Verlag, New York.

Ruiz-Torres, A.J., Paletta, G., Pérez, E. 2013. Parallel machine scheduling to minimize the makespan with sequence dependent deteriorating effects. Computers & Operations Research, 40(8), 2051-2061.

Toksari, M.D., Güner, E. 2010. The common due-date early/tardy scheduling problem on a parallel machine under the effects of time-dependent learning and linear and nonlinear deterioration. Expert Systems with Applications 37 (1): 92-112.

Wang, L.Y., Huang, X., Ji P., Feng, E.M. 2014. Unrelated parallel-machine scheduling with deteriorating maintenance activities to minimize the total completion time. Optimization Letters 8(1): 129-134.

Yang, D. L., Cheng, T. C. E., & Yang, S. J. 2014. Parallel-machine scheduling with controllable processing times and rate-modifying activities to minimise total cost involving total completion time and job compressions. International Journal of Production Research, 52(4), 1133-1141.

Yang, S.J. 2011. Parallel machines scheduling with simultaneous considerations of position-dependent deterioration effects and maintenance activities. Journal of the Chinese Institute of Industrial Engineers, 28(4), 270–280.

Yang, D.L., Cheng, T., Yang, S.J., Hsu, C.J. 2012. Unrelated parallel-machine scheduling with aging effects and multi-maintenance activities. Computers & Operations Research, 39(7), 1458–1464.

Table 1. Experiment Results: Makespan and percent of best solutions found.

				Cmax				*% best*			
m	***n/m***	***dgen***	***t***	***BH1***	***BH2C***	***BH2D***	***BH3***	***BH1***	***BH2C***	***BH2D***	***BH3***
2	10	(1%,6%)	3	514.0	512.7	514.1	512.5	12%	28%	4%	56%
			9	526.5	525.7	527.4	526.1	24%	32%	16%	28%
		(5%, 10%)	3	526.9	527.4	527.2	525.7	12%	0%	32%	56%
			9	549.2	548.5	549.0	549.0	24%	20%	28%	28%
	15	(1%,6%)	3	776.2	774.7	775.5	774.3	8%	16%	20%	56%
			9	797.9	796.7	797.9	798.4	20%	52%	24%	4%
		(5%, 10%)	3	780.8	780.4	780.4	779.1	24%	4%	8%	64%
			9	816.7	815.6	815.5	816.3	36%	20%	40%	4%
	20	(1%,6%)	3	1050.0	1049.2	1049.9	1048.4	8%	12%	28%	52%
			9	1080.9	1079.8	1080.2	1081.5	24%	48%	20%	8%
		(5%, 10%)	3	1067.4	1066.9	1066.0	1065.3	12%	0%	20%	68%
			9	1117.4	1116.2	1115.7	1117.1	20%	20%	56%	4%
5	10	(1%,6%)	3	519.7	517.3	519.0	517.3	4%	48%	4%	44%
			9	531.9	530.3	534.1	531.3	16%	76%	0%	8%
		(5%, 10%)	3	527.5	525.5	526.4	524.9	0%	16%	4%	80%
			9	549.6	547.3	548.6	548.8	8%	72%	12%	8%
	15	(1%,6%)	3	768.0	765.5	767.8	765.8	4%	60%	4%	32%
			9	789.4	786.9	789.8	789.8	4%	92%	4%	0%
		(5%, 10%)	3	795.0	792.6	793.2	791.6	4%	4%	0%	92%
			9	830.6	827.8	828.6	829.6	12%	60%	28%	0%
	20	(1%,6%)	3	1049.0	1046.9	1049.3	1047.1	0%	56%	16%	28%
			9	1079.2	1077.2	1079.4	1079.9	12%	80%	8%	0%
		(5%, 10%)	3	1041.8	1039.3	1038.5	1038.0	0%	0%	36%	64%
			9	1090.6	1087.4	1088.1	1089.2	4%	72%	24%	0%
10	10	(1%,6%)	3	529.4	526.3	528.9	526.2	0%	44%	4%	52%
			9	542.1	539.7	543.6	541.3	12%	80%	4%	4%
		(5%, 10%)	3	534.0	530.7	531.8	530.2	0%	20%	4%	76%
			9	555.8	552.5	555.1	554.4	0%	92%	0%	8%
	15	(1%,6%)	3	782.3	779.4	782.3	779.8	0%	72%	8%	20%
			9	803.8	800.9	804.1	804.1	4%	96%	0%	0%
		(5%, 10%)	3	800.8	797.9	798.7	797.1	0%	0%	12%	88%
			9	836.5	833.3	834.8	836.1	0%	96%	4%	0%
	20	(1%,6%)	3	1046.7	1044.1	1046.1	1044.4	0%	60%	4%	36%
			9	1076.7	1073.8	1076.3	1077.5	4%	92%	4%	0%
		(5%, 10%)	3	1058.7	1056.7	1056.2	1055.5	0%	0%	24%	76%
			9	1108.2	1105.3	1106.1	1108.2	0%	72%	28%	0%
20	10	(1%,6%)	3	522.0	518.4	521.3	518.6	0%	64%	0%	36%
			9	534.5	531.9	536.7	533.0	4%	68%	12%	16%
		(5%, 10%)	3	527.6	523.8	525.0	523.6	0%	24%	0%	76%
			9	549.1	545.5	547.9	548.6	0%	88%	12%	0%
	15	(1%,6%)	3	795.2	792.4	794.7	792.9	0%	80%	0%	20%
			9	816.7	813.8	817.4	817.2	0%	100%	0%	0%
		(5%, 10%)	3	794.4	791.0	791.4	790.5	0%	12%	12%	76%
			9	829.9	826.1	827.7	830.2	0%	96%	4%	0%
	20	(1%,6%)	3	1047.0	1044.0	1046.1	1044.5	0%	80%	0%	20%
			9	1077.2	1074.0	1076.8	1078.6	0%	100%	0%	0%
		(5%, 10%)	3	1044.0	1041.0	1040.6	1040.4	0%	0%	40%	60%
			9	1092.5	1088.8	1089.7	1092.6	0%	84%	16%	0%

