International Journal of Industrial Engineering Computations 7 (2016) 283–294

Contents lists available at GrowingScience

International Journal of Industrial Engineering Computations

homepage: www.GrowingScience.com/ijiec

Minimizing the total tardiness for the tool change scheduling problem on parallel machines

Antonio Costa[a], Fulvio Cappadonna[b] and Sergio Fichera[a*]

[a]*Department of Industrial and Mechanical Engineering, University of Catania, Catania, viale A. Doria 6, 95126, Italy*
[b]*Department. of Electrical Electronic Information Engineering, University of Catania, Catania, viale A. Doria 6, 95126, Italy*

C H R O N I C L E

Article history:
Received July 22 2015
Received in Revised Format August 17 2015
Accepted October 14 2015
Available online October 14 2015

Keywords:
Scheduling
Parallel machines
Tool change
Total tardiness minimization
Linear programming

A B S T R A C T

This paper deals with the total tardiness minimization problem in a parallel machines manufacturing environment where tool change operations have to be scheduled along with jobs. The mentioned issue belongs to the family of scheduling problems under deterministic machine availability restrictions. A new model that considers the effects of the tool wear on the quality characteristics of the worked product is proposed. Since no mathematical programming-based approach has been developed by literature so far, two distinct mixed integer linear programming models, able to schedule jobs as well as tool change activities along the provided production horizon, have been devised. The former is an adaptation of a well-known model presented by the relevant literature for the single machine scheduling problem with tool changes. The latter has been specifically developed for the issue at hand. After a theoretical analysis aimed at revealing the differences between the proposed mathematical models in terms of computational complexity, an extensive experimental campaign has been fulfilled to assess performances of the proposed methods under the CPU time viewpoint. Obtained results have been statistically analyzed through a properly arranged ANOVA analysis.

1. Introduction

Machine availability restrictions are a crucial issue to be taken into account whenever real scheduling problems arising from industrial practices have to be tackled. According to Pinedo (2012), there are many reasons why machines may temporarily be inoperative during production sessions. Some of these reasons depend on random circumstances, e.g. breakdowns and repairing operations; others, like preventive maintenance or tool changes, are based on deterministic processes. The research area connected with this latter case received many contributions during the last decades and the common thought was that, in real-world manufacturing environments, maintenance activities should be planned along with production operations. Three comprehensive reviews covering this topic have been provided by Lee et al. (1997), Schmidt (2000) and Ma et al. (2010), respectively.

* Corresponding author. Tel: 00390957382411 Fax: 0039095337994
E-mail: sfichera@diim.unict.it (S. Fichera)

doi: 10.5267/j.ijiec.2015.10.002

In this paper, we consider the scheduling problem in a parallel machine environment with tool change unavailability. Scheduling jobs under tool change considerations is quite common in many industrial environments such as Computer Numerically Controlled (CNC) work centers, Integrated Circuit (IC) testing operations and Printed Circuit Board (PCB) production (Low et al., 2010). Tool replacement operations in industrial applications may be motivated by part mix production (Tang & Denardo 1988; Hertz et al. 1998), but the tool change issue induced by wear usually has a greater impact on production performances, as confirmed by Gray et al. (1993) who stated that, in real-world applications, tool changes due to wear are approximately 10 times more frequent than those induced by job mix.

The tool change scheduling problem has mainly concerned the single machine: Akturk et al. (2004) addressed the problem of minimizing the total completion time on a single machine subject to tool wear, providing theoretical worst-case performance bounds for the Shortest Processing Time (SPT) heuristic. Akturk et al. (2007) investigated the total completion time minimization objective on a single CNC machine subject to tool wear and controllable machining conditions. The same authors proposed a Mixed Integer Linear Programming (MILP) formulation capable of solving up to 30-job problems, and a Genetic Algorithm (GA) to be employed for larger-sized instances. Chen (2008) evaluated the efficiency of two alternative MILP models for solving the total tardiness minimization problem on a single CNC machine subject to tool changes. Xu et al. (2013) considered a variation of the classical single machine scheduling problem with tool changes, wherein a set of special jobs has to be processed within the first prefixed time units of a tool life. In order to minimize the makespan, they proposed two MILP formulations and six heuristic algorithms to be employed for small and large-sized instances, respectively.

As far as parallel machine scheduling is concerned, literature extensively analyzed the effects of deterministic availability restrictions. Lee (1991) proposed a modified Longest Processing Time (LPT)-based algorithm for minimizing makespan in a production environment made by m identical machines not simultaneously available at time zero. Five years later, the same author (1996) discussed the performance of List Scheduling (LS) and LPT heuristics for the identical parallel machine problem wherein one machine is always available and the other ones are subject to a single unavailability interval during the planning time. The effectiveness of LPT heuristic was also investigated by Hwang et al. (1998) and Hwang, et al. (2005) for the parallel machine scheduling problem where some workstations are planned to be shut down during a-priori known time interval. Liao et al. (2005) provided an exact algorithm for a two-machine problem with one machine needing preventive maintenance or periodical repair during a fixed period. Lin and Liao (2007) dealt with the more general case where both machines have a single unavailability interval; they proposed an exact procedure based on a lexicographic search able to solve instances with up to 1,000 jobs. Xu and Li (2008) developed a polynomial-time approximation algorithm for the parallel machine scheduling problem with almost periodic maintenance activities, i.e., where the difference between the time of two consecutive maintenance activities is always within a-priori known value. Wang and Wei (2011) investigated the computational complexity of the parallel machine scheduling problem with deteriorating maintenance activities. In this case, the planned maintenance time increases when a delay of its starting time occurs.

