BI Norwegian Business School

This file was downloaded from BI Open, the institutional repository (open access) at BI Norwegian Business School https://biopen.bi.no

It contains the accepted and peer reviewed manuscript to the article cited below. It may contain minor differences from the journal's pdf version.

Caporin, M., Gupta, R., & Ravazzolo, F. (2021). Contagion between real estate and financial markets: A Bayesian quantile-on-quantile approach. The North American Journal of Economics and Finance, 55, 101347.
https://doi.org/10.1016/j.najef.2020.101347

Copyright policy of Elsevier, the publisher of this journal.
The author retains the right to post the accepted author manuscript on open web sites operated by author or author's institution for scholarly purposes, with an embargo period of 0-36 months after first view online.
http://www.elsevier.com/journal-authors/sharing-your-article#

Minimizing the sum of completion times on a single machine with health index and flexible maintenance operations

Louise Penz[1,4] Stéphane Dauzère-Pérès[1,3] Margaux Nattaf[2*]

[1]Mines Saint-Etienne, Univ Clermont Auvergne
CNRS, UMR 6158 LIMOS
CMP, Department of Manufacturing Sciences and Logistics
Gardanne, France
E-mail: dauzere-peres@emse.fr

[2]Univ. Grenoble Alpes, CNRS, Grenoble INP[†], G-SCOP, 38000 Grenoble, France
E-mail: margaux.nattaf@grenoble-inp.fr

[3]Department of Accounting and Operations Management
BI Norwegian Business School
Oslo, Norway

[4]Normandie Univ, UNIHAVRE, UNIROUEN, INSA Rouen, LITIS
76 600 Le Havre, France
E-mail: louise.penz@univ-lehavre.fr

Abstract

This paper is motivated by the development of Industry 4.0 and the need to better integrate production and maintenance decisions. Our problem considers a single machine on which jobs of different families are scheduled to minimize the sum of completion times. The machine has a health index which decreases when jobs are processed. To restore the machine health, maintenance operations must be scheduled. Moreover, to be scheduled, each job requires the machine to have a minimum health index which depends on the job family. Two cases are studied: (1) The daily case with a single flexible maintenance operation, and (2) The weekly case with two flexible maintenance operations. The second case is shown to be NP-complete. Two Mixed Integer Linear Programming models are presented for each case. The first model uses "classical" positional variables, while the second model improves the first model by using the notion of *master sequence*. Different valid inequalities are also proposed. Computational experiments show that the second model is much more efficient than the first model when solved with a standard solver, and the impact of the valid inequalities is discussed.

Keywords: Single-machine scheduling; Flexible maintenance; Machine health index; Mathematical programming; Master sequence

1. Introduction

Most of the papers in the literature on scheduling consider that a machine is either fully available or not at all. For example, a machine is considered unavailable when a maintenance operation is

*Corresponding author.
[†]Institute of Engineering Univ. Grenoble Alpes

sequenced (deterministic event) or when a breakdown occurs (random event). However, even if the machine is available, it may not be in a perfect state and the process quality may not be guaranteed. With the development of Industry 4.0 concept and the Internet of Things, more and more machines are equipped with sensors to evaluate their status in real time. This is for instance true in the semiconductor industry, where studies show the impact of the health of the machine on the process quality (Chen and Wu (2007)), and degradation models of the machine are developed (Rostami et al. (2021)) that can be used to schedule jobs accordingly. The scheduling problem in this paper is inspired from the problem studied in Kao et al. (2018) where batching machines with health indices are considered and both productivity and quality risk are optimized. However, very differently from Kao et al. (2018), where jobs are only scheduled between maintenance operations and the risk to schedule a job on a machine is considered, we do not allow a job to be scheduled on the machine if the job health index requirement is not met and we schedule maintenance operations to restore the health of the machine. Hence, our objective is to take the degradation of the machine into account when scheduling both jobs and maintenance operations. An important feature of this study is that, because they have different health index requirements, the jobs have different unavailability periods on the machine. These flexible unavailability periods depend on the schedule of the jobs, the health of the machine and the maintenance operations, making the problem particularly complex. We believe this is the first time an integrated production and maintenance scheduling problem where jobs have different health index requirements is considered.

Let us clarify the scheduling problem studied in this paper. A single machine, typically a critical machine in a factory, is considered with a health index. This index decreases when a job is processed on the machine and can be restored through maintenance operations. Jobs are grouped in families which correspond to different types of products. All jobs in the same family have the same processing time, a common and relevant assumption (see for instance Bitar et al. (2016), Nattaf et al. (2019) and Bitar et al. (2021)), and the same health index requirement. A job of family f can start on the machine if the current health index of the machine can remain above the health requirement of f during the entire processing of the job. Otherwise, jobs of f can no longer be sequenced on the machine, and a maintenance operation is required to process jobs of f again. However, jobs of other families might still be scheduled before the maintenance operation if their health index requirement is lower than the one of f. We assume that the maintenance operation leads to an "as good as new" state, restoring the health index of the machine at its maximum level. Maintenance operations have a fixed duration but their start times are flexible and thus can be optimized. The objective is to minimize the flow time, i.e. the sum of completion times. Our assumptions are that the machine is maintained at most two times in a week, and at most one time in any day. The first assumption usually implies the second one, as maintenance operations are usually spread over time, to avoid carrying out a maintenance operation when the machine is in very good condition, and consequently there are not two maintenance operations in a single day. Thus, two cases are considered:

- **The daily case,** where jobs are sequenced on a time horizon smaller than a day. Indeed, if at the start of the day the machine health index is too small, then a maintenance operation must be considered and scheduling the jobs may be difficult. In this case, at most one maintenance operation is sequenced.
- **The weekly case,** where jobs are sequenced on a time horizon of a week. In this case, at most two maintenance operations are sequenced.

This paper considers the notion of master sequence, which is introduced by Dauzère-Pérès and Sevaux (2003, 2004) for the problem of minimizing the number of tardy jobs with release dates on a

single machine. The master sequence is a sequence of jobs from which at least one optimal sequence can be generated. This notion is adapted to our problem, and Mixed Integer Linear Programming models are derived. To evaluate the efficiency of these models, other classical Mixed Integer Linear Programming models using positional variables are developed. Computational experiments compare the performances of the different models, and show that the models using the notion of master sequence are much more efficient.

The paper is organized as follows. In Section 2, the literature on scheduling problems with maintenance operations is reviewed. Our problem is formalized in Section 3 and its complexity is discussed. This section also introduces mathematical models using positional variables. In Section 4, the notion of master sequence is studied. Mathematical models using this notion and valid inequalities are proposed in Section 5. In Section 6, numerical results are presented and discussed. Finally, we conclude and give perspectives in Section 7.

2. Literature review

In the field of scheduling, many studies have been conducted on problems with maintenance operations. Most papers consider maintenance operations as constraints, which can be seen as fixed unavailability periods (Ma et al. (2010); Mati (2010); Mor and Mosheiov (2012); Tamssaouet et al. (2018)). In other papers, maintenance operations are flexible periods with constraints on start times. Indeed, in Chen (2006), Yang et al. (2011) and Qamhan et al. (2020), maintenance operations must be sequenced in time windows. Detti et al. (2019) study a robust single machine scheduling problem with flexible maintenance while minimizing both the makespan and the total completion time. Another constraint found in the literature is that the maintenance operations have to be sequenced before deadlines (Qi et al. (1999); Luo et al. (2015); Ying et al. (2016)).

Some papers consider that maintenance operations have an impact on the processing time of jobs. In this case, maintenance operations are not mandatory. Lee and Leon (2001) and Mosheiov and Sidney (2010) work on the rate modifying activity, which is a maintenance operation that changes the production rate of the equipment under consideration. Thus, the processing times of jobs depend on whether the job is sequenced before or after the maintenance operation. Zhang et al. (2017) and He et al. (2020) work on linear deteriorating jobs, where the job processing time increases as the time interval between the job and the previous maintenance operation increases.

However, the studies discussed above do not take the machine health into account, although the machine health can now be measured. Therefore, there is more and more interest in the study of preventive maintenance operations. The objective is to predict the future machine health based on real data and to use it to schedule maintenance operations in the best possible way. Therefore, the failure rate of the machine is reduced. Among these problems, we distinguish between stochastic and deterministic problems.

Regarding stochastic problems, in Tao et al. (2014), the health index of the machine is determined by a function depending on the time and on the risk of breakdown. Chen and Wu (2007) study the prognosis of the health index under aging Markovian deterioration. Sharifi and Taghipour (2021) consider a single-machine scheduling problem where the machine can be in different multi-deterioration states and different maintenance actions are considered. As in our problem, the authors also consider that the maintenance operations are not already scheduled or time-based. An interesting survey on the integration of production and maintenance scheduling decisions is also proposed in Sharifi and Taghipour (2021). Salama and Srinivas (2021) also study a single-machine scheduling problem with deterioration effect, where the sum of various costs is minimized instead of the makespan. Ghaleb

et al. (2021) consider a flexible job-shop system where production and maintenance schedules that consider the conditions of machines are jointly optimized in real time. Recently, Yang et al. (2022) study the scheduling of jobs on a single machine with uncertain condition, where the processing times of jobs depend on the machine condition and maintenance actions can be performed. Molaee et al. (2021) study a single machine scheduling problem with family setup times and a single random failure to minimize the maximum expected tardiness. However, in these papers, all jobs have the same requirements to be processed on the machine, and the impact of the machine condition does not depend on the type of jobs.

Regarding deterministic problems, some papers consider that the health index of the machine can be deduced from the type of jobs sequenced on the machine. Bock et al. (2012) and Luo et al. (2019) study a problem where jobs have indices of deterioration. The health of the machine decreases according to these indices but all jobs have the same health index requirement. The maintenance duration varies and the longer the maintenance duration, the larger the health index increase. In another study, Kao et al. (2018) consider batching machines that gradually deteriorate after processing jobs. The objective is to balance between maximizing productivity and minimizing quality risk but maintenance operations are not considered.

To our knowledge, no previous study considers our integrated job and maintenance scheduling problem where the availability of the machine depends on the jobs and the maintenance operations, and jobs have different health index requirements.