Table 2. Experiment Results: Error measures.

				Error1				*Error 2*			
m	***n/m***	***dgen***	***t***	***BH1***	***BH2C***	***BH2D***	***BH3***	***BH1***	***BH2C***	***BH2D***	***BH3***
2	10	(1%,6%)	3	0.35%	0.10%	0.39%	0.07%	65%	21%	72%	17%
			9	0.26%	0.11%	0.44%	0.19%	46%	21%	68%	27%
		(5%, 10%)	3	0.27%	0.37%	0.36%	0.05%	39%	70%	45%	19%
			9	0.25%	0.12%	0.23%	0.22%	40%	28%	50%	50%
	15	(1%,6%)	3	0.30%	0.10%	0.21%	0.05%	81%	32%	50%	18%
			9	0.21%	0.06%	0.22%	0.27%	54%	15%	48%	61%
		(5%, 10%)	3	0.24%	0.20%	0.18%	0.03%	55%	69%	42%	9%
			9	0.22%	0.11%	0.10%	0.20%	51%	37%	38%	60%
	20	(1%,6%)	3	0.19%	0.10%	0.18%	0.03%	53%	42%	58%	17%
			9	0.14%	0.04%	0.08%	0.20%	49%	19%	33%	72%
		(5%, 10%)	3	0.22%	0.17%	0.09%	0.02%	71%	68%	36%	7%
			9	0.18%	0.08%	0.03%	0.16%	64%	36%	20%	58%
5	10	(1%,6%)	3	0.53%	0.06%	0.39%	0.06%	70%	9%	70%	15%
			9	0.36%	0.05%	0.78%	0.24%	38%	4%	80%	31%
		(5%, 10%)	3	0.51%	0.13%	0.31%	0.02%	82%	28%	56%	9%
			9	0.46%	0.05%	0.29%	0.32%	69%	15%	50%	56%
	15	(1%,6%)	3	0.35%	0.03%	0.34%	0.07%	75%	6%	60%	18%
			9	0.33%	0.02%	0.38%	0.38%	65%	3%	71%	71%
		(5%, 10%)	3	0.44%	0.14%	0.21%	0.00%	86%	36%	43%	2%
			9	0.36%	0.02%	0.12%	0.25%	78%	5%	27%	62%
	20	(1%,6%)	3	0.23%	0.03%	0.26%	0.05%	74%	11%	58%	24%
			9	0.20%	0.01%	0.22%	0.27%	56%	6%	62%	81%
		(5%, 10%)	3	0.38%	0.14%	0.05%	0.01%	98%	45%	20%	3%
			9	0.31%	0.01%	0.08%	0.17%	83%	4%	22%	59%
10	10	(1%,6%)	3	0.62%	0.05%	0.56%	0.04%	82%	7%	73%	5%
			9	0.49%	0.04%	0.80%	0.34%	57%	7%	80%	42%
		(5%, 10%)	3	0.71%	0.09%	0.30%	0.01%	95%	17%	51%	3%
			9	0.59%	0.00%	0.48%	0.35%	73%	1%	68%	51%
	15	(1%,6%)	3	0.37%	0.01%	0.38%	0.06%	82%	4%	73%	14%
			9	0.36%	0.00%	0.40%	0.39%	66%	0%	71%	72%
		(5%, 10%)	3	0.47%	0.10%	0.20%	0.01%	100%	26%	44%	1%
			9	0.38%	0.00%	0.18%	0.34%	84%	2%	41%	76%
	20	(1%,6%)	3	0.26%	0.01%	0.21%	0.04%	85%	4%	72%	14%
			9	0.27%	0.00%	0.24%	0.35%	67%	1%	63%	88%
		(5%, 10%)	3	0.31%	0.12%	0.07%	0.01%	97%	43%	26%	3%
			9	0.28%	0.01%	0.08%	0.27%	80%	3%	30%	76%
20	10	(1%,6%)	3	0.73%	0.02%	0.59%	0.06%	89%	3%	76%	8%
			9	0.52%	0.04%	0.94%	0.25%	60%	9%	79%	26%
		(5%, 10%)	3	0.77%	0.06%	0.28%	0.01%	99%	7%	38%	2%
			9	0.67%	0.01%	0.45%	0.58%	83%	1%	57%	72%
	15	(1%,6%)	3	0.37%	0.01%	0.30%	0.08%	91%	2%	76%	22%
			9	0.36%	0.00%	0.45%	0.42%	67%	0%	74%	77%
		(5%, 10%)	3	0.51%	0.07%	0.13%	0.01%	100%	18%	30%	2%
			9	0.46%	0.00%	0.19%	0.49%	81%	0%	37%	86%
	20	(1%,6%)	3	0.29%	0.01%	0.20%	0.05%	94%	2%	68%	20%
			9	0.30%	0.00%	0.26%	0.43%	65%	0%	60%	92%
		(5%, 10%)	3	0.37%	0.09%	0.04%	0.02%	100%	25%	14%	8%
			9	0.35%	0.01%	0.09%	0.36%	86%	2%	24%	86%

