LI Simpósio Brasileiro de Pesquisa Operacional
Limeira–SP, 02 a 06 de setembro de 2019

A new model and heuristic to a job and maintenance machine scheduling problem with sequence dependent deterioration

Primeiro Autor
Filiação
Endereço da Instituição
e-mail

Segundo Autor
Filiação
Endereço da Instituição
e-mail

RESUMO

Máquinas que possuem alto nível de ocupação comumente apresentam problemas de desgaste que impactam o seu desempenho e fazem com que manutenções corretivas sejam necessárias. Neste trabalho tratamos um problema de escalonamento de tarefas em máquinas paralelas não correlatas com tempo de processamento dependente da deterioração e manutenção. Neste problema buscamos minimizar o *makespan* através da definição de sequências de tarefas e momentos de manutenções de um conjunto de máquinas. O tempo processamento de cada tarefa aumenta por um fator de deterioração q (dependente da tarefa e máquina) a cada tarefa executada desde a última manutenção. Apresentamos neste artigo uma versão linearizada de um modelo matemático disponível na literatura que descreve o problema e uma heurística baseada na meta-heurística ILS para resolvê-lo. Então comparamos os resultados obtidos pela heuristica com os resultados da modelo matemático e de algoritmos da literatura.

PALAVRAS CHAVE. Scheduling. Modelagem matemática. Meta-heuristica.

Tópicos (Escalonamento de tarefas, Modelagem matemática, Linearização, Meta-heurística, ILS)

ABSTRACT

Machines that have a high occupation levels commonly present deterioration problems which impacts on their performance, making needed corrective maintenances. In this paper we deal with a unrelated parallel machine job scheduling with time processing depending of deterioration and maintenance. In this problem, we aim to minimize the makespan through the definition of job sequences and maintenance events to a set of machines. The processing time of each job rises by a deterioration factor q (depending on machine and job) after each job executed after the last maintenance. We present here a linear version of a mathematical model available on the literature to describe the problem as well an ILS based heuristic to solve it. So we compare the results obtained by the proposed heuristic with the results got through the model and literature algorithms.

KEYWORDS. Scheduling. Mathematical modeling. Meta-heuristic.

Paper topics (Job scheduling, Mathematical model, Linearization, Meta-heuristic, ILS)

1. Introduction

Fatigue and deterioration commonly affect the performance of persons or machine executing an activity, causing the increase of processing time, errors or resource waste. In some cases it happens frequently, making necessary to include stop and maintenance events on the scheduling in order to recover the full performance of the executor and provide a overall best functioning of the system.

In this paper we deal with the machine job scheduling problem on environments where deterioration is a relevant factor in a short time horizon, like medical services, student examination and computer processing. In those scenarios, we have a significant increase of processing time after the execution of each job due deterioration and a short maintenance procedure (as a repair, stop, cooling, warming, etc...). In this way, we aim to optimize a scheduling metric (makespan) by choosing which jobs will be processed in each machine, in the process order and also when to execute a maintenance procedures.

Our main contribution in this paper is a linear and improved version of the model proposed by [**?**] to describe the problem for making possible it be solved in MILP solver. We also provide a heuristic procedure that allows to solve the version of problem that considers unrelated machines (with different deterioration rates and different processing and maintenance times).

In the next session we present a short view of the previous works that deal with scheduling with deterioration problem. In the section 3 we describe the problem and next, on section 4, the model proposed by [**?**] and our linear version. In the section 5 we present the proposed heuristic and a adaptation of the heuristic proposed in [**?**] to solve a identical machine version of the problem. In the section 6 we describe the computational experiments and finally on section 7 we present the results of the experiments performed.

2. Literature review

Scheduling with deterioration problems have been investigated since the late 80's, with the first known study presented in [**?**], where the processing time depends on the job starting time. This article was followed in next years by [**?**], [**?**] and [**?**] that used similar ways to evaluate the deterioration.

The published studies can be divided in two groups according to how deterioration is estimated. The first and older group estimate the deterioration based on the job starting time, supposing that a machine is deteriorated with the same rate by any job,without considering the nature of job. The second group estimate the deterioration based on the set of jobs processed previously, considering the eventual different deterioration rates due to the hardness of some jobs.

Among the works in which the deterioration is dependent on starting time we can highlight yet [**?**] that proved that the makespan minimization with parallel machines problem is strongly NP-hard even for a two machine case and [**?**] that created a polynomial-time approximation scheme for the case with a fixed number of machines . In [**?**], [**?**] and [**?**] the concept of step-deterioration is used to describe situations where the processing time just change if the job is processed after a threshold time.

The first paper with a sequence-dependent deterioration was [**?**]. In this paper is presented a non-linear model to describe the problem. This same problem is the subject in [**?**] paper, where is proposed an ILS and a combined ILS with RVND algorithm to solve it.