3. Problem analysis and mathematical modeling

Section 3.1 formally describes the problem and presents the daily and weekly cases. Each case is illustrated by an example in Section 3.2. Complexity results are given in Section 3.3. Finally, mathematical models with positional variables are presented for both cases in Section 3.4.

3.1. Problem description

A set of N jobs $\{J_1, \ldots, J_N\}$ have to be sequenced on a single machine. Jobs are grouped into families in the set $\mathcal{F} = \{1, \ldots, F\}$. For each family $f \in \mathcal{F}$, the number of jobs to scheduled is given by N_f and the processing time of jobs in f is p_f. Note that p_f is the same for all the jobs in family f.

At each moment of the schedule, a health index associated to the machine is defined as follows:

$$\begin{aligned} H : \{0, \ldots, T\} &\longrightarrow \{0, \ldots, H^{max}\} \\ t &\longrightarrow H(t) \end{aligned}$$

where T is the duration of the scheduling horizon, and H^{max} is the maximum health level of the machine.

At the beginning of the schedule, the health index of the machine $H(0)$ is denoted by H^{start}. When a job of family f is sequenced, the health of the machine decreases by p_f, i.e. if a job of family f starts at time t, then $H(t + p_f) = H(t) - p_f$. A maintenance operation of duration p_m can be sequenced to restore the health of the machine. After a maintenance operation, the health index is set to its maximum level H^{max}. As discussed in more details in Section 7, considering the health index of the machine and processing times as stochastic variables instead of deterministic ones is an interesting but challenging research perspective.

Finally, each family f has a health index requirement H_f^{min}. Thus, a job in f cannot start if the health index of the machine is lower than H_f^{min} when the job is completed. Then, a job can

only be started at time t if the current health index of the machine $H(t) \geq H_f^{min} + p_f$. We believe that considering the health index requirement of a job family as deterministic is the most realistic assumption, as H_f^{min} is typically given by process engineers and is thus rarely changing over time. The notations are summarized in Table 1. The goal is to minimize the flow time, i.e. the sum of completion times.

As discussed in the general introduction, two cases are considered:

- **The daily case**. In this case, the time horizon corresponds to one day and only one maintenance operation can be scheduled. To ensure the feasibility of the problem, H^{max} is assumed to be large enough to schedule all the jobs after the maintenance operation. Hence, jobs after the maintenance operation can be sorted in Shortest Processing Time (SPT) (Smith (1956)) order and respect their health index requirements. Thus, the main objective is to determine which jobs are sequenced before the maintenance operation.

- **The weekly case**. In this case, the time horizon corresponds to one week and two maintenance operations can be scheduled. As in the daily case, we assume that the health of the machine after the second maintenance operation is sufficiently large to schedule all the remaining jobs in SPT order.

H^{start}	Health index of the machine at the beginning
H^{max}	Maximum health of the machine
p_m	Maintenance duration
N	Number of jobs
$\mathcal{F}$	Set of families
N_f	Number of jobs of family $f \in \mathcal{F}$
p_f	Processing time of jobs of $f \in \mathcal{F}$
H_f^{min}	Health index requirement of $f \in \mathcal{F}$

Table 1: List of parameters

3.2. Illustrative examples

Figures 1 and 2 show illustrative examples of schedules for the daily and weekly cases. In both cases, Figures 1a and 2a present the family characteristics and Figures 1b and 2b the machine characteristics. Figures 1c and 2c show the optimal scheduled associated with the instance. Furthermore, the numbers above the jobs show the health indices of the machine at the beginning and at the end of each job. The dotted lines linked to the H^{min} boxes represent the health index requirements of the families. Indeed, machine health decreases by the processing time of the scheduled jobs, and schedules without idle time are dominant. Thus, a family health requirement can be converted into a time t such that either jobs are scheduled (and completed) before t or after the maintenance operation.

3.2.1. Daily case

Figure 1 provides an instance and the optimal schedule for the daily case. In this example, since $H^{start} = 97$ and $H_{f_1}^{min} = 70$, jobs of f_1 has to be scheduled either before $t = 97 - 70 = 23$ or after the maintenance operation. In the solution, the first sequenced job belongs to family f_1. As $p_{f_1} = 2$, the health index of the machine decreases from $H^{start} = 97$ to 95. Before the maintenance operation, we notice that the jobs are not in SPT order. Indeed, it is not possible to interchange a job of f_2 with the

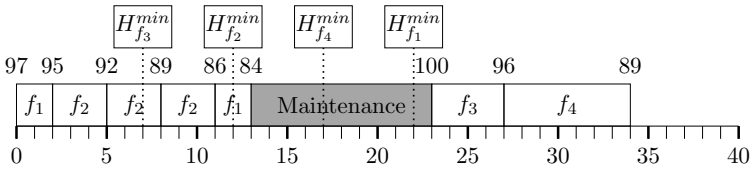
second job of f_1 because of the health index requirement $H^{min}_{f_2} = 85$. If the interchange is performed, the health index of the machine at the end of the job of f_2 will be 84, which is lower than $H^{min}_{f_2}$. After the maintenance operation, the health index requirements are not displayed. $H^{max} = 100$ is large enough to schedule all the jobs in SPT order while satisfying the health index requirements.

	f_1	f_2	f_3	f_4
N_f	2	3	1	1
p_f	2	3	4	7
H^{min}_f	75	85	90	80

(a) Family characteristics

H^{start}	H^{max}	p_m
97	100	10

(b) Machine characteristics



(c) Optimal schedule for the daily case

Figure 1: Example for the daily case

3.2.2. Weekly case

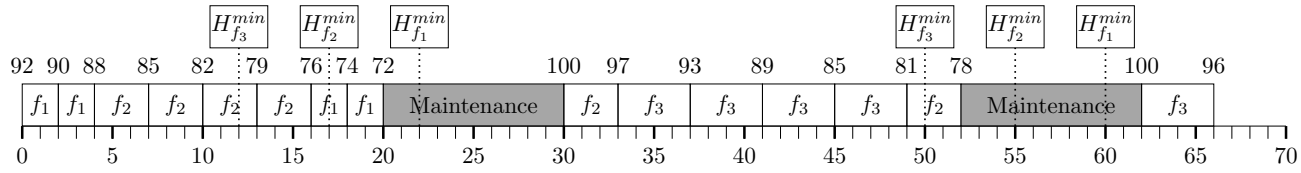
Figure 2 provides an instance and the optimal schedule for the weekly case. The first three H^{min}_f boxes correspond to the health index requirements of the jobs before the first maintenance operation. The next three boxes correspond to the health index requirements for the jobs between the two maintenance operations. After the second maintenance operation is completed, only one job remains, which respects its health index requirement.

	f_1	f_2	f_3
N_f	4	6	5
p_f	2	3	4
H^{min}_f	70	75	80

(a) Family characteristics

H^{start}	H^{max}	p_m
92	100	10

(b) Machine characteristics



(c) Optimal schedule for the weekly case

Figure 2: Example for the weekly case

3.3. Complexity result

This section first describes two polynomial cases for the daily problem. The weekly case is then shown to be NP-complete. Unfortunately, the complexity of the general daily problem remains open.

Theorem 1. If $\sum_{f\in\mathcal{F}} N_f \cdot p_f \leq H^{start} - \max_{f\in\mathcal{F}} H_f^{min}$, the daily scheduling problem with health index requirements can be solved in polynomial time.

Proof. Recall that $[0, H^{start} - H_f^{min}]$ is the time interval in which it is possible to schedule jobs of f before the maintenance operation. Thus $[0, H^{start} - \max_{f\in\mathcal{F}} H_f^{min}]$ is the time interval in which it is possible to schedule jobs of any family before the maintenance operation.

Therefore, if $\sum_{f\in\mathcal{F}} N_f \cdot p_f \leq H^{start} - \max_{f\in\mathcal{F}} H_f^{min}$, no maintenance operation is required. Indeed, all jobs can be scheduled in $[0, H^{start} - \max_{f\in\mathcal{F}} H_f^{min}]$ and scheduling the jobs in SPT order gives the optimal solution. □

Note that if scheduling the jobs in SPT order without maintenance satisfies the health requirement constraints of all jobs, then the SPT order gives the optimal solution. When this is not the case, we define two time intervals: I_1 and I_2. I_1 (resp. I_2) is the time interval in which jobs are scheduled before (resp. after) the maintenance operation. Since, after the maintenance operation, the health index is set to its maximum value H^{max}, which is assumed to be large enough to schedule all the remaining jobs, jobs after the maintenance operation can be sorted in SPT order and respect their health index requirements. Thus, the constraints on I_2 for the health index requirements can be ignored. The main objective is to determine the jobs that need to be sequenced before the maintenance operation.

Another polynomial case for the daily problem is stated in the following theorem.

Theorem 2. If $H_f^{min} = H_k^{min},\ \forall (k, f) \in \mathcal{F}^2$, the daily scheduling problem with health index requirements can be solved in polynomial time.

Proof. We show that the following procedure gives the optimal solution: (1) Sort jobs in SPT order (minimum family indices in case of equality) and schedule the jobs one by one as long as the health requirement is met, and (2) When this is no longer the case and if there are jobs left to schedule, schedule a maintenance operation and schedule the remaining jobs in SPT order.

Note that jobs of the same family are scheduled consecutively, and that at most one family has some of its jobs scheduled before the maintenance operation and the other jobs scheduled after.

Let π be the schedule obtained by following the proposed procedure and $f(\pi)$ the resulting flow time of the schedule. $f(\pi)$ can be divided into two values : $f_p(\pi)$, the processing contribution to the flow time, and $f_m(\pi)$, the contribution of the maintenance operation to the flow time. The first value $f_p(\pi)$ is computed as the sum of the completion times of jobs if no maintenance operation is scheduled. The second value $f_m(\pi)$ is defined as the number of jobs scheduled after the maintenance operation multiplied by the maintenance duration.

Since, in π, jobs are scheduled in SPT order, $f_p(\pi)$ is minimal. The value of $f_m(\pi)$ depends on the number of jobs scheduled after the maintenance operation. Thus, the smaller the number of jobs after the maintenance, the smaller the objective. We can prove that our procedure schedules the maximum number of jobs before the maintenance operation with a classical exchange argument. □

Theorem 3. When at least two maintenance operations can be sequenced, the scheduling problem with health index requirements, i.e. the weekly case problem, is NP-complete.