The scheduling problem in a parallel machine environment with tool change unavailability has been proposed by Yang et al. (2012). They provided an efficient algorithm for minimizing the total machine load in a system made by m unrelated workstations needing continuous maintenance to counteract aging effects. In this problem, the actual processing time of a product on a machine increases with respect to the number of products already processed on the machine, due to wearing of the cutting tool. The quality of the cutting tool influences the time required for processing a product but not the quality characteristics of the same product. Therefore, the cutting tool is replaced by a new one or is maintained after it has processed some products to improve its production efficiency. Zarook and Abedi (2014) considered the same scheduling problem. Their objective is to jointly find the optimal maintenance frequencies, the optimal positions of the maintenance activities and the optimal job sequences on the machines, such that the total early/tardy and maintenance cost are minimized.

In this paper, another and more realistic condition is considered as proposed by Chen (2008): when the cutting tool exceeds a fixed employment time corresponding to its tool life, it is replaced because it can damage the product. On the contrary, in the model proposed by Yang et al. (2012) the effects of tool wear on the product are negligible and it is possible to never change the tool on the machine.

The present paper tackles the problem of scheduling *n* jobs on a manufacturing system made by *m* identical parallel machines subject to periodical tool changes due to wear. In order to minimize the total tardiness, two distinct MILP formulations are proposed. The remainder of the paper is organized as follows. Section 2 reports the problem statement. Section 3 illustrates the MILP models developed In Section 4, the alternative mathematical programming formulations are compared under the size complexity and the computational performance viewpoints. Finally, Section 5 concludes the paper.

2. Problem description

The proposed Parallel Machines Tool Change Scheduling (PMTCS) problem can be stated as follows. Let us consider a set of *n* jobs to be worked on a manufacturing system made by *m* identical parallel machines. Each job has to be processed by only one machine before the exit from the system is allowed, and each machine cannot process more than one job at a time. Each machine makes use of a tool to process jobs. Due to wear effects, tools have a limited life, hereinafter referred to as *TL*. When a tool reaches the end of its life, it must be replaced. Every tool change operation generates a machine unavailability for a time interval equal to *TC*. Pre-emption is not allowed, i.e. job processing cannot be interrupted once it has started. Therefore, if the residual life of a tool is not enough to complete a job on a given machine, such a tool must be replaced with a new one before the job can be processed. The objective function to be minimized is the total tardiness *T*, and job descriptors like processing times and due dates are assumed to be a-priori known. All jobs have release time equal to zero, i.e., they are ready to be worked at the beginning of the planning period.

With regards to the computational complexity of the investigated issue, it is worth noting that Biskup et al. (2008) put in evidence that the total tardiness minimization problem on identical parallel machines is $\mathcal{NP}$-hard. A PMTCS problem can be easily relaxed to a classical parallel machine scheduling issue when the life of a tool tends to infinity. As a result, the scheduling problem here investigated is $\mathcal{NP}$-hard, as well. Since the seminal work of Kan (1976), the scheduling literature has traditionally employed mathematical programming models to formally describe objective and constraints of problems to be solved. Although most of scheduling problems are classified as $\mathcal{NP}$-hard, the computational performances of modern calculators, along with the new developments under the optimization engines viewpoint have made such techniques an effective way to optimally solve both small- and medium-sized issues. Recently, Edis et al. (2013) highlighted the effectiveness of MILP-based approaches for solving unconventional parallel machines scheduling problem with additional resources.
The mathematical models proposed for the problem at hand are based on the concept of “slot”. A slot is a position, ranging from 1 to *n*, to which a job may be assigned for processing by a certain machine. Since the problem entails *m* parallel machines, the total number of provided slots is equal to $n \cdot m$, though only *n* slots on the whole will actually be occupied by jobs. The following notation is used for the two proposed models, hereinafter named PMTCS-1 and PMTCS-2, respectively:

$j = 1,2,\ldots,n$	index for jobs;
$l, h = 1,2,\ldots,n$	indexes for slots on each machine;
$i = 1,2,\ldots,m$	index for machines;
p_j	processing time of job *j;*
d_j	due date of job *j;*

TL	tool life;
TC	tool change time;
M	a large positive number;

2.1. The PMTCS-1 mathematical programming model

This model consists of an adaptation of the MILP formulation devised by Chen (2008) for the single machine case. It is characterized by the use of variable E_{il}, which measures the elapsed tool life after the l-th job on machine i, i.e., the job located on slot l, is completed. Furthermore, whether a tool installed on machine i is replaced immediately after the l-th job, variable B_{il} records its total usage time. The completion time of the l-th job on machine i is denoted by F_{il}. A graphical illustration of the aforementioned decision variables is shown in Fig. 1.