Maintenance activities are introducted on [**?**] and [**?**], in a parallel machine scheduling with position-dependent deterioration, where the time is not important to define the deterioration but the jobs are yet considered equivalent on deterioration potential. We can cite yet [**?**] that still uses the position-dependent deterioration concept, but without maintenance activities.

The first work where there is a sequence-dependent deterioration and maintenance activities is [**?**]. In this paper, the processing time of a job depend on the sequence of activities performed before it. After this, in the best of our knowledge only [**?**] worked in the same subject, changing slightly the problem definition to make the job deterioration influences its own processing time.

This paper is based on the problem defined by [**?**] and in the next sections we will describe it, present a new mathematical formulation to replace the existent non-linear formulation and propose an ILS based heuristic to solve it.

3. Problem description

The job and maintenance machine scheduling problem with sequence dependent deterioration looks to define where a set of jobs should be scheduled on a set of machines and when to proceed manteinances in those machine in order to optimize some scheduling metric. The processing time of each job on each machine depends of the machine deterioration, that increases after a job be executed and is reset after a maintenance. A maintenance takes a significant amount of time to be done. Then, we can describe the problem as a follows.

Let $J = 1, 2, 3, .., |J|$ be a set of independent and non preemptive jobs to be processed on a set $M = 1, 2, ..., |M|$ of unrelated parallel machines that suffer a significant deterioration after each job processed.

If $p_{jm} \geq 0$ is the baseline processing time of the job $j \in J$ on machine $m \in M$, $\delta_{jm} \geq 1$ is the accumulated delay factor caused by deterioration at machine $m \in M$ just before the start of job $j \in J$ and $d_{jm} \geq 1$ the additional delay factor caused by deterioration on the machine $m \in M$ after the job $j \in J$, we can state that:

- The actual processing time of machine $j \in J$ on machine $m \in M$ is equals to $p_{jm} * \delta_{jm}$
- The accumulated delay factor caused by deterioration on machine $m \in M$ after the processing in it of job $j \in J$ is equals to $\delta_{jm} * d_{jm}$

Considering yet that all jobs are available on time zero, the objective is create a job processing sequence on each machine in order to optimize a scheduling metric, in this article, to minimize the makespan.

In these version of the problem the number of maintenances is unbounded and can be done in any time between the end of a job processing and the beginning of other job processing. Empty intervals on machines are not forbidden but does not make sense. A maintenance in one machine does not impact a maintenance in any other machine and all the machines can be repaired at same time without loss of overall performance.

From this point, we will use the notation exposed on this section to all the rest of the paper.

4. Mathematical model

In this section we present the mathematical model proposed by [**?**] and the modifications done in it to create its linearized version. We also include some constraints to avoid some scenarios not previously predicted in the original model.

The proposed model works based on machine slots that could be filled with a job or a maintenance. Those slots are used to define the position of an activity on a machine schedule and have no fixed duration as it could have in a time indexed formulation.

In this way, we can define if a job $j \in J$ will be executed in the slot $h \in H$ of machine $m \in M$ using the binary variable x_{jmh} and similarly if a maintenance will be executed at slot $h \in H$ of the machine $m \in M$ with the binary variable s_{mh}.

According to the activity performed in a machine slot we can define the resulting performance in the next machine slot. This is stored at the variable q_{mh} that saves the current performance of the machine $m \in M$ in the slot $h \in H$, being its value between 1 to represent a 100% performance (no deterioration) and 0 (total deterioration).

With these definitions we can state the model bellow.

$$\min C_{max} \tag{1}$$

Subject to

$$\sum_{j \in J} x_{jmh} + s_{mh} \leq 1 \qquad \forall m \in M, h \in H \tag{2}$$

$$\sum_{m \in M} \sum_{h \in H} x_{jmh} = 1 \qquad \forall j \in J \tag{3}$$

$$\sum_{j \in J} \sum_{h \in H} \frac{p_{j,m}}{q_{kh}} * x_{jmh} + \sum_{h \in H} t_h * s_{mh} - C_{max} \leq 0 \qquad \forall m \in M \tag{4}$$

$$q_{mh} - \sum_{j \in J} 1 - d_{jm} * q_{m(h-1)} * x_{jm(h-1)} - s_{m(h-1)} = 0 \qquad \forall h \in H \backslash \{1\}, m \in M \tag{5}$$

$$q_{m1} = 1 \qquad \forall m \in M \tag{6}$$

$$x_{jmh} \in \{0, 1\} \qquad \forall j \in J, m \in M, h \in H \tag{7}$$

$$s_{mh} \in \{0, 1\} \qquad \forall m \in M, h \in H \tag{8}$$

$$q_{mh} \geq 0 \qquad \forall m \in M, h \in H \tag{9}$$

In the Equation (**??**) we have the objective function, that is just the value of the variable C_{max} that represents the makespan.The Equation (**??**) states that just one activity can be done by slot. Equation (**??**) assigns an unique job to a machine slot. On Equation (**??**) computes the maskespan based on the finish time on each machine and (**??**) computes the deterioration on each machine slot. Equation (**??**) initializes the q value in each machine and finally Equations (**??**) to (**??**) defines the variables intervals.