Proof. The problem is shown to be NP-complete by reducing the *Partition* problem which is NP-complete (Garey et al. (1988)).

Partition. Given a set $\mathcal{A} = \{a_1, \ldots, a_{2n}\}$ of $2n$ positive integers and a positive integer B such that $2B = \sum_{a_i \in \mathcal{A}} a_i$, can $\mathcal{A}$ be partitioned into 2 disjoints subsets $\mathcal{A}_1$ and $\mathcal{A}_2$ such that $\sum_{a_i \in \mathcal{A}_1} a_i = \sum_{a_i \in \mathcal{A}_2} a_i = B$?

For a given instance of *Partition*, we can construct the following scheduling problem with health index requirements P with $2n$ jobs in polynomial time. Let

$$\begin{aligned} p_f = a_f,\ n_f = 1,\ H_f^{min} = 0 \qquad & \forall f \in \{1, \ldots, 2n\} \\ H^{start} = H^{max} &= B \\ p_m &> (n+1) \cdot B \\ C = \frac{n}{2} \cdot p_m + (n+1) \cdot B & \end{aligned}$$

The decision version of the problem P is: Is there a schedule π^* such that $f(\pi) \leq C$?

Lemma 1. If there exists a solution for *Partition*, then there exists a schedule π^* for problem P satisfying $f(\pi^*) \leq C$.

Proof. Suppose $\mathcal{A}_1$ and $\mathcal{A}_2$ are the subsets of the solution to *Partition*. Suppose also, without loss of generality, that $|\mathcal{A}_2| \leq |\mathcal{A}_1|$. The solution of P is constructed by scheduling the jobs corresponding to elements of $\mathcal{A}_1$ (resp. $\mathcal{A}_2$) before (resp. after) the maintenance operation in SPT order.

As $H^{start} = H^{max} = B$ and $\sum_{a_i \in \mathcal{A}_1} a_i = \sum_{a_i \in \mathcal{A}_2} a_i = B$, each job health requirement is satisfied.

Let $p_{[i]}$ be the processing time of the $i-th$ job of the schedule. Thus:

$$\begin{aligned} f_p(\pi^*) &= \sum_{i=1}^{2n} (2n - i + 1) p_{[i]} \\ &= 2n \cdot p_{[1]} + (2n-1) \cdot p_{[2]} + \cdots + 2 \cdot p_{[2n-1]} + p_{[2n]} \\ &\leq |\mathcal{A}_1| \sum_{p_{[i]} \in \mathcal{A}_1} p_{[i]} + \sum_{p_{[i]} \in \mathcal{A}_1} p_{[i]} + |\mathcal{A}_2| \sum_{p_{[i]} \in \mathcal{A}_2} p_{[i]} \\ &= (n+1) \cdot B \end{aligned}$$

And, $f_m(\pi^*) = |\mathcal{A}_2| \cdot p_m$. Thus, $f(\pi^*) = f_p(\pi^*) + f_m(\pi^*) \leq \frac{n}{2} \cdot p_m + (n+1) \cdot B = C$. □

Lemma 2. For problem P, if there exists a schedule π^* satisfying $f(\pi^*) \leq C$, then we have a solution to *Partition*.

Proof. The solution to *Partition* is constructed as follows. Jobs scheduled before (resp. after) the maintenance operation π^* belongs to $\mathcal{A}_1$ (resp. $\mathcal{A}_2$). We have that $\sum_{a_i \in \mathcal{A}_1} a_i = \sum_{a_i \in \mathcal{A}_2} a_i = B$. First, $\sum_{a_i \in \mathcal{A}_1} a_i \leq B$ since health requirement are not satisfied otherwise. Furthermore, $\sum_{a_i \in \mathcal{A}_1} a_i \geq B$, since $\sum_{a_i \in \mathcal{A}_2} a_i > B$. □

The two previous lemmas prove the NP-completeness of P. □

3.4. Mathematical Modeling

This section presents two MILP models using positional variables, i.e. binary variables that define the positions of jobs in the sequence. This choice was motivated by the fact that our problem has similarities with the problem of minimizing the number of late jobs (see Section 4). For the latter problem, the use of positional variables has shown to be effective in Dauzère-Pérès (1995); Dauzère-Pérès and Sevaux (2003, 2004). More recently, Keha et al. (2009) compare different Mixed Integer Programming formulations for single-machine scheduling problems, and show that in models with positional variables, each job is assigned to at most one position, and a position can be occupied by at most one job. The first model, introduced in Section 3.4.1, corresponds to the daily case, and constraints are added in the second model for the weekly case in Section 3.4.2. Several valid inequalities to strengthen the models are proposed in Sections 3.4.3 and 3.4.4.

3.4.1. Daily case

Recall that I_1 (resp. I_2) is the time interval in which jobs are scheduled before (resp. after) the maintenance operation. Since the number of jobs in I_1 and I_2 in an optimal solution is unknown, N positions are considered both in I_1 and in I_2. The following variables are defined to model the daily case:

$u_{k,f}^{I_1} = 1$ iff a job of family f is in position k in interval I_1, i.e. before the maintenance operation, and 0 otherwise,
$C_k^{I_1}$: Completion time of job sequenced at position k in I_1, which is equal to 0 if there is no job at position k,
$u_{k,f}^{I_2} = 1$ iff a job of family f is in position k in interval I_2, i.e. after the maintenance operation, and 0 otherwise,
$C_k^{I_2}$: Completion time of job sequenced at position k in I_2, which is equal to 0 if there is no job at position k.

The Mixed Integer Linear Program **MILP-pos(daily)** is formalized below:

$$\min \sum_{k=1}^{N} (C_k^{I_1} + C_k^{I_2}) \tag{1}$$

$$\sum_{k=1}^{N} (u_{k,f}^{I_1} + u_{k,f}^{I_2}) = N_f \qquad \forall f = 1, \ldots, |\mathcal{F}| \tag{2}$$

$$\sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_1} \leq 1 \qquad \forall k = 1, \ldots, N \tag{3}$$

$$\sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_2} \leq 1 \qquad \forall k = 1, \ldots, N \tag{4}$$

$$C_k^{I_1} \geq \sum_{l=1}^{k} \sum_{f=1}^{|\mathcal{F}|} p_f u_{l,f}^{I_1} + \left(\sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_1} - 1 \right) M^{I_1} \qquad \forall k = 1, \ldots, N \tag{5}$$

$$\sum_{l=1}^{k} \sum_{f=1}^{|\mathcal{F}|} p_f u_{l,f}^{I_1} \leq H^{start} - \sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_1} H_f^{min} \qquad \forall k = 1, \ldots, N \tag{6}$$

$$C_k^{I_2} \geq \sum_{l=1}^{N}\sum_{f=1}^{|\mathcal{F}|} p_f u_{l,f}^{I_1} + p_m + \sum_{l=1}^{k}\sum_{f=1}^{|\mathcal{F}|} p_f u_{l,f}^{I_2} + \left(\sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_2} - 1\right) M^{I_2} \qquad \forall k = 1,\ldots,N \qquad (7)$$

$$C_k^{I_1} \geq 0 \qquad \forall k = 1,\ldots,N \qquad (8)$$

$$C_k^{I_2} \geq 0 \qquad \forall k = 1,\ldots,N \qquad (9)$$

$$u_{k,f}^{I_1} \in \{0,1\} \qquad \forall k = 1,\ldots,N \qquad (10)$$

$$u_{k,f}^{I_2} \in \{0,1\} \qquad \forall k = 1,\ldots,N \qquad (11)$$

where M^{I_1} (resp. M^{I_2}) is an upper bound on the completion time of the last job scheduled before (resp. after) the maintenance operation. The value of M^{I_1} and M^{I_2} are given by:

$$M^{I_1} = H^{start} - \min_{f\in\mathcal{F}} H_f^{min}$$
$$M^{I_2} = M^{I_1} + p_m + H^{max} - \min_{f\in\mathcal{F}} H_f^{min}$$

Objective function (1) minimizes the flow time. Constraints (2) guarantee that exactly N_f jobs of each family f are sequenced. Constraints (3) and (4) guarantee that at most one job is sequenced at each position. Constraints (5) determine the completion time $C_k^{I_1}$ of the job at position k before the maintenance operation if there is a job at position k, i.e. if $\sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_1} = 1$. If $\sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_1} = 0$, M^{I_1} is a positive constant large enough to make the constraint inactive. Constraints (6) ensure that jobs before the maintenance operation are completed before the health index of the machine is too low. If there is no job at position k, the constraints are redundant.

Constraints (7) determine the completion time $C_k^{I_2}$ of the job at position k after the maintenance operation if there is a job at position k, i.e. if $\sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_2} = 1$. If $\sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_2} = 0$, M^{I_2} is a large enough constant to make Constraints (7) redundant. Constraints (8) through (11) define the variable domains.

3.4.2. Weekly case

The model for the weekly case is similar to the model for the daily case but with additional decision variables corresponding to the third time interval I_3, i.e. to the time interval after the second maintenance:

$u_{k,f}^{I_3} = 1$ iff a job of family f is in position k in interval I_3, i.e. after the second maintenance operation, 0 otherwise,
$C_k^{I_3}$: Completion time of job sequenced at position k in I_3, which is equal to 0 if there is no job at position k.