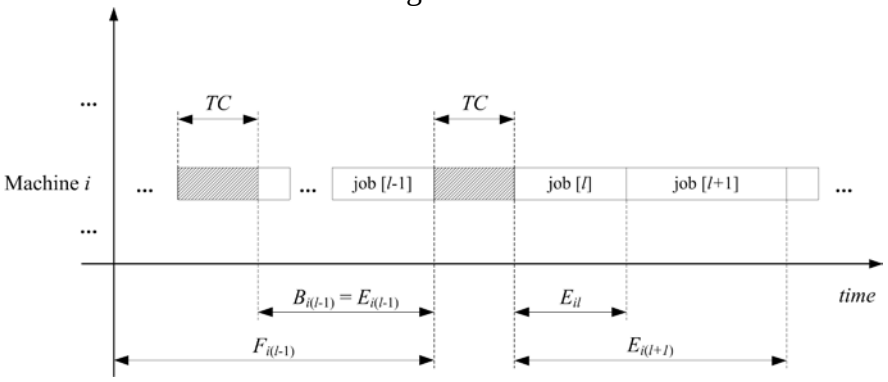


Fig. 1. Illustration of decision variables B_{il}, E_{il}, and F_{il}

Decision variables, objective function and constraints concerning PMTCS-1 model are reported in the following.

Decision variables

$$X_{ilj}\begin{cases}1 \text{ if job } j \text{ is processed in slot } l \text{ on machine } i\\ 0 \text{ otherwise}\end{cases} \tag{1}$$

$$K_{il}\begin{cases}1 \text{ if a tool change is performed on machine } i \text{ immediately after slot } l\\ 0 \text{ otherwise}\end{cases} \tag{2}$$

F_{il} completion time of the l-th job processed on machine i; (3)

T_{il} tardiness of the l-th job processed on machine i; (4)

E_{il} elapsed time between the installation of the tool employed for processing the l-th job on machine i and the completion time of the job itself; (5)

B_{il} elapsed time between the installation of the tool employed for processing the l-th job on machine i and the completion time of the job itself, whether a new tool is installed before slot l+1; (6)

$$\min\ Z = \sum_{i=1}^{m}\sum_{l=1}^{n} T_{il} \tag{7}$$

subject to

$$\sum_{i=1}^{m}\sum_{l=1}^{n} X_{ilj} = 1 \quad j = 1,2,...,n \tag{8}$$

$$\sum_{j=1}^{n} X_{ilj} \le 1 \qquad i = 1,2,...,m \;\; l = 1,2,...,n \tag{9}$$

$$E_{i1} = \sum_{j=1}^{N} p_j \cdot X_{i1j} \quad i = 1,2,...,m \tag{10}$$

$$E_{i(l-1)} + \sum_{j=1}^{N} p_j \cdot X_{ilj} \le E_{il} + M \cdot K_{i(l-1)} \quad i = 1,2,...,m \;\; l = 2,3,...,n \tag{11}$$

$$\sum_{j=1}^{N} p_j \cdot X_{ilj} \le E_{il} + M \cdot (1 - K_{i(l-1)}) \quad i = 1,2,...,m \;\; l = 2,3,...,n \tag{12}$$

$$E_{il} \le TL \quad i = 1,2,...,m \;\; l = 1,2,...,n \tag{13}$$

$$F_{il} - \sum_{j=1}^{N} d_j \cdot X_{ilj} \le T_{il} \quad i = 1,2,...,m \;\; l = 1,2,...,n \tag{14}$$

$$B_{il} \ge E_{il} - M \cdot (1 - K_{il}) \quad i = 1,2,...,m \;\; l = 1,2,...,n \tag{15}$$

$$F_{il} = \sum_{h=1}^{l-1} B_{ih} + TC \cdot \sum_{h=1}^{l-1} K_{ih} + E_{il} \quad i = 1,2,...,m \;\; l = 1,2,...,n \tag{16}$$

$$B_{il}, T_{il} \ge 0 \tag{17}$$

$$X_{ilj}, K_{il} \in \{0,1\} \tag{18}$$

Constraints (8) and (9) ensure that each job is assigned to exactly one slot on one machine, and that each slot does not hold more than one job. Constraint (10) records the time required for processing the first job on each machine. Constraints (11) and (12) measure the elapsed tool life after the job assigned to slot l on machine i is completed; the former holds whether that tool already worked other jobs before the l-th one; the latter holds if that tool was installed just before job in slot l started. Constraint (13) ensures that the usage time of tools never exceeds the provided tool life. Constraint (14) computes the tardiness of the job processed in slot l on machine i. Constraint (15) defines the value of variable B_{il}. Constraint (16) measures the completion time of the l-th job on machine i. It considers the total usage time of previously employed tools, the tool changes already performed on machine i, and the life of the new tool elapsed after job in slot l has been completed. Constraints (17) and (18) define the decision variables.

2.2. The PMTCS-2 mathematical programming model

A different strategy for handling job completion times inspired authors to generate an alternative mathematical model called PMTCS-2. This model employs a variable named RT_{il} to keep trace of the time at which machine i is available to accept a new part, once the job in slot l has been processed. In particular, if job in slot l is followed by a tool change task, then RT_{il} will include the tool change time too (see (l-1)-th job in Figure 2). Once RT_{il} is known, the completion time of the job assigned to slot l+1 may be easily computed just by adding to RT_{il} the processing time of the job itself.