As can be seen, the constraints represented in the Equations (**??**) and (**??**) are not linear. We could make them linear easily if there was multiplication instead of a division in the Equation (**??**). However, due the way used to define the machine performance (and consequently the deterioration) there is not a simple way to do this.

The new formulation proposed uses the variable q_{mh} to represent the processing time delay factor (not the performance) of the machine $m \in M$ on slot $h \in H$ after a sequence of activities (jobs and maintenances). Using this approach, the variable lower bound is equals to 1 (representing a non delay on machine) and the upper bound is the product of all deteriorations in the machine.

With this variable meaning change the actual processing time of the job $j \in J$ on machine $m \in M$ in the period $h \in H$ (that we will represent with the variable a_{jmh}) can be calculated as $p_{jm} * q_{mh}$.

Another modification done is in the d_{jm} definition. Now it will represent the delay caused due the deterioration. It can be calculated using the inverse of the value used on the original model without any significant loss of information.

The resulting model is the following:

$$\min C_{max} \tag{10}$$

Subject to

$$\sum_{j \in J} x_{jmh} + s_{mh} \leq 1 \qquad \forall m \in M, h \in H \tag{11}$$

$$\sum_{m \in M} \sum_{h \in H} x_{jmh} = 1 \qquad \forall j \in J \tag{12}$$

$$a_{jmh} \leq M_1 * x_{jmh} \qquad \forall j \in J, m \in M, h \in H \tag{13}$$

$$a_{jmh} \geq p_{jm} * q_{mh} - M_1 * (1 - x_{jmh}) \qquad \forall j \in J, m \in M, h \in H \tag{14}$$

$$\sum_{j \in J} \sum_{h \in H} a_{jmh} + \sum_{h \in H} t_h * s_{mh} - C_{max} \leq 0 \qquad \forall m \in M \tag{15}$$

$$q_{mh} \geq d_{jm} * q_{m(h-1)} - M_2 * [s_{m(h-1)} + (1 - x_{jm(h-1)})] \qquad j \in J, m \in M, h \in H \backslash \{1\} \tag{16}$$

$$q_{m1} = 1 \qquad \forall m \in M \tag{17}$$

$$x_{jmh} \in \{0, 1\} \qquad \forall j \in J, m \in M, h \in H \tag{18}$$

$$s_{mh} \in \{0, 1\} \qquad \forall m \in M, h \in H \tag{19}$$

$$a_{jmh} \geq 0 \qquad \forall j \in J, m \in M, h \in H \tag{20}$$

$$q_{mh} \geq 1 \qquad \forall m \in M, h \in H \tag{21}$$

In this model the Equations (**??**) to (**??**) are the same of original model. Equations (**??**) and (**??**) are the constraints used to define the actual processing time of the job $j \in J$ on slot $h \in H$ of machine $m \in M$ and uses the constant $M_1 \geq \sum_{j \in J} p_{jm}$ to free the value of variable a_{jmh} according to the value of x_{jmh}. Equation (**??**) is the linear version of Equation (**??**) used to calculate the makespan and the Equation (**??**), with the constant $M_2 >= \prod_{j \in J} d_{jm}$, is the linear version of Equation (**??**) constraint used to evaluate the delay in the processing time. Equation (**??**) is correspondent to the Equation (**??**) and Equations (**??**) to (**??**) are constraints to define the bound of the variables.

Beside those constraints we define some others to better describe some situations in the problem, mainly yielding a consistent schedule of the activities. They are presented bellow:

$$s_{k(h-1)} + s_{kh} <= 1 \qquad \forall h \in H\backslash\{1\} \tag{22}$$

$$s_{k(h-1)} + \sum_{j \in J} x_{jm(h-1)} \geq s_{kh} + \sum_{j \in J} x_{jmh} \qquad \forall h \in H\backslash\{1\}, m \in M \tag{23}$$

$$x_{jkh} - \sum_{l \in N} x_l m(h-1) - s_{m(h-1)} \leq 0 \qquad \forall j \in J, m \in M, h \in H\backslash\{0\} \tag{24}$$

The Equation (**??**) states that is not permitted to proceed two maintenances in two consecutive slots. Actually nothing avoids it, but in a optimal solution it does not make senses, because the second maintenance has no effect on deterioration and spends more time (unless the maintenance time is equals to zero, a non-realistic scenario).

The Equations (**??**) and (**??**) are complementary and states that empty slots in the middle of a scheduling are not allowed. In other words, if the previous slot has not activity, it is because the machine use stopped there and its not possible to restart it. As the machine slots have no minimum time duration, it avoids strange solutions with several non used slot between two used slots.