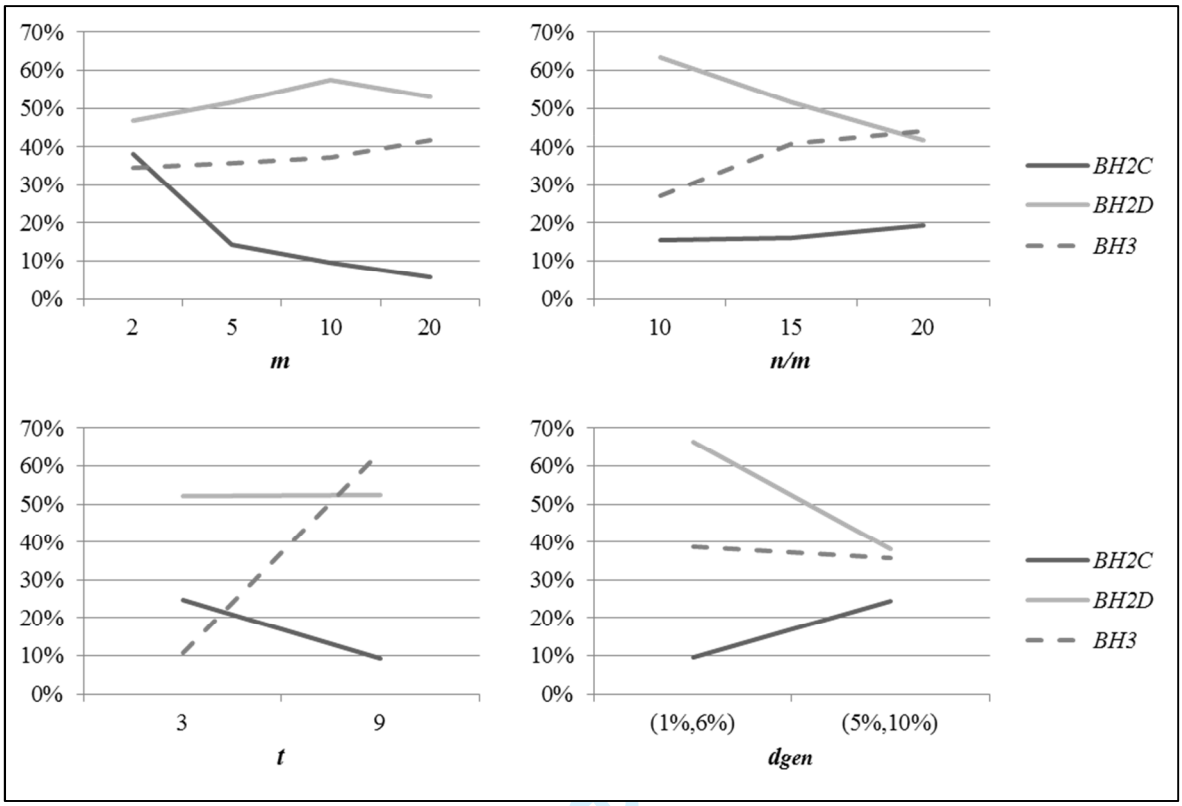
Table 3. Experimental Results by Variable: Makespan and percent of best solutions.