The MILP **MILP-pos(weekly)** is written below:

$$\min \sum_{k=1}^{N} (C_k^{I_1} + C_k^{I_2} + C_k^{I_3}) \qquad (12)$$

$$\sum_{k=1}^{N} (u_{k,f}^{I_1} + u_{k,f}^{I_2} + u_{k,f}^{I_3}) = N_f \qquad \forall f = 1,\ldots,|\mathcal{F}| \qquad (13)$$

Constraints from (3) to (11)

$$\sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_3} \leq 1 \qquad \forall k = 1,\ldots,N \qquad (14)$$

$$\sum_{l=1}^{k}\sum_{f=1}^{|\mathcal{F}|} p_f u_{l,f}^{I_2} \leq H^{max} - \sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_2} H_f^{min} \qquad \forall k = 1,\ldots,N \tag{15}$$

$$C_k^{I_3} \geq \sum_{l=1}^{N}\sum_{f=1}^{|\mathcal{F}|} p_f (u_{l,f}^{I_1} + u_{l,f}^{I_2}) + 2p_m + \sum_{l=1}^{k}\sum_{f=1}^{|\mathcal{F}|} p_f u_{l,f}^{I_3} + \left(\sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_3} - 1\right) M^{I_3} \qquad \forall k = 1,\ldots,N \tag{16}$$

$$C_k^{I_3} \geq 0 \qquad \forall k = 1,\ldots,N \tag{17}$$

$$u_{k,f}^{I_3} \in \{0,1\} \qquad \forall k = 1,\ldots,N \tag{18}$$

where $M^{I_3} = M^{I_2} + p_m + H^{max} - \min_{f\in\mathcal{F}} H_f^{min}$ is an upper bound on the makespan.

Objective function (12) minimizes the flow time. Constraints (13) guarantee that exactly N_f jobs of each family f are sequenced. Constraints (14) guarantee that at most one job is sequenced in each position after the second maintenance operation. Constraints (15) ensure that the job at position k between the two maintenance operations is completed before the health index of the machine is too low. If there is no job at position k, the constraints are redundant. Constraints (16) determine the completion times of jobs after the second maintenance operation. Constraints (17) and (18) define the variables.

3.4.3. Valid inequalities based on dominant positions

A set of valid inequalities based on dominant positions is presented to improve the mathematical models by avoiding symmetries. Indeed, because there are $2N$ positions, and N jobs, there are N empty positions. The same order of jobs can have many different empty positions. The valid inequalities based on dominant positions ensure that only the first positions in I_1, I_2 and I_3 are non empty. This is ensured by Constraints (19) for I_1, Constraints (20) for I_2, and Constraints (21) for I_3.

$$\sum_{f=1}^{|\mathcal{F}|} u_{k-1,f}^{I_1} \geq \sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_1} \qquad \forall k = 2,...,N \tag{19}$$

$$\sum_{f=1}^{|\mathcal{F}|} u_{k-1,f}^{I_2} \geq \sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_2} \qquad \forall k = 2,...,N \tag{20}$$

$$\sum_{f=1}^{|\mathcal{F}|} u_{k-1,f}^{I_3} \geq \sum_{f=1}^{|\mathcal{F}|} u_{k,f}^{I_3} \qquad \forall k = 2,...,N \tag{21}$$

The computational efficiency of these inequalities is discussed in Section 6.1.3.

3.4.4. Valid inequalities based on SPT order

For the daily and weekly cases, jobs can be sequenced in SPT order in the last interval. Valid inequalities can be defined to impose the SPT order. Constraints (22) define the SPT sequence, and are applied on I_2 for the daily case, and on I_3 for the weekly case.

$$u_{k,f_2}^{I} + u_{l,f_1}^{I} \leq 1 \qquad \forall k,l = 1,\ldots,N \quad \forall f_1,f_2 = 1,\ldots,|\mathcal{F}| \text{ such that } k < l,\ p_{f_1} \leq p_{f_2} \tag{22}$$

The computational efficiency of these inequalities is discussed in Section 6.1.3.

4. Notion of Master Sequence

The notion of master sequence is introduced in Dauzère-Pérès and Sevaux (2003, 2004) and is adapted to our problem in this section.

The master sequence is a sequence of jobs from which at least one optimal schedule can be generated. In a master sequence, each job may appear several times and a schedule is "created" from the master sequence by selecting at most one position for each job. The largest possible master sequence is $(J_1, J_2, \ldots, J_N, J_1, J_2, \ldots, J_N, \ldots, J_1, J_2, \ldots, J_N)$, where $(J_1, J_2, \ldots, J_N)$ is repeated N times, and thus has N^2 positions. The objective is to reduce the size of this naive sequence while ensuring that an optimal schedule can always be generated.

Section 4.1 presents theorems on the order between jobs in an optimal sequence. These theorems allow us to compute a master sequence with a smaller size than the naive master sequence, and are used to define an algorithm to build this sequence. Other theorems reducing the size of the master sequence even more are given in Section 4.2.

The results presented in this section show how to compute the master sequence of jobs for time intervals in which the health requirement conditions hold. Indeed, if there are no health requirements, computing the master sequence is easy since it corresponds to the SPT order. The results are presented for interval I_1 (before the first maintenance operation) but holds for interval I_2 of the weekly case (between the two maintenance operations).

4.1. Building the master sequence

Let us first present the following theorem.

Theorem 4. When minimizing $\sum_j C_j$, if two job families f_1 and f_2 are such that $p_{f_1} \leq p_{f_2}$, then there is an optimal sequence where a job of family f_1 is never sequenced between two jobs of family f_2.

Proof. Suppose that, in an optimal solution, an interval contains three jobs $J_1 \in f_1$, $J_{2,1}$ and $J_{2,2} \in f_2$. Let $J_{2,1}$ (resp. $J_{2,2}$) be sequenced before (resp. after) J_1. Exchanging $J_{2,1}$ and J_1 reduces the completion times of the schedule since $p_{f_1} \leq p_{f_2}$. In addition, the health requirements are still satisfied. Indeed, advancing forward job J_1 cannot violate its health requirement and $J_{2,1}$ is moved before $J_{2,2}$ which satisfies its health requirement. Figure 3 illustrates the proof of the theorem. □

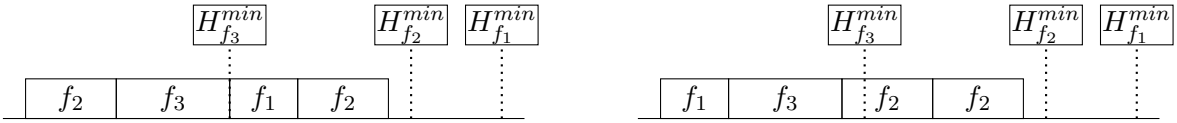


Figure 3: Illustration of the interchange argument in proof of Theorem 4.

Now, let us present the second theorem used to build the master sequence.

Theorem 5. When minimizing $\sum_j C_j$, if two job families f_1 and f_2 are such that $p_{f_1} \leq p_{f_2}$, and $H^{min}_{f_1} \geq H^{min}_{f_2}$, then there exists an optimal sequence where no job of family f_1 is sequenced after a job of family f_2.

Proof. The theorem can again be proven by an interchange argument. If a job of family f_1 is sequenced after a job of family f_2 in an interval, then the two jobs can be interchanged while satisfying their health index requirements (since $H^{min}_{f_1} \geq H^{min}_{f_2}$) and all health index requirements of jobs between them (since $p_{f_1} \leq p_{f_2}$) and reducing $\sum_j C_j$. □

Figure 4 illustrates Theorem 5 and the interchange argument of the proof. The left schedule does not satisfies the conditions of the theorem while the second schedule, where jobs of f_1 and f_2 are swapped, does.

Figure 4: Illustration of the interchange argument in proof of Theorem 5

Theorem 5 shows that, in an optimal solution where $p_{f_1} \leq p_{f_2}$, jobs of family f_1 can only be scheduled after jobs of family f_2 if $H^{min}_{f_1} < H^{min}_{f_2}$. In this case, Theorem 4 shows that there exists an optimal sequence where the jobs of f_1 are scheduled consecutively after the jobs of f_2.

From the above analysis, it is possible to derive a "master sequence of families" which is built recursively using Algorithm 1. In this algorithm, families in $\mathcal{F}$ are ranked in non-decreasing order of their processing times and, for families with the same processing time, in non-increasing order of their health index requirements.

Algorithm 1 Building the master sequence of families

1: **procedure** MASTERSEQUENCE($\mathcal{F}$)
2: **for all** $f \in \mathcal{F}$ **do**
3: $MS \leftarrow MS \cup f$
4: $\overline{\mathcal{F}} \leftarrow \{\overline{f} \in \mathcal{F}$ such that $\overline{f} \neq f$, $p_{\overline{f}} \leq p_f$ and $H^{min}_{\overline{f}} < H^{min}_{f}\}$
5: $MS \leftarrow MS \cup MasterSequence(\overline{\mathcal{F}})$
6: **end for**
7: **end procedure**

Algorithm 1 considers families in non-decreasing order of their processing times and add them in the master sequence (line 3). The corresponding positions in the master sequence are called the **generating positions**. After the addition of a new family in the master sequence, the master sequence generated by families with lower processing times and lower health index requirements is added. These positions in the master sequence are called the **generated positions** (line 5). These positions are generated for the purpose of planning a generated family after its generating family. If no job of the generating family is sequenced, there is no reason to consider the generated positions. A valid inequality using this idea is developed in Section 5.2.2.

Algorithm 1 computes the master sequence of families. To use this sequence in a MILP model, the master sequence of jobs has to be derived from the master sequence of families. To ensure that all possible job orders can be reached, a family f in the master sequence is split into N_f jobs.

Example. Let us apply Algorithm 1 on the instance of Section 3.2.1. First, $\boldsymbol{f_1}$ is added to the master sequence in Line 3. Because, no other families satisfied the condition of Line 4, the algorithm goes back to Line 3 and adds $\boldsymbol{f_2}$ to the master sequence.

The only family satisfying $p_f \leq p_{f_2}$ and $H^{min}_f < H^{min}_{f_2}$ is f_1, thus $\overline{\mathcal{F}} = f_1$ (Line 4). The master sequence generated by $\overline{\mathcal{F}} = f_1$ is f_1 and is added to the master sequence in Line 5. Note that the position of f_1 is generated by f_2.

Then $\boldsymbol{f_3}$ is added in Line 3. We have $\overline{\mathcal{F}} = \{f_1, f_2\}$. The master sequence generated by $\overline{\mathcal{F}} = \{f_1, f_2\}$ is: f_1, f_2, f_1. This sequence is added in Line 5. Finally, $\boldsymbol{f_4}$ is added in Line 3 and $\overline{F} = \{f_1\}$. Thus, f_1 is added to the master sequence in line 5.