Then, we have in the Equations (**??**) to (**??**) the complete formulation of the model proposed by us and used in the computational experiments of the paper.

4.1. Considerations about the slot based formulation

One of the main concerns about the slot formulation to describe this problem is to define the number of slots available on each machine. Once any job (except the last) in a machine can be followed by a maintenance, a pessimistic value to the number of slots could be $2 * |J|$. However, it is obviously a excessive value, that could be lowered.

The option used by us in this formulation is to check if, to all machines, the sum of the shortest jobs processing times one machine (without considering deterioration) is bigger than the processing time of a single job in any of the other machines. If it is true, we can be sure that the optimal solution has at most $(|J| - |M|)$ on a machine and at most $2(|J| - |M|)$ slots used. As the number of jobs is often much greater than the number of machines, this reduction is not so big, but allows us to get a significant reduction on number of variables and a observable improvement on solving performance.

In other hand, this formulation allow us to decide if the next activity will be a maintenance or not just by looking the current slot state, as in a Markov chain model. It make easy to model some problem characteristics, specially the accumulated deterioration.

5. Heuristic algorithms

In this section we present the heuristic we propose to solve the job and maintenance machine scheduling problem with sequence-dependent deterioration. It is an ILS based algorithm with four local searches and two perturbations, that are used in an alternate way. This algorithm has as stop criteria the number of iterations without improvement (30) and a time limit (30 minutes).

```
   input : J, M, p, d, t
   output: s*
1  nonImproveIters ← 0 ;
2  startTimer (temp)
3  s* ← constructiveHeuristic (J, M, p, d, t);
4  s ← s*
5  while nonImproveIters ≤ MAX_IT  and  temp < MAX_TIME do
6      nonImproveIters ← nonImproveIters + 1 ;
7      s ← internal2SwapLS (s, J, M, p, d, t);
8      s* ← updateBestGlobal (s, s*, nonImproveIters);
9      s ← internal3SwapLS (s, J, M, p, d, t);
10     s* ← updateBestGlobal (s, s*, nonImproveIters);
11     s ← external2SwapLS (s, J, M, p, d, t);
12     s* ← updateBestGlobal (s, s*, nonImproveIters);
13     s ← external3SwapLS (s, J, M, p, d, t);
14     s* ← updateBestGlobal (s, s*, nonImproveIters);
15     if nonImproveIters mod5 = 0 then
16         s ← doPertubation (s)
17     end
18 end
```

Algorithm 1: ILS heuristic

In the Algorithm **??** we can see the ILS function. It receive as input a set of jobs J, a set of machines M, the maintenance time of each machine and the processing time and deterioration delay of each job on each machine.

At line 3 we the variable $s*$ representing the best solution gets the solution obtained in the constructive heuristic. In this paper we tried two constructive algorithms. The first, that we will reference as *longest processing time first*(LPTF), orders all jobs by non-increasing processing time order and after that, starting from the first machine, assigns to each machine the non used job with longest processing time on that machine. The second, referenced as *lowest delay first* (LDF) orders job by increasing deterioration delay order.After that, as in the previous method, we start from the first machine and insert one job by time on each machine, choosing the non used job with lowest delay.

Once the jobs are inserted in the machines, we put the maintenance operations. The strategy used is to evaluate the accumulated delay, starting from the last job to the first, and every time the deterioration delay becomes bigger than maintenance processing time, a maintenance is inserted in the position anterior to the job where the delay surpassed the maintenance processing time.

In the sequence of ILS method, we have from line 7 to 16 a sequence of local searches. After any local search the function *updateBestGlobal* is called to evaluate if the new solution is best than the global solution and set the *nonImproveIters* variable value to 0.

The first local search performed, *internal2SwapLS*, swaps the position, at each machine, of two jobs.Those jobs are chosen as the index of two nested *for* loops. The number of swaps tested is limited by a big value (30000), that eventually is not reached.

The second local search in the sequence(*internal3SwapLS*) changes the position of three jobs in two different ways. Starting from a tuple of three indexes (i_1, i_2, i_3), taken from three nested for loops, it tests two movements (i_3, i_1, i_2) and (i_2, i_3, i_1). The other three possible moves are equivalent to simple two-position swaps, and are not executed. The number of moves tested is limited, as in the previous function

The two following local searches are inter-machines movements. The *external2SwapLS* procedure swap two jobs between two machines (one job of first machine changes the position with one job of second machine). The machines are chosen by the index of two pairs of nested *for* loops (one for each machine) and the jobs are chosen in the same way.

Finally the *external3SwapLS* procedure is analogous with the *external2SwapLS*, changing three jobs of three different machines but adopting the same move strategy of the *internal3SwapLS*.

After all these local searches, if the condition showed on line 17 is satisfied, we proceed a

perturbation of the current solution.