		Cmax				*%best*			
Parameter	***Level***	***BH1***	***BH2C***	***BH2D***	***BH3***	***BH1***	***BH2C***	***BH2D***	***BH3***
m	2	800.3	799.5	799.9	799.5	19%	21%	25%	36%
	5	797.7	795.3	796.9	796.1	6%	53%	12%	30%
	10	806.2	803.4	805.3	804.6	2%	60%	8%	30%
	20	802.5	799.2	801.3	800.9	0%	66%	8%	25%
n/m	10	533.7	531.5	533.5	532.0	7%	48%	9%	36%
	15	800.9	798.4	800.0	799.5	7%	54%	11%	29%
	20	1070.5	1068.2	1069.1	1069.3	5%	49%	20%	26%
dgen	(1%, 5%)	794.0	791.7	794.1	792.9	7%	64%	8%	22%
	(6%, 10%)	809.4	807.0	807.6	807.6	7%	36%	19%	39%
t	3	787.4	785.2	786.3	784.7	4%	29%	12%	55%
	9	815.9	813.5	815.4	815.8	10%	71%	14%	5%
	overall	**801.7**	**799.3**	**800.9**	**800.3**	**7%**	**50%**	**13%**	**30%**

Table 4. Experimental Results by Variable: Error measures.

		error1				*error2*			
parameter	***level***	***BH1***	***BH2C***	***BH2D***	***BH3***	***BH1***	***BH2C***	***BH2D***	***BH3***
m	2	0.24%	0.13%	0.21%	0.12%	56%	38%	47%	35%
	5	0.37%	0.06%	0.29%	0.15%	73%	14%	52%	36%
	10	0.43%	0.04%	0.32%	0.18%	80%	10%	58%	37%
	20	0.48%	0.03%	0.33%	0.23%	85%	6%	53%	42%
n/m	10	0.51%	0.08%	0.47%	0.19%	68%	16%	63%	27%
	15	0.36%	0.05%	0.25%	0.19%	76%	16%	52%	41%
	20	0.27%	0.05%	0.14%	0.15%	76%	19%	42%	44%
dgen	(1%, 5%)	0.35%	0.04%	0.38%	0.18%	68%	10%	66%	39%
	(6%, 10%)	0.40%	0.09%	0.19%	0.16%	79%	24%	38%	36%
t	3	0.41%	0.09%	0.26%	0.04%	82%	25%	52%	11%
	9	0.35%	0.03%	0.31%	0.31%	65%	9%	52%	64%
	overall	**0.38%**	**0.06%**	**0.29%**	**0.17%**	**73%**	**17%**	**52%**	**37%**

Figure 1. Main effect interactions.



Appendix 1. Illustrative Example

Consider an illustrative example with nine jobs ($n = 9$) and two identical machines ($m = 2$). Table 1 presents for each job j, j=1, …, 9, the processing time p_j, the deterioration effect d_j, and the effect ratio $p_j(1 - d_j)/d_j$. The time t_k to perform a maintenance event on a machine k, k=1, 2, is 5; i.e., $t_1 = t_2 = 5$.