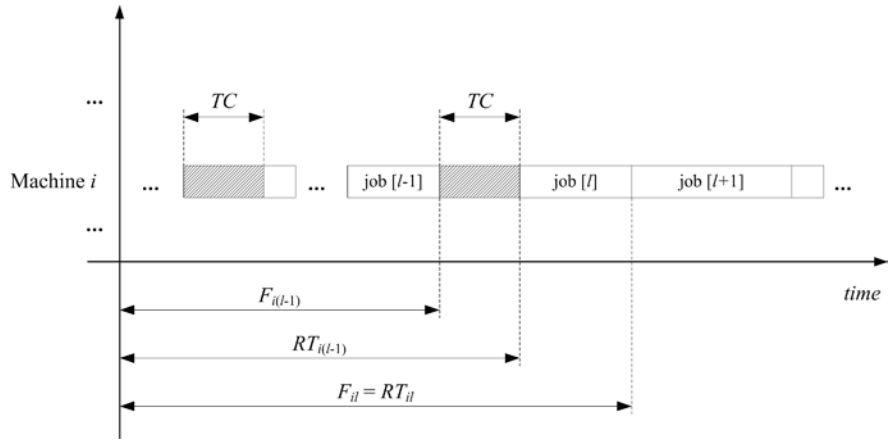


Fig. 2. Role of variables RT_{il} and F_{il}

PMTCS-2 model makes use of the decision variables defined in Eqs. (1-4), and Eq. (5). RT_{il} replaces variable B_{il} defined in Eq. (6). The objective function is the same of Eq. (7). In addition to constraints (8), (9), (10), (11), (12), (13), and (14) from PMTCS-1, the following constraints must be added to formalize PMTCS-2 model:

$$F_{i1} \geq E_{i1} \quad i = 1,2,...,m \tag{19}$$

$$F_{il} \geq RT_{i(l-1)} + \sum_{j=1}^{N} p_j \cdot X_{ilj} \quad i = 1,2,...,m \;\; l = 2,3,...,n \tag{20}$$

$$RT_{il} \geq F_{il} + TC \cdot K_{il} \quad i = 1,2,...,m \;\; l = 1,2,...,n \tag{21}$$

$$T_{il} \geq 0 \tag{22}$$

$$X_{ilj}, K_{il} \in \{0,1\} \tag{23}$$

Constraint (19) states that the completion time of the job allocated in the first slot of each machine must correspond to the tool usage time required to process that job. Constraint (20) handles the completion time of all remaining slots. Constraint (21) calculates the ready time of machine i to process the (l+1)-th job, considering the completion time of the l-th job, and the tool change time, whether it occurs immediately after slot l.

3. Comparison of MILP models

This section focuses on a comparison campaign involving the two proposed mathematical models, namely PMTCS-1 and PMTCS-2, for solving the total tardiness tool change scheduling problem on identical parallel machines. First of all, models are evaluated in terms of size complexity by calculating the number of variables and constraints characterizing each of them. Then, a quantitative analysis concerning the average CPU time required by each model to solve a proper benchmark of test cases is illustrated.

3.1. Comparison in terms of size complexity

The size complexity of a MILP model depends on the number of constraints and variables characterizing the model itself. As far as the proposed tool change scheduling problem is concerned, the only parameters affecting the size complexity of the mathematical formulations are number of jobs and machines, namely n and m, respectively.

Table 1
Size complexity of MILP models

Model	Number of binary variables	Number of continuous variables	Number of constraints	Number of big-M constraints
PMTCS-1	$mn(n+1)$	$4mn$	$n(7m+1)-m$	$2m(n-1)+mn$
PMTCS-2	$mn(n+1)$	$4mn$	$n(7m+1)-m$	$2m(n-1)$

Table 1 shows the number of binary variables, continuous variables, constraints, and big-M constraints in terms of *n* and *m* for both the devised models. It can be noted that both models hold the same number of binary and continuous variables. The total number of constraints coincide as well. However, a significant difference is ascribable to the number of constraints where the big-M parameter, i.e. the large positive number, appears. Although big-M parameters are often used in MILP theory to linearly relax dichotomous constraints, their presence strongly affects the computational complexity of the model, thus increasing the time required to convergence (Naderi and Salmasi 2012). In light of what was stated before, PMTCS-1 model is expected to be less efficient than PMTCS-2, as it holds $m \cdot n$ additional big-M constraints. The comparison analysis reported in the next subsection clearly confirms such assumption.

3.2 Comparison in terms of CPU time

In order to quantitatively assess the computational performance of the devised MILP models, an extensive comparative analysis has been performed making use of an experimental benchmark proposed by literature. Indeed, a set of problems have been generated according to the method of Chen (2008), properly adapted to the parallel-machines case. The processing times have been drawn from a uniform distribution in the range [5, 10]. Due dates have been randomly generated between $\overline{T}(1-\tau-R/2)$ and $\overline{T}(1-\tau+R/2)$, where *R* is an experimental parameter known as due date range and τ is the so-called tardiness factor. Parameter $\overline{T}$ estimates the average operating time of each machine, being calculated according to the following formula:

$$\overline{T} = \frac{\sum_{j=1}^{n} p_j}{m} \tag{24}$$

A full factorial experimental plan entailing 128 classes of problems has been generated taking into account the following factors and levels of variation:

- number of jobs (*n*): 4 levels (6, 8, 12, 15);
- number of machines (*m*): 2 levels (2, 3);
- tool life (*TL*): 2 levels (10, 18);
- tool change time (*TC*): 2 levels (2, 4);
- due date range (*R*): 2 levels (0.2, 0.6);
- tardiness factor (τ): 2 levels (0.2, 0.6).

For each class of problems, 5 instances have been randomly generated in terms of p_j. Therefore, a total of $128 \cdot 5 = 640$ test cases have been evaluated on the whole. Each instance has been solved through both the proposed MILP models, running on an IBM ILOG CPLEX® 12.6 (64 bit) platform installed on a workstation powered by two quad-core 2,39 GHz processors with 24 GB RAM. Two distinct performance indicators have been considered for the comparison analysis, the former being the CPU times needed by each model to converge, the latter being the so-called Relative Percentage Deviation (*RPD*):

$$RPD = 100 \cdot \frac{Model_{time} - Best_{time}}{Best_{time}}, \tag{25}$$

where $Model_{time}$ is the CPU time required by either PMTCS-1 or PMTCS-2, and $Best_{time}$ is the lowest CPU time among those reached by the two MILP models for the same test case.