The perturbation procedure used by us calls, alternately two functions. The first one reverses the order of all jobs in one machine and reevaluates the position of the maintenance activities. The second does a shift of one position on machine scheduling, assigning to the machine i the jobs of the machine i+1, and the jobs of the last machine to the first. The order of the jobs in this perturbation does not change and the position of maintenance activities are recalculated.

Due to performance issues, in the local search described above there are no maintenance position swaps, except on the *internal2SwapLS* function, where after each swap the maintenances are reallocated. In the constructive heuristics and perturbation procedures the maintenances are also evaluated.

To confront the results of this heuristic with those of the literature we also adapted the heuristic *Bh2c*, proposed by **?**. This heuristic is a multistart constructive heuristic that, starting from a list of jobs ordered by processing time, tries to insert each job in the end of the group with lower completion time on current solution or in a new group, being the number of groups limited according to the iteration of method. Each group represents, in this case, a sequence of jobs between two maintenances. After that, as the machines are identical, the groups are assigned to the machines starting from the shortest to the longest, in order to minimize the makespan. There is not local search in the heuristic, which makes its very quick.

In our adaptation, the jobs are also sort by processing time, but the assignment of a job to a machine is done before the assignment of a job to a group. We chose the machine with lowest total processing time to receive the new job and after that the group is chosen like in the original method. This small change allow us to have a close to the literature algorithm that works in an unrelated machine scenario and permits to compare our method with the state of art algorithm.

To provide a better understanding of these algorithms and also a possibility of result comparisons in posterior researches we make all the codes available on url ....

5.1. Computational experiments

To test the performance of the heuristics in solving the problem and the fitness of the model to describe the problem we created a instance set based on the parameters described by **?**. To test the model were created three instances to each parameter set combination (number of machines, ration machines by job, deterioration interval and maintenance time interval) summing 144 instances. The heuristics were tested in a large set with 10 instances to each parameter set combination.

The instances created have machines identical processing times to all jobs but different deterioration rates and maintenance times, in a semi-unrelated configuration. This approach was used to make easier to compare the results obtained by our heuristics and the *Bh2c* heuristic proposed by **?** created to solve a identical machines scenario. To make the result more directly comparable the *Bh2c* heuristic was adapted to consider unrelated machines, as described in the previous version. All the algorithms, however, works with instances having machines different processing times.

All the experiments were done in a Ubuntu 18.10 machine with 32 GB of ram and processor Core i7. The algorithms were implemented using Julia language (version 1.1.0) and the model was solved by Cplex 12.6 in default configuration called through the Julia optimization library JuMP with one hour runtime limit. As none of the algorithms uses random walks we run each instance just one time to each algorithm.

6. Results

In this section we make a brief description of the results obtained by each one of methods used to solve the problem. We will start with a description of results obtained in the heuristic procedures and after that compare the results of the best heuristic with those obtained by the model.

As we said before, we create a set of 480 instances, with 10 instances to each combination of parameters. These parameters are number of machines $|M|$, ratio of jobs by machine $|J|/|B|$, deterioration range d and maintenance times range mt. Due to space limitations, in the tables showed in this section we will group the instance results by parameter combination, without individualize the repetitions.

The *LPTF-ILS* heuristic (in other words, the *ILS* based heuristic with *LPTF* constructive algorithm) obtained the best solution in 351 of the 480 instances. It was followed *LDF-ILS* heuristic with 74 best solutions and finally the literature heuristic (*Bh2c*) with 55 best solutions.

As it was stated before, the *Bh2c* is a simple iterative-constructive procedure, without local searches on each. It obviously reflects on the average runtime. While we had an average processing time of $26,11$ seconds and 29.48 seconds in the LPTF and LDF versions of the ILS algorithm respectively, in the *Bh2c* algorithm the average runtime was of only 0.55 seconds

On Table 1 we can the results condensed by instance parameters. We can see in this table that *LPTF-ILS* heuristic has the best solution on almost 80% of cases and how this dominance is spread in all configurations. Another interesting aspect is the fast increase of *Bh2c* heuristic runtime when the number of machines is ≤ 10. When compared with the ILS heuristic runtime we see a 300 times difference fall in smaller instances fall to a 10 times difference on bigger ones, what indicate the existence of a performance degeneration in this algorithm to large instances.

Coincidentally is also in the bigger instances that the *Bh2c* showed its bests results when compared with the other heuristics. But it happened just when the maintenance times were in the interval between 1 and 3. Probably it is due a lack of ability of the ILS algorithm in deal with instances where the ratio delay by maintenance is time is greater and manteinances are more useful to reduce the makespan.

A most "hidden" result found is the advantage in ordering the jobs by non-increasing processing time (longest processing times first). This result, already mentioned on literature to one machine or multi-parallel identical machines problems can be explained intuitively by a simple rule: once if the shorter jobs are done after the potential delays are also lower. Even if this strategy does not bring a optimal solution as in other scheduling with deterioration problems it could provide a good initial and final solution. Its use also appears have no direct impact on convergence speed of the algorithm, as we can see comparing the runtimes of the two versions of the ILS.