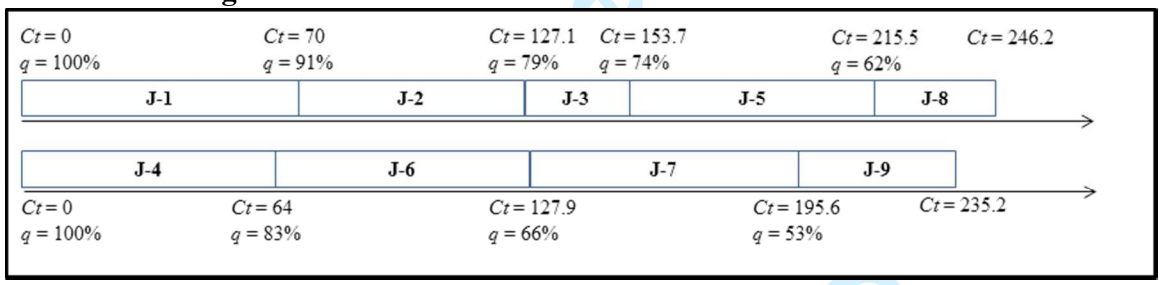
Table A1. Job characteristics

Job j	Processing time $p_{i1} = p_{i2}$	Deterioration effect $d_{i1} = d_{i2}$	Effect ratio $p_{ik}(1 - d_{ik})/d_{ik}$
1	70	9%	707.8
2	52	13%	348
3	21	6%	329
4	64	17%	312.5
5	46	17%	224.6
6	53	20%	212
7	45	20%	180
8	19	11%	153.7
9	21	20%	84

BH1

The first stage of the heuristic with *sorting rule* = non-increasing order of p_j yields the schedule displayed in Figure 1, which has no scheduled maintenance activities, and has a makespan of 246.2.

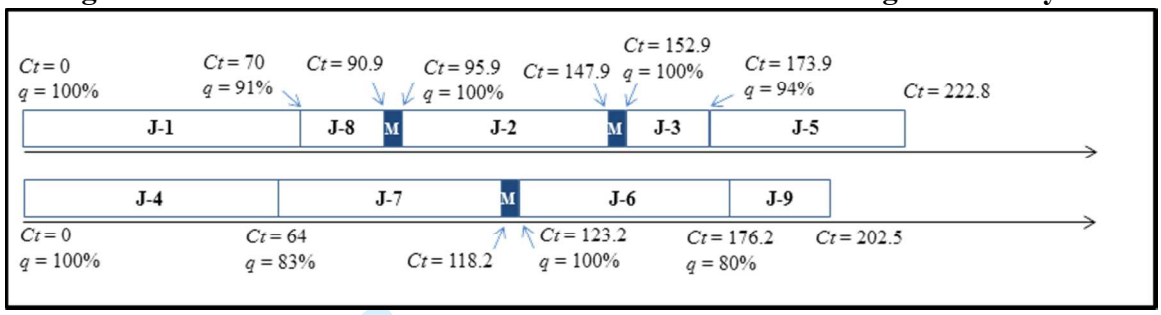
Figure A1. Machine schedules with no maintenance events.



The second stage starts with $k = 1$ and sets $\omega_1 = 1$. Since the current load is 246.2 and the sum of baseline process times is 208, Lemma 2 does not apply (nor do the other two conditions in Step 10). The job to group assigning process (Steps 3-8) results in $G_{1,1}$ = {J2, J5} and $G_{1,2}$ = {J1, J3, J8} with loads of 104.9 and 115.3 respectively. When these groups are assigned to machine k, the completion time of the machine is reduced to 225.2. The process next returns to Step 1, where the makespan machine is now machine 2 (i.e., k = 2) with a makespan of 235.2. As in the previous case, none of the conditions in Step 10 apply. The process in Steps 3-8 results in $G_{2,1}$ = {J4, J7} and $G_{2,2}$ = {J6, J9} with loads of 118.2 and 79.3 respectively. When these groups are scheduled into machine k, the completion time of the machine is now 202.5. The process returns to Step 1, with $k = 1$ and $\omega_1 = 2$. None of the ending conditions is met in Step 2, and Steps 3-8 result in $G_{1,1}$ = {J1, J8}, $G_{1,2}$ = {J2} and $G_{1,3}$ = {J3, J5} with loads of 90.9, 52 and 69.9 respectively. Scheduling these groups and two maintenance events into machine k results in a completion time of

222.8. As the completion time is reduced, the process returns to Step 1, but ends in Step 2 as $C^* - \omega_k t_k - \sum_{j \in N_k} p_j \leq t_k$ is true: 222.8 – 3(5) – 208 ≤ 5. The resulting schedule for *BH1* using as *sorting rules LPT* and non-increasing r_j is presented in Figure 2.