Tables 2, 3, 4, and, 5 display the average CPU times (in seconds) and the average *RPD* values obtained by the two models for instances with 6, 8, 12, and 15 jobs, respectively. Within each table, the lowest average CPU time and the lowest average *RPD* obtained for each class of problems are denoted in bold.

Table 2
Computational results for problems with 6 jobs

m	*TL*	TC	τ	*R*	Average CPU time (s)		Average *RPD*	
					PMTCS-1	PMTCS-2	PMTCS-1	PMTCS-2
2	10	2	0.2	0.2	0.60	**0.46**	79.02	**13.55**
				0.6	1.56	**0.38**	315.12	**0.00**
			0.6	0.2	0.74	**0.46**	285.74	**3.06**
				0.6	1.61	**0.06**	3319.90	**0.00**
		4	0.2	0.2	**0.44**	0.63	**0.00**	43.60
				0.6	**0.60**	0.71	**11.89**	34.76
			0.6	0.2	**0.42**	0.43	30.29	**18.05**
				0.6	0.34	**0.06**	472.72	**0.00**
	18	2	0.2	0.2	0.47	**0.43**	143.52	**14.21**
				0.6	0.97	**0.24**	746.09	**0.00**
			0.6	0.2	0.60	**0.16**	462.25	**0.00**
				0.6	0.87	**0.06**	1591.11	**0.00**
		4	0.2	0.2	**0.43**	0.51	**46.43**	49.48
				0.6	0.47	**0.22**	214.86	**4.66**
			0.6	0.2	0.34	**0.13**	181.46	**0.00**
				0.6	0.52	**0.06**	773.90	**0.00**
3	10	2	0.2	0.2	0.60	**0.40**	65.69	**4.45**
				0.6	1.72	**0.21**	1234.82	**0.00**
			0.6	0.2	0.60	**0.48**	181.27	**14.44**
				0.6	1.47	**0.39**	1485.13	**26.95**
		4	0.2	0.2	0.58	**0.50**	18.67	**3.30**
				0.6	0.64	**0.47**	70.78	**0.00**
			0.6	0.2	0.55	**0.49**	24.33	**4.05**
				0.6	0.40	**0.10**	316.07	**0.00**
	18	2	0.2	0.2	0.60	**0.48**	65.00	**1.14**
				0.6	1.17	**0.43**	354.24	**0.00**
			0.6	0.2	0.70	**0.09**	677.19	**0.00**
				0.6	1.19	**0.10**	1087.91	**0.00**
		4	0.2	0.2	0.70	**0.60**	114.51	**20.13**
				0.6	0.67	**0.24**	356.13	**0.00**
			0.6	0.2	0.47	**0.37**	168.81	**17.53**
				0.6	0.55	**0.16**	418.56	**0.00**
grand average					0.74	**0.33**	478.54	**8.54**

Table 3
Computational results for problems with 8 jobs

m	*TL*	TC	τ	*R*	Average CPU time (s)		Average *RPD*	
					PMTCS-1	PMTCS-2	PMTCS-1	PMTCS-2
2	10	2	0.2	0.2	3.54	**2.07**	77.23	**0.00**
				0.6	12.10	**2.53**	385.97	**0.00**
			0.6	0.2	2.95	**0.94**	224.09	**0.00**
				0.6	7.09	**0.70**	903.61	**0.00**
		4	0.2	0.2	**1.53**	1.76	**11.88**	31.40
				0.6	4.83	**2.69**	104.28	**1.83**
			0.6	0.2	1.84	**1.48**	55.78	**45.18**
				0.6	2.30	**1.09**	998.74	**0.00**
	18	2	0.2	0.2	1.46	**1.30**	26.14	**6.72**
				0.6	2.42	**1.25**	116.85	**0.00**
			0.6	0.2	1.70	**0.47**	827.76	**0.00**
				0.6	5.05	**0.57**	1144.53	**0.00**
		4	0.2	0.2	**1.37**	1.66	**0.00**	25.23
				0.6	2.29	**1.41**	69.01	**0.00**
			0.6	0.2	0.91	**0.86**	19.65	**18.61**
				0.6	1.71	**0.73**	136.19	**0.00**
3	10	2	0.2	0.2	**2.02**	2.09	**6.05**	7.46
				0.6	16.04	**1.92**	1276.72	**0.00**
			0.6	0.2	1.82	**0.57**	661.80	**0.00**
				0.6	5.80	**0.73**	701.65	**0.00**
		4	0.2	0.2	2.44	**1.66**	49.61	**0.00**
				0.6	5.13	**3.33**	65.16	**5.46**
			0.6	0.2	2.25	**0.97**	166.26	**2.59**
				0.6	3.73	**1.49**	207.38	**0.00**
	18	2	0.2	0.2	1.68	**1.16**	48.86	**0.00**
				0.6	2.91	**1.30**	119.29	**0.00**
			0.6	0.2	2.39	**0.55**	323.51	**0.00**
				0.6	14.45	**0.85**	1494.23	**0.00**
		4	0.2	0.2	**1.84**	1.92	**9.55**	19.98
				0.6	2.22	**1.20**	92.86	**0.00**
			0.6	0.2	1.61	**0.97**	78.43	**2.16**
				0.6	2.30	**0.85**	172.69	**0.00**
grand average					3.80	**1.35**	330.49	**5.21**