The master sequence of families is:

$$\boldsymbol{f_1},\ \boldsymbol{f_2},\ f_1,\ \boldsymbol{f_3},\ f_1,\ f_2,\ f_1,\ \boldsymbol{f_4},\ f_1$$

The families in bold correspond to the generating positions added in Line 3. The master sequence of jobs is obtained by replacing each family by its jobs. For example, each occurrence of family f_1 is replaced by the two jobs of family f_1. The master sequence of jobs is:

$$\boldsymbol{J_{f_1}},\ \boldsymbol{J_{f_1}},\ \boldsymbol{J_{f_2}},\ \boldsymbol{J_{f_2}},\ \boldsymbol{J_{f_2}},\ J_{f_1},\ J_{f_1},\ \boldsymbol{J_{f_3}},\ J_{f_1},\ J_{f_1},\ J_{f_2},\ J_{f_2},\ J_{f_2},\ J_{f_1},\ J_{f_1},\ \boldsymbol{J_{f_4}},\ J_{f_1},\ J_{f_1}$$

4.2. Reducing the length of the master sequence

We show in this section that it is possible to reduce the length of the master sequence of jobs.

In the master sequence, let us consider the generating position corresponding of family f_j. Consider also family f_i associated to a position generated by f_j. Thus, we have $p_{f_i} \leq p_{f_j}$ and $H^{min}_{f_j} > H^{min}_{f_i}$. Note that, in interval $[0, H^{start} - H^{min}_{f_j}]$, it is possible to schedule both jobs of f_i and jobs of f_j. Therefore, since $p_{f_i} \leq p_{f_j}$, jobs of f_i will always be scheduled before jobs of f_j in this interval.

In interval $[H^{start} - H^{min}_{f_j}, H^{start} - H^{min}_{f_j} + H^{min}_{f_j} - H^{min}_{f_i}]$, only jobs of f_i can be scheduled. Thus, $\left\lceil \frac{H^{min}_{f_j} - H^{min}_{f_i}}{p_{f_i}} \right\rceil$ is the maximum number of jobs of f_i that can be completed in this time interval. Hence, the maximum number of jobs of f_i that can be scheduled after the jobs of f_j is equal to $\min\left(N_{f_i}, \left\lceil \frac{H^{min}_{f_j} - H^{min}_{f_i}}{p_{f_i}} \right\rceil\right)$.

Thus, when splitting family positions into job positions, family positions f_i generated by family f_j are split into $\min\left(N_{f_i}, \left\lceil \frac{H^{min}_{f_j} - H^{min}_{f_i}}{p_{f_i}} \right\rceil\right)$ jobs.

Example. Let us use again the example of Section 3.2.1. In the master sequence computed with Algorithm 1, family f_3 in position 4 generates family f_2 in position 6.

In interval $[0, 97 - 90] = [0, 7]$, jobs of both f_2 and f_3 can be scheduled. If a job of each family is scheduled in this interval, the jobs of f_2 will be scheduled before the jobs of f_3 since $p_{f_2} \leq p_{f_3}$.

Suppose that a job of f_3 is scheduled in interval $[0, 7]$. Then, it is not possible to have more than two jobs of f_2 scheduled after interval $[0, 7]$. Indeed, if three jobs are scheduled, then one of them is scheduled in $[0, 7]$ and thus cannot be scheduled after the jobs of f_3. This is because, $\left\lceil \frac{H^{min}_{f_3} - H^{min}_{f_2}}{p_{f_2}} \right\rceil = \left\lceil \frac{90-85}{3} \right\rceil = 2$, and thus only two jobs of f_2 can be scheduled in $[H^{start} - H^{min}_{f_j}, H^{start} - H^{min}_{f_j} + H^{min}_{f_j} - H^{min}_{f_i}] = [7, 7 + 5] = [7, 12]$.

In the master sequence, there are two job positions corresponding to the positions of f_2 generated by f_3, and the master sequence of jobs is:

$$\boldsymbol{J_{f_1}},\ \boldsymbol{J_{f_1}},\ \boldsymbol{J_{f_2}},\ \boldsymbol{J_{f_2}},\ \boldsymbol{J_{f_2}},\ J_{f_1},\ J_{f_1},\ \boldsymbol{J_{f_3}},\ J_{f_1},\ J_{f_1},\ \underline{J_{f_2}},\ \underline{J_{f_2}},\ J_{f_1},\ J_{f_1},\ \boldsymbol{J_{f_4}},\ J_{f_1},\ J_{f_1}$$

5. Mathematical modeling based on Master Sequence

In Section 5.1, two Mixed Integer Linear Programming models for the daily and weekly cases are derived from the master sequence. Four types of valid inequalities are introduced to strengthen the models in Section 5.2.

5.1. Mathematical models

If, in an interval, the families are subject to the health index requirements, the optimal sequence is derived from the master sequence. If the health index requirements are not considered, the optimal sequence is derived from the SPT sequence, i.e the sequence of jobs in Shortest Processing Time order. Note that, since jobs can be scheduled before and after the maintenance operation, both sequences are computed with all jobs. The notations are summarized in Table 2.

L	Number of positions in the master sequence
$x(k)$	Family at position k in the master sequence
N	Number of positions in the SPT sequence
$y(k)$	Family at position k in the SPT sequence

Table 2: Notations

5.1.1. Daily case

In the daily case, jobs in interval I_1 are subject to the health index requirements and thus the master sequence is used. In I_2, because there are no health index requirements, the SPT sequence is used. The following variables are used to model the daily case:

$u_k^{I_1} \in \{0,1\}$: $= 1$ if the job at position k is sequenced in interval I_1, i.e. before the maintenance operation, $= 0$ otherwise,
$C_k^{I_1}$: Completion time of job sequenced at position k in I_1, which is equal to 0 if there is no job at position k,
$u_k^{I_2} \in \{0,1\}$: $= 1$ if the job at position k is sequenced in interval I_2, i.e. after the maintenance operation, $= 0$ otherwise,
$C_k^{I_2}$: Completion time of job sequenced at position k in I_2, which is equal to 0 if there is no job at position k.

The Mixed Integer Linear Program **MILP-MS(daily)** is formalized below:

$$\min \sum_{k=1}^{L} C_k^{I_1} + \sum_{k=1}^{N} C_k^{I_2} \tag{23}$$

$$\sum_{\substack{k=1 \\ x(k)=f}}^{L} u_k^{I_1} + \sum_{\substack{k=1 \\ y(k)=f}}^{N} u_k^{I_2} = N_f \qquad \forall f = 1, \ldots, |\mathcal{F}| \tag{24}$$

$$C_k^{I_1} \geq \sum_{l=1}^{k} p_{x(l)} u_l^{I_1} + (u_k^{I_1} - 1) M^{I_1} \qquad \forall k = 1, \ldots, L \tag{25}$$

$$\sum_{l=1}^{k} p_{x(l)} u_l^{I_1} \leq H^{start} - u_k^{I_1} H_{x(k)}^{min} \qquad \forall k = 1, \ldots, L \tag{26}$$

$$C_k^{I_2} \geq \sum_{l=1}^{L} p_{x(l)} u_l^{I_1} + p_m + \sum_{l=1}^{k} p_{y(l)} u_l^{I_2} + (u_k^{I_2} - 1) M^{I_2} \qquad \forall k = 1, \ldots, N \tag{27}$$

$$C_k^{I_1} \geq 0 \qquad \forall k = 1, \ldots, L \tag{28}$$

$$C_k^{I_2} \geq 0 \qquad \forall k = 1, \ldots, N \tag{29}$$

$$u_k^{I_1} \in \{0,1\} \qquad \forall k = 1, \ldots, L \tag{30}$$

$$u_k^{I_2} \in \{0,1\} \qquad \forall k = 1,\dots,N \qquad (31)$$

with $M^{I_1} = H^{start} - \min_{f\in\mathcal{F}} H_f^{min}$ and $M^{I_2} = M^{I_1} + p_m + H^{max} - \min_{f\in\mathcal{F}} H_f^{min}$.

Objective function (23) minimizes the flow time. Constraints (24) guarantee that exactly N_f jobs of each family f are sequenced. Constraints (25) determine the completion times of jobs before the maintenance operation. Constraints (26) ensure that the health index requirements are respected for jobs sequenced before the maintenance operation. Constraints (27) determine the completion times for jobs sequenced after the maintenance operation. Constraints (28) through (31) define the variables.

5.1.2. Weekly case

The weekly case has three intervals, the first two intervals use the master sequence, thus the number of variables of $u_k^{I_2}$ and $C_k^{I_2}$ is now L, and the last interval uses the SPT sequence. The model of the weekly case has additional decision variables corresponding to the third time interval I_3:

$u_{k,f}^{I_3} \in \{0,1\}$: $= 1$ if the job at position k is sequenced in interval I_3, i.e. after the second maintenance operation, $= 0$ otherwise,
$C_k^{I_3}$: Completion time of job sequenced at position k in I_3, which is equal to 0 if there is no job at position k.

The MILP **MILP-MS(weekly)** is written below:

$$\min \sum_{k=1}^{L}(C_k^{I_1} + C_k^{I_2}) + \sum_{k=1}^{N} C_k^{I_3} \qquad (32)$$

$$\sum_{\substack{k=1\\x(k)=f}}^{L}(u_k^{I_1} + u_k^{I_2}) + \sum_{\substack{k=1\\y(k)=f}}^{N} u_k^{I_3} = N_f \qquad \forall f = 1,\dots,|\mathcal{F}| \qquad (33)$$

Constraints (25) and (26)

$$C_k^{I_2} \geq \sum_{l=1}^{L} p_{x(l)} u_l^{I_1} + p_m + \sum_{l=1}^{k} p_{x(l)} u_l^{I_2} + (u_k^{I_2} - 1)M^{I_2} \qquad \forall k = 1,\dots,L \qquad (34)$$

$$\sum_{l=1}^{k} p_{x(l)} u_l^{I_2} \leq H^{max} - u_k^{I_2} H_{x(k)}^{min} \qquad \forall k = 1,\dots,L \qquad (35)$$

$$C_k^{I_3} \geq \sum_{l=1}^{L} p_{x(l)}(u_l^{I_1} + u_l^{I_2}) + 2p_m + \sum_{l=1}^{k} p_{y(l)} u_l^{I_3} + (u_k^{I_3} - 1)M^{I_3} \qquad \forall k = 1,\dots,N \qquad (36)$$

Constraints (28) and (30)

$$C_k^{I_2} \geq 0 \qquad \forall k = 1,\dots,L \qquad (37)$$

$$C_k^{I_3} \geq 0 \qquad \forall k = 1,\dots,N \qquad (38)$$

$$u_k^{I_2} \in \{0,1\} \qquad \forall k = 1,\dots,L \qquad (39)$$

$$u_k^{I_3} \in \{0,1\} \qquad \forall k = 1,\dots,N \qquad (40)$$

with $M^{I_3} = M^{I_2} + p_m + H^{max} - \min_{f} H_f^{min}$

Objective function (32) minimizes the flow time. Constraints (33) guarantee that exactly N_f jobs of each family f are sequenced. Constraints (34) determine the completion times between the two maintenance operations. Constraints (35) ensure that the health index requirements are respected between the two maintenance operations. Constraints (36) determine the completion times after the second maintenance operation. Constraints (37) through (40) define the variables.