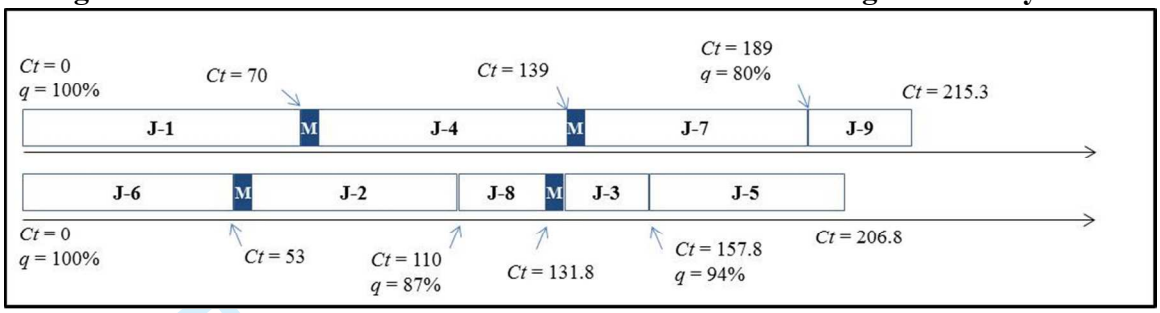
Figure A2. Machine schedule with three maintenance events generated by BH1.



BH2

This heuristic starts with the number of groups equal to the number of machines. Let's consider the case where the *sorting rule* is non-increasing p_j and selecting the group by minimum $C(G)$, the schedule generated by the first iteration is the same as the one shown in Figure 1 with a makespan of 246.2. The next iteration sets $w^* = 3$, and the following resulting group assignments G^*_1 = {J1,J7,J8}, G^*_2 = {J3,J4,J5}, and G^*_3 = {J2,J6,J9} with loads of 145.5, 148, and 143.1, respectively. Loading these three groups into the two machines results in a makespan of 283.6, thus the new schedule is not accepted. The next iteration has $w^* = 4$, and the results in the following group assignments: G^*_1 = {J1,J9}, G^*_2 = {J3,J4,J8}, G^*_3 = {J6,J7}, and G^*_4 = {J2,J5} with loads of 93.1, 113.4, 109.2, and 104.9, respectively. Loading these four groups into the two machines in non-increasing order of their load (considering deterioration) results in a makespan of 219.12 (the sum of groups 3 and 4 and one maintenance event). The next iteration with $w^* = 5$ results does not result in a lower makespan. The next iteration with $w^* = 6$ results in the following group assignments G^*_1 = {J1}, G^*_2 = {J4}, G^*_3 = {J6}, G^*_4 = {J2,J8}, G^*_5 = {J3, J5}, and G^*_6 = {J7,J9} with loads of 70, 64, 53, 73.8, 69.9, and 71.3 respectively. The resulting schedule has groups 3, 4, and 5 assigned to one machine, and the remaining groups to the other machine as shown in Figure 3, for a makespan of 215.3. The next iterations do not generate a schedule with a makespan lower than 215.3.

Figure A3. Machine schedule with four maintenance events generated by BH2.



BH3

Step 1 sorts the jobs in a non-decreasing order of d_j giving L={J3,J1,J8,J2,J4,J5,J6,J7,J9}, and in a non-decreasing order of z_j yielding Z={J1,J4,J6,J2,J5,J7,J3,J9,J8}. Table 2 displays L and Z along with the corresponding d_j, p_j and z_j. Step 2 selects $j1$=3 and $j2$=1. Step 3 constructs a group G'_1={J3,J1} with $d_{G'1}$=14.46%. Because $d_{G'1} < z_8$, where z_8=$\boldsymbol{max}_{j\in Z}\{z_j\}$=20.83%, G'_1 can be augmented with additional jobs. Consequently, Step 3 removes J3 and J1 from both L and Z, inserts G'_1 into L resulting in L={J8,J2,G_1,J4,J5,J6,J7,J9} and Z ={J4,J6,J2,J5,J7,J9,J8}, and goes to Step 6.

Table A2: Ordered lists of d_j and z_j.