In Table 2 we can see the results obtained by the model compared with those obtained by the best heuristic in the instance subset. None of the model solutions is optimal, but the best found after one hour runtime.

We can observe from the Table 2 a prevalence of model best values in small instances, specially with only two machines. As the number of machines increases, the heuristic solutions become better and finally when the number of jobs is greater than 150 (and so the number of slots is greater than 350) the model is not able to find solutions anymore. In most of cases where a solution is not found, the process is aborted on Linux for exceeding the machine memory capacity.

Indeed, strange model objective function values are noted already in instances with 100 jobs and 5 machines, what may indicate the difficulty of Cplex in create branches to explore more the feasible space. In these solutions what commonly happen is a scheduling of several jobs in few machines without any maintenance, making the deterioration grows exponentially and so the makespan.

In any case, by providing feasible solutions in more than 60% of instances tested the model proved to be satisfactory in providing good solutions and a hope of future improvements instead of only a formal description of the problem. Even if optimal solutions were not found we could solve

Table 1: Heuristic results comparison

Instance				LPTF-ILS		LDF-ILS		Bh2C		Best solution
$\|M\|$	$\|J\|/\|M\|$	det. range	maintenance time	values sum	avg. time(s)	value sum	avg. time	value sum	avg. time	
2	10	(1.01 - 1.05)	(1 - 3)	5165.89	3.06	**5146.14**	2.70	5233.28	0.00	5146.14
			(1 - 9)	5091.02	2.91	**5021.02**	3.30	5297.27	0.00	5021.02
		(1.05 - 1.10)	(1 - 3)	5371.03	2.80	**5351.74**	3.26	5472.31	0.00	5351.74
			(1 - 9)	**5673.94**	3.48	5791.06	2.69	6070.00	0.00	5673.94
	15	(1.01 - 1.05)	(1 - 3)	**8037.09**	3.68	8183.19	3.85	8281.55	0.01	8037.09
			(1 - 9)	**8641.63**	3.09	8722.62	3.84	9140.32	0.01	8641.63
		(1.05 - 1.10)	(1 - 3)	8491.96	3.85	**8481.73**	3.96	8635.00	0.01	8481.73
			(1 - 9)	**8317.41**	3.69	8400.27	3.30	9008.67	0.01	8317.41
	20	(1.01 - 1.05)	(1 - 3)	10357.63	4.06	**10261.50**	4.83	10643.25	0.02	10261.50
			(1 - 9)	**10925.55**	4.27	10963.10	5.21	11710.62	0.02	10925.55
		(1.05 - 1.10)	(1 - 3)	**10727.32**	4.22	10870.55	4.85	11004.08	0.02	10727.32
			(1 - 9)	**10657.14**	5.03	10737.06	5.07	11692.73	0.02	10657.14
5	10	(1.01 - 1.05)	(1 - 3)	**5345.53**	14.02	5670.88	14.78	5486.12	0.05	5345.53