Table 4
Computational results for problems with 12 jobs

m	*TL*	TC	τ	*R*	Average CPU time (s)		Average *RPD*	
					PMTCS-1	PMTCS-2	PMTCS-1	PMTCS-2
2	10	2	0.2	0.2	45.27	**8.19**	386.52	**0.00**
				0.6	1643.08	**9.93**	16608.40	**0.00**
			0.6	0.2	54.62	**5.87**	640.05	**0.00**
				0.6	2772.98	**11.67**	38104.28	**0.00**
		4	0.2	0.2	15.19	**7.43**	117.37	**1.64**
				0.6	116.85	**15.78**	860.40	**0.00**
			0.6	0.2	11.87	**4.29**	221.80	**0.00**
				0.6	101.40	**5.92**	3583.56	**0.00**
	18	2	0.2	0.2	3.75	**2.72**	39.43	**0.00**
				0.6	122.73	**3.05**	3634.34	**0.00**
			0.6	0.2	8.02	**2.23**	288.39	**0.00**
				0.6	1804.08	**2.93**	64345.17	**0.00**
		4	0.2	0.2	3.66	**2.77**	37.81	**0.00**
				0.6	127.81	**5.12**	2548.59	**0.00**
			0.6	0.2	3.65	**2.14**	80.46	**0.00**
				0.6	30.13	**3.05**	938.42	**0.00**
3	10	2	0.2	0.2	48.62	**15.42**	333.28	**0.00**
				0.6	779.75	**14.96**	8071.05	**0.00**
			0.6	0.2	111.10	**5.06**	2143.50	**0.00**
				0.6	862.16	**8.35**	27416.18	**0.00**
		4	0.2	0.2	12.44	**10.66**	33.89	**6.38**
				0.6	160.53	**25.95**	487.81	**0.00**
			0.6	0.2	20.63	**4.77**	332.17	**0.00**
				0.6	60.56	**12.23**	579.87	**0.00**
	18	2	0.2	0.2	4.45	**3.17**	41.89	**1.33**
				0.6	627.87	**3.47**	18516.18	**0.00**
			0.6	0.2	23.68	**3.73**	507.90	**0.00**
				0.6	1972.27	**2.71**	92404.20	**0.00**
		4	0.2	0.2	4.87	**3.87**	29.99	**3.20**
				0.6	15.99	**4.57**	233.69	**0.00**
			0.6	0.2	6.96	**2.59**	188.45	**0.00**
				0.6	88.46	**3.52**	3131.30	**0.00**
grand average					364.54	**6.82**	8965.20	**0.39**

Table 5
Computational results for problems with 15 jobs

m	*TL*	TC	τ	*R*	Average CPU time (s)		Average *RPD*	
					PMTCS-1	PMTCS-2	PMTCS-1	PMTCS-2
2	10	2	0.2	0.2	2304.81	**398.93**	991.26	**0.00**
				0.6	3601.03	**179.18**	7178.36	**0.00**
			0.6	0.2	2873.38	**38.42**	14217.98	**0.00**
				0.6	3602.19	**309.04**	6845.90	**0.00**
		4	0.2	0.2	211.53	**78.41**	164.41	**0.62**
				0.6	2052.42	**247.43**	1377.76	**0.00**
			0.6	0.2	312.97	**25.91**	1840.63	**0.00**
				0.6	2525.35	**16.07**	20358.86	**0.00**
	18	2	0.2	0.2	18.58	**6.12**	218.13	**0.00**
				0.6	1103.83	**7.91**	14478.03	**0.00**
			0.6	0.2	200.74	**4.74**	4827.97	**0.00**
				0.6	3600.75	**6.51**	62470.95	**0.00**
		4	0.2	0.2	11.09	**6.49**	90.95	**0.00**
				0.6	807.39	**15.69**	3794.84	**0.00**
			0.6	0.2	30.70	**4.03**	601.98	**0.00**
				0.6	271.38	**8.21**	3451.33	**0.00**
3	10	2	0.2	0.2	795.90	**395.18**	99.35	**5.37**
				0.6	3601.80	**112.50**	4403.33	**0.00**
			0.6	0.2	987.03	**50.79**	9884.07	**0.00**
				0.6	3099.27	**811.78**	6363.95	**0.00**
		4	0.2	0.2	193.08	**137.79**	214.22	**7.07**
				0.6	1557.99	**208.38**	1378.08	**0.00**
			0.6	0.2	491.27	**13.27**	6033.56	**0.00**
				0.6	1404.10	**71.89**	3414.17	**0.00**
	18	2	0.2	0.2	33.15	**5.24**	475.00	**0.00**
				0.6	2829.96	**7.52**	41039.08	**0.00**
			0.6	0.2	181.82	**8.32**	3873.00	**0.00**
				0.6	3602.55	**13.11**	39121.61	**0.00**
		4	0.2	0.2	59.01	**10.59**	565.06	**0.00**
				0.6	63.14	**9.54**	554.62	**0.00**
			0.6	0.2	13.62	**3.54**	284.55	**0.00**
				0.6	1074.33	**8.38**	13996.58	**0.00**
grand average					1359.88	**100.65**	8581.55	**0.41**

The aforementioned numerical results confirm the difference between the two proposed models under the computational time viewpoint. As concerns 6-job related instances, PMTCS-2 model is faster than