5.2. Valid inequalities

To improve the models based on the master sequence, four types of valid inequalities are developed.

5.2.1. Valid inequalities based on dominant positions

These valid inequalities are based on dominant positions, as for the models with positional variables in Section 3.4.3. For each occurrence of a family, only the first positions should be occupied. Constraints (41) are applied on I_1 and Constraints (42) are applied on I_2 for the daily case. The same constraints are adapted to the weekly case.

$$u^{I_1}_{k-1} \geq u^{I_1}_k \qquad \forall k = 2, ..., L \text{ such that } x(k-1) = x(k) \tag{41}$$

$$u^{I_2}_{k-1} \geq u^{I_2}_k \qquad \forall k = 2, ..., N \text{ such that } y(k-1) = y(k) \tag{42}$$

5.2.2. Valid inequalities based on generating positions

A family f_i is generated by family f_j because, in an optimal sequence, a job of f_i can be after a job of f_j. If there is no job of f_j sequenced, there is no need to consider the generated jobs of f_i. The function $G(k)$ returns all positions of jobs in the last family that generate the job at position k. Constraints (43) are applied on I_1 for the daily and weekly cases, and on I_2 for the weekly case.

$$\sum_{l \in G(k)} u^I_l \geq u^I_k \qquad k = 1, ..., L \qquad \text{such that } G(k) \neq \emptyset \tag{43}$$

5.2.3. Valid inequalities based on Theorem 5

Let two families f_i and f_j be such that $p_{f_i} \leq p_{f_j}$ and $H^{min}_{f_i} \geq H^{min}_{f_j}$. If another family generates family f_i, some jobs of f_j are before jobs of f_i in the master sequence. By Theorem 5, there exists an optimal sequence where there are no jobs of f_i after jobs of f_j. Constraints (44) guarantee that if a job of f_j is sequenced, all jobs of f_i cannot be sequenced after this job. These valid inequalities are applied on intervals using the master sequence.

$$u^I_k + u^I_l \leq 1 \qquad \forall k, l = 1, \ldots, L \qquad \text{such that } l < k, \; p_{x(k)} \leq p_{x(l)} \text{ and } \quad H^{min}_{x(k)} \geq H^{min}_{x(l)} \tag{44}$$

5.2.4. Valid inequalities based on job order around maintenance operations

These valid inequalities order jobs around maintenance operations, and are based on Lemma 3.

Lemma 3. If two families f_1 and f_2 are such that $p_{f_1} \leq p_{f_2}$ and $H^{min}_{f_2} \geq H^{min}_{f_1}$, then there exists an optimal sequence such that there is no job of f_2 in an interval using the master sequence if there is a job of f_1 in a following interval using the SPT sequence.

Proof. If there is a job of f_2 in an interval using the master sequence and a job of f_1 in a following interval using the SPT sequence, it is possible to interchange the two jobs. The completion times of jobs between the two interchanged jobs decrease. Because $H^{min}_{f_2} \geq H^{min}_{f_1}$, the job of f_1 respects its health index requirement. And because the job of f_2 is in the interval using the SPT sequence, it respects its health index requirement by definition. □

For the daily case, Constraints (45) can be applied between I_1 and I_2.

$$u^{I_1}_k + u^{I_2}_l \leq 1 \qquad \forall k = 1, \ldots, L \qquad l = 1, \ldots, N \text{ such that } p_{x(k)} \geq p_{y(l)} \text{ and } H^{min}_{x(k)} \geq H^{min}_{y(l)} \tag{45}$$

For the weekly case, these valid inequalities can be applied between I_1 and I_3, and between I_2 and I_3.

6. Computational experiments

The numerical results for the daily case are presented in Section 6.1 and of the weekly case in Section 6.2. Section 6.1.1, resp. 6.2.1, describes how the instances are generated for the daily case, resp. weekly case. Section 6.1.2 compares MILP-pos(daily) and MILP-MS(daily) without and with their valid inequalities and the reduction of the length of the master sequence. Section 6.1.3 analyzes the impact of the valid inequalities and the reduction of the length of the master sequence on the numerical results of MILP-MS(daily). Section 6.2.2 analyzes the numerical results for the weekly case and compares them with the results obtained for the daily case. Note that more computational experiments are conducted on the daily case than on the weekly case, as the daily case is simpler. The conclusions from the daily case on the impact of the mathematical models, the valid inequalities and the length of the master sequence were also observed for the weekly case.

The Mixed Integer Linear Programming models are implemented in C++ and solved using IBM ILOG CPLEX Optimization Studio 12.10 with 4 threads. The maximum computational time is set to 10 minutes. The programs run on a computer with 4 cores. The processor is an Intel Xeon CPU W3550.

6.1. Daily case

6.1.1. Design of experiments

To compare the different models and the impact of the valid inequalities, different sets of instances have been randomly generated. The larger the number of jobs and the number of families, the harder the instances should be to solve. The number of families ranges from 3 to 15 and the number of jobs from 10 to 500. For each number of families and number of jobs, 100 instances were randomly generated as follows:

- The processing time of a family is randomly generated from a uniform distribution between 1 and 5: $p_f \in [\![1, 5]\!]$.
- The health index requirement of a family is randomly selected in the set $\{80, 70, 60, 50\}$ with respectively a probability in the set $\{20\%, 20\%, 30\%, 30\%\}$, i.e. there is a 20% probability that the health index requirement is 80. Having the largest, i.e. most constraining, health index requirement has the smallest probability.
- H^{start} is randomly generated from a uniform distribution between 50 and 500 which is the maximum number of jobs.
- We ensure that it is always possible to schedule at least one job of each family before the maintenance operation, i.e. $H^{start} - H_f^{min} \geq p_f, \forall f$.
- The maintenance operation takes at least four times the maximum processing time of any job, which is equal to 5, thus the maintenance duration is set to 20.

In addition, to make the instances more difficult to solve, the instances are generated such that (1) The SPT sequence is not the same than the decreasing order of the health index requirements and (2) Two families do not have both the same processing time and the same health index requirement. Finally, if scheduling jobs in SPT order gives a feasible solution, i.e. where all jobs satisfy their health requirements, we discard the instance and generate a new one.

Note that, for the daily case, H^{max} has to be set to a sufficiently large value to ensure that jobs scheduled after the maintenance can be scheduled in SPT order. Thus, we set $H^{max} = \max_{f \in \mathcal{F}} p_f \cdot N_{max} = 5 \times 500 = 2600$.

6.1.2. Comparison of the mathematical models

Let us first compare MILP-pos(daily) and MILP-MS(daily) with and without their valid inequalities and the reduction of the length of the master sequence. The rows "MILP-pos(daily)" and "MILP-MS(daily)" correspond to the results for the MILP models without valid inequalities and the reduction. The rows "MILP-pos-all(daily)" and "MILP-MS-all(daily)" show the results for the MILP models with their valid inequalities and the reduction. As a reminder, MILP-pos(daily) can be used with valid inequalities based on dominant positions and valid inequalities for the SPT sequence in the last interval, and MILP-MS can be used with four types of valid inequalities and the reduced master sequence.

To compare the different formulations arising from the choice of different variables, we look at the number of solutions proven optimal in 10 minutes, written in column "Nb opt". As there are 100 instances, this number is between 0 and 100. The comparison is also done on the mean and the maximum computational times for all instances (columns below "Time (sec)"), and the mean and the maximum optimality gaps provided by IBM ILOG CPLEX (columns below "Gap"). More precisely, column "Mean" below "Gap" corresponds to the sum of the optimality gaps for all instances, even when the optimality gap is equal to 0, divided by the number of instances. The minimum computational time is considered when it is not close to 0.

Table 3 shows the numerical results for different numbers of families and jobs. Note that MILP-MS(daily) strongly dominates MILP-pos(daily). Indeed, for 3 families and 10 jobs, MILP-pos(daily) takes much more time to solve the instances (about 3.8 seconds on average), while all other computational times are lower than 1 second. For 4 families and 15 jobs, MILP-pos(daily) takes 280.7 seconds on average, while MILP-MS(daily) only takes 4.4 seconds on average. Thus the computational time of MILP-MS(daily) is 1.6% of the computational time of MILP-pos(daily). Moreover, MILP-pos(daily) only finds 63 optimal solutions, i.e. there are 37 instances for which an optimal solution was not found or the optimality of the solution was not proved in 10 minutes. MILP-pos(daily) is actually the only MILP that cannot guarantee to find some optimal solutions for 15 jobs. For both MILP-pos(daily) and MILP-MS(daily), valid inequalities and the reduction of the length of the master sequence have a significant impact, since the computational times are very close to 0 for MILP-pos-all(daily) and MILP-MS-all(daily).

As expected, the average computational time increases when the number of jobs increases. For 3 families, when the number of jobs increases from 10 to 15, the computational time of MILP-pos(daily) increases from an average of 3.8 seconds to 271 seconds, and the computational time of MILP-MS(daily) increases from an average of 0.1 seconds to 5.4 seconds. The behaviour is not the same when the number of families increases. The averages computational times of MILP-pos(daily) increase when the number of families increases, while the average computational times of MILP-MS(daily) decreases for 15 jobs.