			(1 - 9)	**5162.58**	13.28	5618.03	14.67	5461.96	0.05	5162.58
		(1.05 - 1.10)	(1 - 3)	**5358.78**	15.61	5768.07	13.40	5467.86	0.06	5358.78
			(1 - 9)	**5523.56**	14.82	5992.41	17.13	5960.48	0.05	5523.56
	15	(1.01 - 1.05)	(1 - 3)	**7862.03**	20.14	8372.86	22.20	8097.04	0.17	7862.03
			(1 - 9)	8015.33	19.76	**8386.29**	20.23	8620.50	0.16	8015.33
		(1.05 - 1.10)	(1 - 3)	**8183.43**	18.47	8660.87	17.21	8396.36	0.18	8183.43
			(1 - 9)	**8117.55**	22.30	8725.42	16.22	8814.16	0.17	8117.55
	20	(1.01 - 1.05)	(1 - 3)	**10115.80**	30.08	10989.06	29.10	10453.02	0.39	10115.80
			(1 - 9)	**11068.47**	25.51	11887.68	21.09	11886.53	0.38	11068.47
		(1.05 - 1.10)	(1 - 3)	**10750.06**	26.21	11255.20	32.40	11066.53	0.40	10750.06
			(1 - 9)	**10858.24**	26.78	11522.51	27.93	12020.59	0.38	10858.24
10	10	(1.01 - 1.05)	(1 - 3)	5345.01	24.54	6166.87	30.33	**5324.08**	0.57	5324.08
			(1 - 9)	**5545.84**	23.22	6360.34	26.58	5684.28	0.55	5545.84
		(1.05 - 1.10)	(1 - 3)	5548.39	24.02	6533.09	26.34	**5519.30**	0.58	5519.30
			(1 - 9)	**5735.63**	24.30	6511.42	29.57	6035.28	0.56	5735.63
	15	(1.01 - 1.05)	(1 - 3)	**8085.82**	40.31	8955.02	32.87	8162.06	1.87	8085.82
			(1 - 9)	**8000.85**	40.99	8856.95	47.02	8403.02	1.86	8000.85
		(1.05 - 1.10)	(1 - 3)	**8059.88**	41.72	9096.06	41.11	8111.80	1.94	8059.88
			(1 - 9)	**8321.30**	38.70	9214.92	39.50	8954.31	1.90	8321.30
	20	(1.01 - 1.05)	(1 - 3)	**10823.95**	67.19	12169.92	68.93	11090.56	4.54	10823.95
			(1 - 9)	**10838.02**	65.02	11843.52	97.55	11475.90	4.58	10838.02
		(1.05 - 1.10)	(1 - 3)	**10572.30**	65.78	11805.80	92.79	10749.27	4.71	10572.30
			(1 - 9)	**11234.70**	81.99	12517.12	60.17	12289.36	4.51	11234.70
20	10	(1.01 - 1.05)	(1 - 3)	5606.80	68.82	6624.71	104.21	**5496.81**	7.00	5496.81
			(1 - 9)	**5808.86**	70.04	6674.04	70.99	5860.39	7.00	5808.86
		(1.05 - 1.10)	(1 - 3)	5610.61	76.43	6778.45	69.75	**5427.72**	7.23	5427.72
			(1 - 9)	**5672.84**	75.23	6786.92	76.32	5914.92	7.10	5672.84
	15	(1.01 - 1.05)	(1 - 3)	**8266.51**	188.08	9574.26	199.17	8277.80	24.43	8266.51
			(1 - 9)	**8315.54**	193.38	9527.92	165.24	8575.00	24.56	8315.54
		(1.05 - 1.10)	(1 - 3)	8336.71	156.26	9781.78	172.81	**8280.65**	24.94	8280.65
			(1 - 9)	**8435.96**	168.96	9717.73	139.54	8946.37	24.37	8435.96
	20	(1.01 - 1.05)	(1 - 3)	**10684.36**	414.02	12366.67	286.62	10821.52	59.29	10684.36
			(1 - 9)	**11138.94**	391.27	12593.99	449.63	11671.24	58.49	11138.94
		(1.05 - 1.10)	(1 - 3)	**11104.89**	468.26	12706.41	455.65	11151.75	60.56	11104.89
			(1 - 9)	**11230.21**	428.91	13026.22	317.59	12054.92	60.24	11230.21