PMTCS-1 in 28 classes of problems out of 32, also assuring a lower *RPD* value in 29 classes out of 32. However, there is a negligible difference between the two proposed approaches as both models need less than 2 seconds to solve any instance. Similar remarks may be done for 8-job related problems, where PMTCS-2 outperforms PMTCS-1 in 28 classes of problems out of 32 under both the average CPU time and the *RPD* value viewpoints, though the average CPU time is always lower than 17 seconds. A clearer difference emerges with reference to 12-job problems, where PMTCS-1 takes, on average, more than 6 minutes to find the global optimum against 7 seconds required by PMTCS-2. Looking at the average *RPD* values, PMTCS-2 outperforms the first model in all the provided classes of problems, thus pointing out a smaller computational time spread over the minimum. The advantage of using PMTCS-2 instead of PMTCS-1 is more evident for 15-job instances, since PMTCS-2 outperforms PMTCS-1 in all classes of problems, assuring an average CPU time of 100.65 seconds against 1,359.88 seconds required by PMTCS-1.The *RPD* performance indicator confirms the difference between the two proposed models as denoted by the corresponding bold values in Table 5. The grand average values arising from Tables 2 to 5 have been reported in Fig. 3, with the aim of highlighting the average computational time required by the two proposed models as the number of jobs increased.

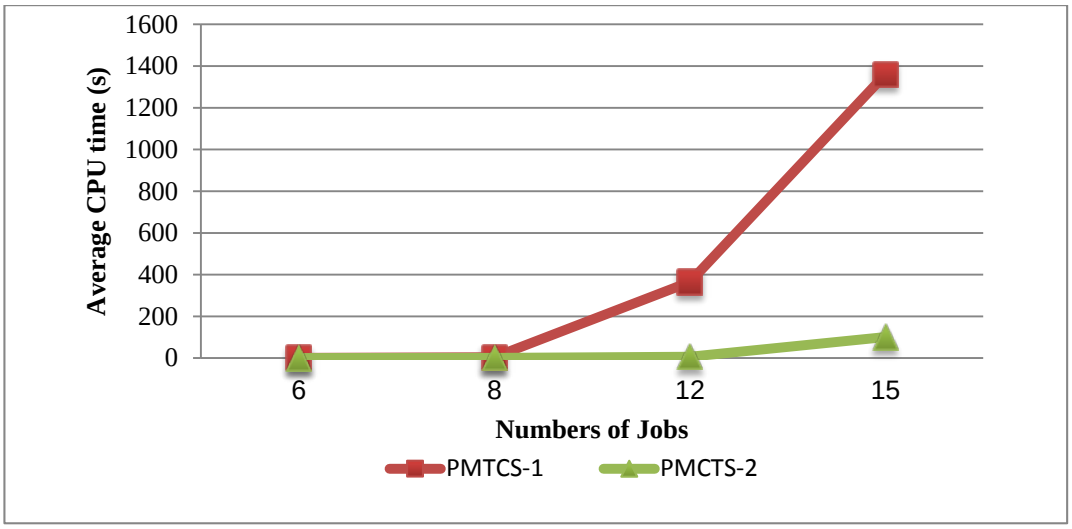
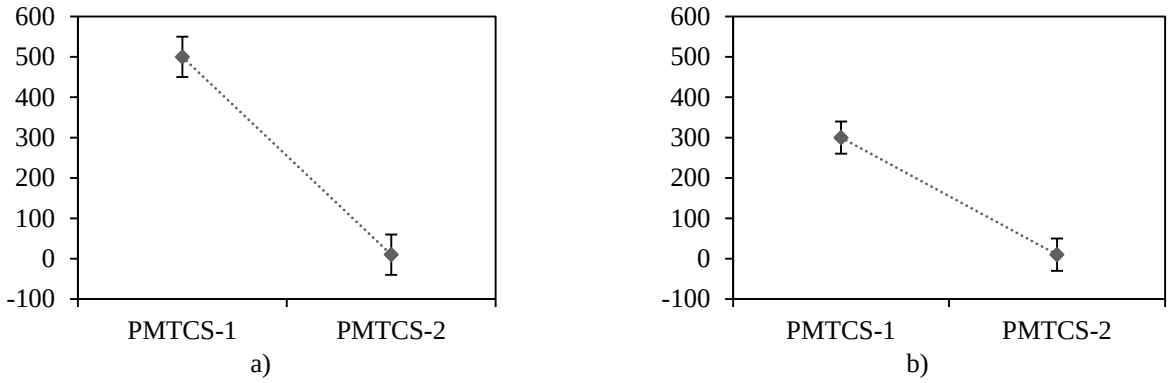


Fig. 3. Average computational times at varying number of jobs for each model

In order to infer a statistical conclusion over the difference observed between the tested MILP formulations, an ANOVA analysis (Montgomery, 2007) has been performed through Stat-Ease Design Expert® 7.0.0 commercial tool. In particular, the LSD intervals at 95% confidence level for the *RPD*s connected to each model has been computed. The corresponding charts, which refer to problems with 6, 8, 12, and 15 jobs are reported in Figure 4-a), -b), -c) and -d), respectively. Since the two LSD bars of each chart do not overlap, it can be stated that PMTCS-2 outperforms PMTCS-1 in a statistically significant manner, regardless of the number of scheduled jobs.