Table 4 compares the performances of the MILP models with their valid inequalities and the reduction of the length of the master sequence on larger instances. The number of families is set to 5 and the number of jobs is set to 100. Note that there are larger differences between MILP-pos-all(daily) and MILP-MS-all(daily). On average, MILP-MS-all(daily) gives better results than MILP-pos-all(daily). MILP-pos-all(daily) does not prove the optimality of 13 solutions, while MILP-MS-all(daily) proves the optimality of all solutions. The average computational time of MILP-MS-all(daily) is only 2.2 seconds, while the average computational time of MILP-pos-all(daily) is 275.9 seconds, more than 100 times larger. Moreover, the minimum computational time of MILP-pos-all(daily) is 35.4 seconds, while all the minimum computational time of MILP-MS-all(daily) is very close to 0 and its maximum computational time is only 20.3 seconds.

Nb fam	Nb jobs	MILP	Nb opt	Time (sec)		Gap (%)	
				Mean	Max	Mean	Max
3	10	MILP-pos(daily)	100	3.8	105.4	0.0	0.0
		MILP-pos-all(daily)	100	0.1	0.2	0.0	0.0
		MILP-MS(daily)	100	0.1	0.7	0.0	0.0
		MILP-MS-all(daily)	100	0.0	0.0	0.0	0.0
	15	MILP-pos(daily)	60	271.0	600	2.1	10.5
		MILP-pos-all(daily)	100	0.2	0.4	0.0	0.0
		MILP-MS(daily)	100	5.4	112.5	0.0	0.0
		MILP-MS-all(daily)	100	0.0	0.1	0.0	0.0
4	10	MILP-pos(daily)	100	5.1	101.4	0.0	0.0
		MILP-pos-all(daily)	100	0.1	0.2	0.0	0.0
		MILP-MS(daily)	100	0.1	0.6	0.0	0.0
		MILP-MS-all(daily)	100	0.0	0.0	0.0	0.0
	15	MILP-pos(daily)	63	280.7	600	2.4	13.0
		MILP-pos-all(daily)	100	0.3	0.8	0.0	0.0
		MILP-MS(daily)	100	4.4	84.6	0.0	0.0
		MILP-MS-all(daily)	100	0.0	0.2	0.0	0.0

Table 3: Numerical results for models with and without their valid inequalities and the reduction of the length of the master sequence for different numbers of families and jobs.

MILP	Nb opt	Time (sec)			Gap (%)	
		Min	Mean	Max	Mean	Max
MILP-pos-all(daily)	87	35.4	275.9	600	0.2	7.9
MILP-MS-all(daily)	100	0.2	2.2	20.3	0.0	0.0

Table 4: Numerical results for 5 families and 100 jobs.

6.1.3. Analysis of the valid inequalities and the reduction of the length of the master sequence

Since MILP-MS(daily) strongly dominates MILP-pos(daily), the latter is not considered in the remaining numerical results. In this section, the impact of the valid inequalities and the reduction of the length of the master sequence on the computational times is analyzed for MILP-MS(daily). The following notations are used:

Dom corresponds to the valid inequalities based on dominant positions (Section 5.2.1).

Gen corresponds to the valid inequalities based on generating positions (Section 5.2.2).

Uni corresponds to the valid inequalities based on Theorem 5, so based on the unique position of jobs (Section 5.2.3).

PAM corresponds to the valid inequalities based on the positioning of jobs around the maintenance operation (Section 5.2.4).

Red corresponds to the reduction of the length of the master sequence (Section 4.2).

The tables in this section have the following structure: The first four columns indicate the valid inequalities added to the model. The fifth column indicates if the length of the master sequence is

reduced or not. A cross indicates that the valid inequalities are applied on the model or if the master sequence is reduced. The sixth column is the number of instances proven optimal in 10 minutes. The seventh and eighth columns display the computational times (mean and maximum). The ninth and tenth columns display the gap (mean and maximum).

Table 5 shows the numerical results for 4 families and 25 jobs with MILP-MS(daily). For the 100 instances, the results for all combinations of valid inequalities and the reduction of the length of the master sequence are provided. Table 5 shows the most interesting results. The valid inequalities that give the best results are Dom, with an average computational time of 0.3 seconds. The second type of valid inequalities that performs well is PAM, with 170.1 seconds on average, followed by Uni, Red and Gen. When we apply any type of valid inequalities and the reduction of the length of the master sequence, it gives better results than MILP-MS(daily) with nothing. Even though all valid inequalities and the reduction of the length of the master sequence give better results, valid inequalities Dom have a very large impact (0.3 seconds on average) in contrast to the others (over 170 seconds on average). The penultimate row shows the numerical results when all the other valid inequalities and the reduction of the length of the master sequence are added (62.4 seconds on average). Since we test all combinations, we know that these are the best results if we exclude the valid inequalities Dom. All combinations that include Dom give computational times very close to 0.

Valid inequalities				Red	Nb	Time (sec)		Gap (%)	
Dom	Gen	Uni	PAM		opt	Mean	Max	Mean	Max
					57	294.6	600	3.3	21.0
	×				58	285.6	600	3.1	19.2
				×	60	278.9	600	2.6	18.8
		×			61	277.8	600	2.8	18.3
			×		75	170.1	600	1.4	15.6
	×	×	×	×	91	62.4	600	0.4	8.0
×					100	0.3	3.1	0.0	0.0

Table 5: Numerical results for 4 families and 25 jobs solved by MILP-MS(daily), with different combinations of valid inequalities and the reduction of the length of the master sequence.

Adding Dom to the model reduces the computational times considerably. To reach higher computational times, Table 6 shows the numerical results on largest instances, with 5 families and 100 jobs. Numerical results of all combinations of valid inequalities and the reduction of the length of the master sequence including Dom can be found. The most relevant results are presented in Table 6. For the valid inequalities excluding Dom, the order of impact remains the same. PAM has a significant impact, because the average computational time decreases from 132.9 to 51.2 seconds. Then Uni and Gen improve the results. Unlike the results for 4 families and 25 jobs, Red has more impact than the other valid inequalities excluding Dom (47.1 seconds in average). Indeed, this reduction limits the number of jobs in time intervals. Since there are many jobs in large instances, the impact of this limitation is drastic. After testing all combinations, the best combination is when all valid inequalities are added with a reduced master sequence (last row).

Figure 5 shows the numerical results for 15 families and 500 jobs, using MILP-MS(daily) with all valid inequalities and the reduction of the length of the master sequence. 57 solutions on 100 are proven to be optimal and time's data are available on Table 5a. Gap's data for solution not proved optimal are available on Table 5b. For the solutions solved to the optimum, the computational times are spread out over a large range, between 7.2 and 561.3 seconds. It is the same for the instance not solved to the optimum. The gap go from 0.3% to 81.9%. We thus find that, when the number of jobs

Valid inequalities				Red	Nb	Time (sec)		Gap (%)	
Dom	Gen	Uni	PAM		opt	Mean	Max	Mean	Max
×					90	132.9	600	0.6	14.0
×	×				93	94.3	600	0.5	12.6
×		×			95	88.9	600	0.1	3.7
×			×		97	51.2	600	0.0	1.1
×				×	98	47.1	600	0.0	0.6
×	×	×	×	×	100	2.2	20.3	0.0	0.0

Table 6: Numerical results for 5 families and 100 jobs with MILP-MS(daily), with different combinations of valid inequalities and the reduction of the length of the master sequence.

is too large (500 jobs is a large number for most scheduling problems), the computational time might become prohibitive, even for the easiest instances.

Time (sec)		
Min	Mean	Max
7.2	162.2	561.3

(a) Time for the instances solved to the optimum.

Gap (%)		
Min	Mean	Max
0.3	22.2	81.9

(b) Gap for instances not solved to the optimum.

Figure 5: Numerical results for 15 families and 500 jobs for MILP-MS(daily) with all valid inequalities and the reduction of the length of the master sequence.

6.2. Weekly case

6.2.1. Design of experiments

The instances for the weekly case have been generated as follows:

- The values of p_f, H_f^{min} and p_m (maintenance duration) are generated as in the daily case, i.e. $p_f \in [\![1, 5]\!]$, H_f^{min} randomly selected in the set $\{80, 70, 60, 50\}$ with probability $\{20\%, 20\%, 30\%, 30\%\}$ and $p_m = 20$.
- H^{max} is set to 100.
- H^{start} is between 75 and 100.
- As for the daily case, we ensure that it is always possible to schedule at least one job of each family before the maintenance operation, i.e. $H^{start} - H_f^{min} \geq p_f, \forall f$.
- Two families cannot have the same processing time and the same health index requirements.

6.2.2. Analysis of the numerical results

As for the daily case, the computational times of MILP-MS(weekly) are much smaller than the computational times of MILP-pos(weekly), which are also significantly impacted by the valid inequalities and the reduction of the length of the master sequence.

Let us first focus on the models without valid inequalities and without the reduction of the length of the master sequence. Table 7 shows the numerical results for 3 families and 15 jobs for the weekly case, which can be compared to the results for the daily case in Table 3. Since three intervals are considered for the weekly case, there are more variables and constraints. The average computational

times increase for the weekly case (358.3 seconds for MILP-pos(weekly) and 31.8 seconds for MILP-MS(weekly)) compared to the daily case (271 seconds for MILP-pos(daily) and 5.4 seconds for MILP-MS(daily)). In addition, the number of unsolved instances is larger for the weekly case (52 for MILP-pos(weekly) and 1 for MILP-MS(weekly)) than for the daily case (40 for MILP-pos(daily) and 0 for MILP-MS(daily)).

MILP	Nb	Time (sec)		Gap (%)	
	opt	Mean	Max	Mean	Max
MILP-pos(weekly)	48	358.3	600	4.0	16.5
MILP-MS(weekly)	99	31.8	600	0.1	5.5

Table 7: Numerical results for 3 families and 15 jobs for the weekly case.