hard scenarios with two machines better than with the heuristic and close to it in other situations. In our next investigations, we will try to explore theoretical results already know to create a better model and try to get the first optimal solutions.

7. Conclusion

In this paper we presented a new model to the machine job and maintenance scheduling problem with sequence dependent deterioration, as well a heuristic algorithm to solve it.

The model proposed was able to get non-optimal solutions to more than 60% of the instances tested and have the best solution found in cases with a low number of machines.

Our ILS based heuristic was able to reach good results when compared with a adapted version of state of art algorithm, mainly when the maintenance processing times were higher.

As we believe on the theoretical potential of solving well this problem we will continue to investigate it. In the next steps we will try to improve the performance of the heuristic presented by choosing better the maintenance positions, using more theoretical results available on the literature and trying more space efficient solution representations. In other hand, we will look for the development of alternative models that could enable to use robust and exact optimization techniques like column generation.

Table 2: Model and best heuristic results comparison

Instance				Model	Best heuristic
$\|M\|$	$\|J\|/\|M\|$	det. range	maintenance time		
2	10	(1.00 - 1.05)	(1 - 3)	**1508.93**	1544.40
			(1 - 9)	**1488.68**	1529.95
		(1.05 - 1.10)	(1 - 3)	**1282.40**	1325.69
			(1 - 9)	**1544.54**	1628.10
	15	(1.00 - 1.05)	(1 - 3)	**2541.33**	2610.55
		(1.00 - 1.05)	(1 - 9)	**2445.65**	2557.38
		(1.05 - 1.10)	(1 - 3)	**2620.72**	2688.34
			(1 - 9)	**2324.74**	2416.93
	20	(1.00 - 1.05)	(1 - 3)	**3063.25**	3123.56
			(1 - 9)	**2170.06**	2280.81
		(1.05 - 1.10)	(1 - 3)	**2041.28**	2123.05
			(1 - 9)	**1776.17**	1841.48
5	10	(1.00 - 1.05)	(1 - 3)	**1094.66**	1088.06
			(1 - 9)	**985.36**	1019.08
		(1.05 - 1.10)	(1 - 3)	1134.36	**1104.51**
			(1 - 9)	4111.84	**1074.09**
	15	(1.00 - 1.05)	(1 - 3)	4135.38	**1493.68**
			(1 - 9)	2170.51	**1574.19**
		(1.05 - 1.10)	(1 - 3)	4475.05	**1658.04**
			(1 - 9)	2049.83	**1526.15**
	20	(1.00 - 1.05)	(1 - 3)	47692.86	**1959.60**
			(1 - 9)	3093.18	**2231.79**
		(1.05 - 1.10)	(1 - 3)	5672.07	**2072.73**
			(1 - 9)	5740.00	**2390.12**
10	10	(1.00 - 1.05)	(1 - 3)	539462.00	**1094.23**
			(1 - 9)	46701.00	**1091.42**
		(1.05 - 1.10)	(1 - 3)	22649.00	**1089.04**
			(1 - 9)	819636.00	**1116.17**
	15	(1.00 - 1.05)	(1 - 3)	500363.00	**1608.41**
			(1 - 9)	528772.00	**1621.94**
		(1.05 - 1.10)	(1 - 3)	***	1623.52
			(1 - 9)	***	1678.03
	20	(1.00 - 1.05)	(1 - 3)	***	2309.79
			(1 - 9)	***	2083.85
		(1.05 - 1.10)	(1 - 3)	***	2054.90
			(1 - 9)	***	2337.43
20	10	(1.00 - 1.05)	(1 - 3)	***	1108.75
			(1 - 9)	***	1111.32
		(1.05 - 1.10)	(1 - 3)	***	1045.14
			(1 - 9)	***	1110.78
	15	(1.00 - 1.05)	(1 - 3)	***	1713.94
			(1 - 9)	***	1666.68
		(1.05 - 1.10)	(1 - 3)	***	1671.27
			(1 - 9)	***	1653.88
	20	(1.00 - 1.05)	(1 - 3)	***	2109.22
			(1 - 9)	***	2257.76
		(1.05 - 1.10)	(1 - 3)	***	2288.62
			(1 - 9)	***	2232.33

References

Browne, S. e Yechiali, U. (1990). Scheduling deteriorating jobs on a single processor. *Operations Research*, 38(3):495–498.

Cheng, T. e Ding, Q. (2001). Single machine scheduling with step-deteriorating processing times. *European Journal of Operational Research*, 134(3):623–630.

Ding, J., Shen, L., Lü, Z., e Peng, B. (2019). Parallel machine scheduling with completion-time-based criteria and sequence-dependent deterioration. *Computers & Operations Research*, 103:35 – 45. ISSN 0305-0548.

Gupta, J. N. e Gupta, S. K. (1988). Single facility scheduling with nonlinear processing times. *Computers & Industrial Engineering*, 14(4):387–393.

Huang, X. e Wang, J.-J. (2015). Machine scheduling problems with a position-dependent deterioration. *Applied Mathematical Modelling*, 39(10):2897 – 2908. ISSN 0307-904X.

Ji, M. e Cheng, T. (2008). Parallel-machine scheduling with simple linear deterioration to minimize total completion time. *European Journal of Operational Research*, 188(2):342 – 347. ISSN 0377-2217.

Lalla-Ruiz, E. e Voß, S. (2016). Modeling the parallel machine scheduling problem with step deteriorating jobs. *European Journal of Operational Research*, 255(1):21–33.

Leung, J. Y.-T., Ng, C., e Cheng, T. E. (2008). Minimizing sum of completion times for batch scheduling of jobs with deteriorating processing times. *European Journal of Operational Research*, 187(3):1090–1099.

Mosheiov, G. (1991). V-shaped policies for scheduling deteriorating jobs. *Operations Research*, 39 (6):979–991.

Mosheiov, G. (1994). Scheduling jobs under simple linear deterioration. *Computers & Operations Research*, 21(6):653–659.

Mosheiov, G. (1998). Multi-machine scheduling with linear deterioration. *INFOR: Information Systems and Operational Research*, 36(4):205–214.

Ruiz-Torres, A. J., Paletta, G., e M'Hallah, R. (2017). Makespan minimisation with sequence-dependent machine deterioration and maintenance events. *International Journal of Production Research*, 55(2):462–479.

Ruiz-Torres, A. J., Paletta, G., e Pérez, E. (2013). Parallel machine scheduling to minimize the makespan with sequence dependent deteriorating effects. *Computers & Operations Research*, 40 (8):2051 – 2061. ISSN 0305-0548.

Santos, V. L. A. e Arroyo, J. E. C. (2015). Sequenciamento de tarefas em maquinas paralelas considerando desgastes dependentes da sequencia. In *XLVII SBPO, SÃmposio Brasileiro de Pesquisa Operacional*, p. 2572 – 2583.

Yang, D.-L., Cheng, T., Yang, S.-J., e Hsu, C.-J. (2012). Unrelated parallel-machine scheduling with aging effects and multi-maintenance activities. *Computers & Operations Research*, 39(7): 1458 – 1464. ISSN 0305-0548.

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.