L	d_j	p_j		Z	z_j
3	6%	21		1	6.67%
1	9%	70		4	7.25%
8	11%	19		6	8.62%
2	13%	52		2	8.77%
4	17%	64		5	9.80%
5	17%	46		7	10.00%
6	20%	53		3	19.23%
7	20%	45		9	19.23%
9	20%	21		8	20.83%

Because $Z \neq \varnothing$, Step 6 returns to Step 2, which sets $j1$=8 and $j2$=4. As $d_8 > z_4$ and $j1 \neq j2$, Steps 3 and 4 do not apply and Step 5 removes J4 from L and Z; making L={J8,J2,G_1,J5,J6,J7,J9} and Z ={J6,J2,J5,J7,J9,J8}. It schedules J4 on the least loaded machine, which is chosen arbitrarily as machine 1 with C_1=64. Given $Z \neq \varnothing$, the algorithm returns from Step 6 to Step 2, which maintains $j1$=8 and sets $j2$=6. As Steps 3 and 4 do not apply, it reaches Step 5, which removes J6 from L and Z because $d_8 < z_6$; thus, L={J8,J2,G_1,J5,J7,J9} and Z ={J2,J5,J7,J9,J8}. It then schedules J6 on machine 2 with C_2=53. The process returns to Step 2 with $j1$=8 and $j2$=2 and proceeds to Step 5, which removes J2 from L and Z because $d_8 < z_2$ schedules J2 on machine 2 with C_2=110; thus, L={J8,G_1,J5,J7,J9} and Z ={J5,J7,J9,J8}. As $Z \neq \varnothing$ the process continues to Step 6, then to Step 2 with $j1$=8 and $j2$=5. In Step 5, J5 is removed from L and Z because $d_8 < z_5$ and scheduled on machine 1 with C_1=115; thus, L={J8, G_1,J7,J9}, Z ={J7,J9,J8}. The process

iterates back to Step 2, with $j1$=8 and $j2$=7. In Step 5, J7 is removed from L and Z because $d_8 < z_7$, and is scheduled on machine 2 with C_2=160; thus L={J8,G'_1,J9} and Z= {J9,J8}.

The process returns to Step 2, with $j1$=8 and $j2$=9. Because $d_8 < z_9$, Step 3 makes J8 an immediate predecessor of J9, and creates a group G'_2 = {J8,J9} with $d_{G'2}$=28.86%. Consequently, $Z = \varnothing$ and L={G'_1}. Because $Z = \varnothing$, no more jobs can be added to G'_2. Therefore, Step 5 schedules G'_2 on machine 1, whose completion time C_2 becomes 162.60. Because $Z = \varnothing$, Step 6 does not apply. Step 7 schedules G'_1 on machine 2, preceding it by a maintenance event. Thus, C_2=215.47 and $L = \varnothing$. Because $L = \varnothing$, Stage 1 is complete with N_1={J4,J5,J8,J9}, N_2={J6,J2,J7,J3,J1}, and C_{max}=260.47. Jobs J8 and J5 are preceded by maintenance events on machine 1. Similarly, maintenance events precede J2, J7, and J3 on machine 2.

Stage 2 checks if moving the maintenance events can improve the completion time of each machine. Step 8 indicates that for machine 1, J5 and J8 need to be preceded by maintenance event. Similarly, for machine 2, J2, J7 and J3 need to be preceded by maintenance events. Step 9 confirms that no maintenance event should be scheduled between J8 and J9 or between J3 and J1. Step 10 maintains the current completion times of the machines and the makespan as obtained in Stage 1.

Stage 3 rebalances the load between the machines with the objective of reducing the makespan. Step 11 identifies the makespan machine (m_1=2) and the least loaded machine (m_2=1). Step 12 removes J1 from machine 2 resulting in C_2=186. Step 13 positions J1 prior to J9 on machine 1, computes C_1=235.97, and updates C_{max}=235.97. Step 11 is repeated: it sets m_1=1 and m_2=2. Step 12 moves J9 from machine 1 resulting in C_2=208.34. Step 13 positions J1 on machine 1; changing C_1 to 212.89 and C_{max} to 212.89. Further iterations of this stage do not further improve C_{max}. Thus, *BH3* stops with N_1={J4,J5,J8,J1}, N_2={J6,J2,J7,J3,J9}, and C_{max}=212.89. This schedule is displayed in Figure 4.

Figure A4. Machine schedule with five maintenance events generated by BH3.