Table 8 shows additional numerical results obtained on instances with more jobs and families for the daily and weekly cases and the models with all valid inequalities and the reduction of the length of the master sequence. For the weekly case, note that the minimum time, mean time and maximum time to solve the instances significantly increase from MILP-MS-all(daily) to MILP-MS-all(weekly). The average computational time for MILP-pos-all(daily) is 45% of the average computational time for MILP-pos-all(weekly), while the average computational time for MILP-MS-all(daily) is only 2% of the average computational time for MILP-MS-all(weekly). This can be explained by the fact that the master sequence is used in two intervals in the weekly case, and thus in MILP-MS-all(weekly), instead of one interval in the daily case, and thus in MILP-MS-all(daily).

Case	MILP	Nb	Time (sec)		
		opt	Min	Mean	Max
Daily	MILP-pos-all(daily)	100	8.0	65.4	385.1
	MILP-MS-all(daily)	100	0.1	0.9	12.7
Weekly	MILP-pos-all(weekly)	100	50.4	144.8	567.9
	MILP-MS-all(weekly)	100	12.9	45.9	195.6

Table 8: Numerical results for 5 families and 70 jobs for the daily and weekly cases.

7. Conclusions and perspectives

In this paper, a single-machine scheduling problem where the machine has a health index was studied. The objective is to minimize the flow time, i.e. the sum of the completion times of the jobs. The problem complexity comes from the fact that each job has a machine health index requirement, i.e. the job cannot be completed on the machine if the health index of the machine is too low. Two cases were studied: (1) The daily case, where a single maintenance operation is necessary in the schedule to restore the health of the machine and (2) The weekly case, where two maintenance operations might be necessary. The latter case was proved to be NP-complete.

Mixed Integer Linear Programming models relying on positional variables were first proposed with two types of valid inequalities. Then, the notion of master sequence, a sequence of jobs from which at least one optimal sequence can be generated, was introduced. We also showed how the length of the master sequence can be reduced. The master sequence was used to propose new Mixed Integer Linear Programming models and four types of valid inequalities. The numerical results showed that the MILP models based on the master sequence strongly dominate the first MILP models, that reducing

the length of the master sequence is very effective, in particular when there are many jobs, and that one type of valid inequalities significantly decrease the computational times.

The following perspectives are being investigated. The complexity of the daily case still needs to be determined, although our guess is that it is NP-hard. If we want to determine good solutions for larger instances of the problem, heuristics and meta-heuristics could be developed and compared to the exact approaches proposed in this paper. The weighted sum of completion times could also be minimized. The theorems in Section 4 related to the order of jobs in an optimal sequence are no longer valid if jobs have weights. Thus, the algorithm proposed in this paper to build the master sequence must be modified. Other theorems should be developed, as well as a new master sequence. Another interesting topic for future research is the problem on parallel machines, where each machine starts with its own health index. The problem can also be made more complex by considering family setup times, i.e. when changing from one job family to another as for instance in Nattaf et al. (2019), or sequence-dependent and machine-dependent setup times as for instance in Bitar et al. (2021). Also, performing a job on the machine while its health index is low could be allowed, as in Kao et al. (2018), but will induce a risk associated to scrapping the job. A multi-criterion objective function where both the flow time and the total risk are minimized should be considered.

Other longer term perspectives include considering the health index of the machine and the processing time of a job family as stochastic variables. However, if determining the current health index of a machine based on sensor data is feasible, modeling as a stochastic variable how the health index evolves depending on the schedule of jobs is challenging in practice. Although considering processing times as deterministic is still the most common assumption in the scheduling literature but also in practice, stochastic processing times allow to model process uncertainty (see e.g. Flores Gómez et al. (2021)). The main practical challenge is probably to obtain information on the variability of processing times, which remains difficult in many industrial applications.

References

Bitar, A., Dauzère-Pérès, S., Yugma, C., 2021. Unrelated parallel machine scheduling with new criteria: Complexity and models. Computers & Operations Research 132, 1–12.

Bitar, A., Dauzère-Pérès, S., Yugma, C., Roussel, R., 2016. A memetic algorithm to solve an unrelated parallel machine scheduling problem with auxiliary resources in semiconductor manufacturing. Journal of Scheduling 19, 367–376.

Bock, S., Briskorn, D., Horbach, A., 2012. Scheduling flexible maintenance activities subject to job-dependent machine deterioration. Journal of Scheduling 15, 565–578.

Chen, A., Wu, G.S., 2007. Real-time health prognosis and dynamic preventive maintenance policy for equipment under aging Markovian deterioration. International Journal of Production Research 45, 3351–3379.

Chen, J.S., 2006. Single-machine scheduling with flexible and periodic maintenance. Journal of the Operational Research Society 57, 703–710.

Dauzère-Pérès, S., 1995. Minimizing late jobs in the general one machine scheduling problem. European Journal of Operational Research 81, 134–142.

Dauzère-Pérès, S., Sevaux, M., 2003. Using lagrangean relaxation to minimize the weighted number of late jobs on a single machine. Naval Research Logistics 50, 273–288.

Dauzère-Pérès, S., Sevaux, M., 2004. An exact method to minimize the number of tardy jobs in single machine scheduling. Journal of scheduling 7, 405–420.

Detti, P., Nicosia, G., Pacifici, A., Zabalo Manrique de Lara, G., 2019. Robust single machine scheduling with a flexible maintenance activity. Computers & Operations Research 107, 19–31.

Flores Gómez, M., Borodin, V., Dauzère-Pérès, S., 2021. A monte carlo based method to maximize the service level on the makespan in the stochastic flexible job-shop scheduling problem, in: 2021 IEEE 17th International Conference on Automation Science and Engineering (CASE), IEEE. pp. 2072–2077.

Garey, M.R., Tarjan, R.E., Wilfong, G.T., 1988. One-Processor Scheduling With Symmetric Earliness and Tardiness Penalties. Mathematics of Operations Research 13, 330–348.

Ghaleb, M., Taghipour, S., Zolfagharinia, H., 2021. Real-time integrated production-scheduling and maintenance-planning in a flexible job shop with machine deterioration and condition-based maintenance. Journal of Manufacturing Systems 61, 423–449.

He, H., Hu, Y., Liu, W.W., 2020. Scheduling with deterioration effects and maintenance activities under parallel processors. Engineering Optimization 0, 1–18.

Kao, Y.T., Dauzère-Pérès, S., Blue, J., Chang, S.C., 2018. Impact of integrating equipment health in production scheduling for semiconductor fabrication. Computers and Industrial Engineering 120, 450–459.

Keha, A.B., Khowala, K., Fowler, J.W., 2009. Mixed integer programming formulations for single machine scheduling problems. Computers and Industrial Engineering 56, 357–367.

Lee, C.Y., Leon, V.J., 2001. Machine scheduling with a rate-modifying activity. European Journal of Operational Research 128, 119–128.

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

Luo, W., Xu, Y., Tong, W., Lin, G., 2019. Single-machine scheduling with job-dependent machine deterioration. Journal of Scheduling 22, 691–707.

Ma, Y., Chu, C., Zuo, C., 2010. A survey of scheduling with deterministic machine availability constraints. Computers and Industrial Engineering 58, 199–211.

Mati, Y., 2010. Minimizing the makespan in the non-preemptive job-shop scheduling with limited machine availability. Computers and Industrial Engineering 59, 537–543.

Molaee, E., Sadeghian, R., Fattahi, P., 2021. Minimizing maximum tardiness on a single machine with family setup times and machine disruption. Computers & Operations Research 129, 105231.

Mor, B., Mosheiov, G., 2012. Heuristics for scheduling problems with an unavailability constraint and position-dependent processing times. Computers and Industrial Engineering 62, 908–916.

Mosheiov, G., Sidney, J.B., 2010. Scheduling a deteriorating maintenance activity on a single machine. Journal of the Operational Research Society 61, 882–887.

Nattaf, M., Dauzère-Pérès, S., Yugma, C., Wu, C.H., 2019. Parallel machine scheduling with time constraints on machine qualifications. Computers & Operations Research 107, 61–76.

Qamhan, A.A., Ahmed, A., Al-Harkan, I.M., Badwelan, A., Al-Samhan, A.M., Hidri, L., 2020. An exact method and ant colony optimization for single machine scheduling problem with time window periodic maintenance. IEEE Access 8, 44836–44845.

Qi, X., Chen, T., Tu, F., 1999. Scheduling the maintenance on a single machine. Journal of the Operational Research Society 50, 1071–1078.

Rostami, H., Blue, J., Chen, A., Yugma, C., 2021. Equipment deterioration modeling and cause diagnosis in semiconductor manufacturing. International Journal of Intelligent Systems 36, 2618–2638.

Salama, M., Srinivas, S., 2021. Adaptive neighborhood simulated annealing for sustainability-oriented single machine scheduling with deterioration effect. Applied Soft Computing 110, 107632.

Sharifi, M., Taghipour, S., 2021. Optimal production and maintenance scheduling for a degrading multi-failure modes single-machine production environment. Applied Soft Computing 106, 107312.

Smith, W.E., 1956. Various optimizers for single-stage production. Naval Research Logistics Quarterly 3, 59–66.

Tamssaouet, K., Dauzère-Pérès, S., Yugma, C., 2018. Metaheuristics for the job-shop scheduling problem with machine availability constraints. Computers and Industrial Engineering 125, 1–8.

Tao, X., Xia, T., Xi, L., 2014. Opportunistic preventive maintenance scheduling based on theory of constraints, pp. 230–236.

Yang, S.L., Ma, Y., Xu, D.L., Yang, J.B., 2011. Minimizing total completion time on a single machine with a flexible maintenance activity. Computers & Operations Research 38, 755–770.

Yang, W., Chen, L., Dauzère-Pèrés, S., 2022. A dynamic optimisation approach for a single machine scheduling problem with machine conditions and maintenance decisions. International Journal of Production Research 60, 3047–3062.

Ying, K.C., Lu, C.C., Chen, J.C., 2016. Exact algorithms for single-machine scheduling problems with a variable maintenance. Computers and Industrial Engineering 98, 427–433.

Zhang, X., Yin, Y., Wu, C.C., 2017. Scheduling with non-decreasing deterioration jobs and variable maintenance activities on a single machine. Engineering Optimization 49, 84–97.