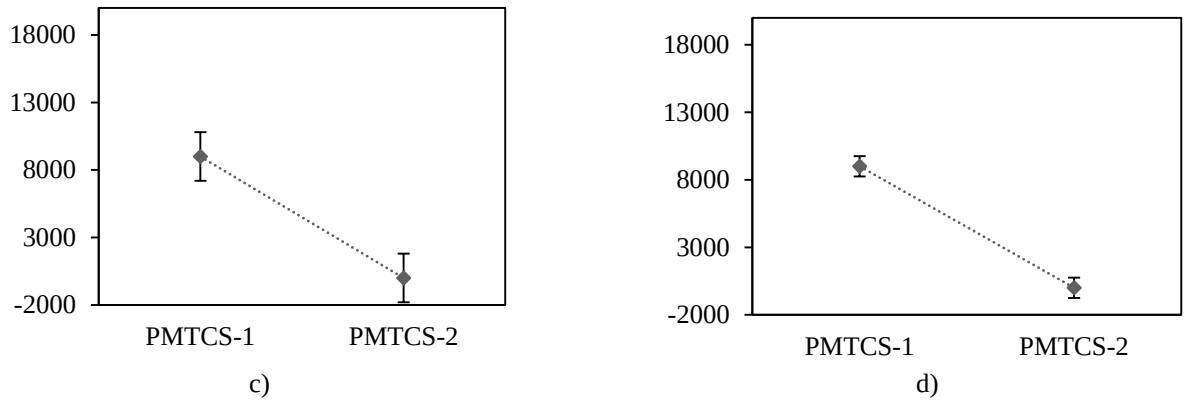


Fig. 4. Means plot with 95% LSD intervals for problems with a) 6 jobs; b) 8 jobs; c) 12 jobs; d) 15 jobs)

4. Conclusions

In this paper, the tool change scheduling problem for a manufacturing system composed by *m* identical parallel machines has been tackled under the total tardiness minimization viewpoint. Two mixed integer linear programming models, namely PMTCS-1 and PMTCS-2, have been developed for the general case of unequal job processing times, the main difference being the modeling strategy adopted to define job completion times. A complexity size analysis revealed the efficiency of PMTCS-2 in solving the problem at hand, due to the lower number of employed big-M constraints. A comprehensive experimental campaign performed on the basis of a well-known benchmark of test cases, properly adapted to the PMTCS problem, revealed the superiority of PMTCS-2 model under the computational time viewpoint. A proper ANOVA analysis confirmed a significant statistical difference between the two proposed models. Future research could entail the development of metaheuristic techniques to properly solve larger-sized instances of the proposed PMTCS problem.

References

Akturk, M. S., Ghosh, J. B., & Gunes, E. D. (2004). Scheduling with tool changes to minimize total completion time: basic results and SPT performance. *European Journal of Operational Research*, *157*(3), 784-790.

Akturk, M. S., Ghosh, J. B., & Kayan, R. K. (2007). Scheduling with tool changes to minimize total completion time under controllable machining conditions. *Computers & operations research*, *34*(7), 2130-2146.

Biskup, D., Herrmann, J., & Gupta, J. N. (2008). Scheduling identical parallel machines to minimize total tardiness. *International Journal of Production Economics*, *115*(1), 134-142.

Chen, J. S. (2008). Optimization models for the tool change scheduling problem. *Omega*, *36*(5), 888-894.

Edis, E. B., Oguz, C., & Ozkarahan, I. (2013). Parallel machine scheduling with additional resources: Notation, classification, models and solution methods. *European Journal of Operational Research*, *230*(3), 449-463.

Gray, A. E., Seidmann, A., & Stecke, K. E. (1993). A synthesis of decision models for tool management in automated manufacturing. *Management science*, *39*(5), 549-567.

Hertz, A., Laporte, G., Mittaz, M., & Stecke, K. E. (1998). Heuristics for minimizing tool switches when scheduling part types on a flexible machine. *IIE transactions*, *30*(8), 689-694.

Hwang, H. C., & Chang, S. Y. (1998). Parallel machines scheduling with machine shutdowns. *Computers & Mathematics with Applications*, *36*(3), 21-31.

Hwang, H. C., Lee, K., & Chang, S. Y. (2005). The effect of machine availability on the worst-case performance of LPT. *Discrete Applied Mathematics*, *148*(1), 49-61.

Lee, C. Y. (1996). Machine scheduling with an availability constraint. *Journal of global optimization*, *9*(3-4), 395-416.

Lee, C. Y. (1991). Parallel machines scheduling with nonsimultaneous machine available time. *Discrete Applied Mathematics*, *30*(1), 53-61

Lee, C. Y., Lei, L., & Pinedo, M. (1997). Current trends in deterministic scheduling. *Annals of Operations Research*, *70*, 1-41.

Liao, C. J., Shyur, D. L., & Lin, C. H. (2005). Makespan minimization for two parallel machines with an availability constraint. *European Journal of Operational Research*, *160*(2), 445-456.

Lin, C. H., & Liao, C. J. (2007). Makespan minimization for two parallel machines with an unavailable period on each machine. *The International Journal of Advanced Manufacturing Technology*, *33*(9-10), 1024-1030.

Low, C., Ji, M., Hsu, C. J., & Su, C. T. (2010). Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance. *Applied Mathematical Modelling*, *34*(2), 334-342.

Ma, Y., Chu, C., & Zuo, C. (2010). A survey of scheduling with deterministic machine availability constraints. *Computers & Industrial Engineering*, *58*(2), 199-211.

Montgomery, D. C., (2007). *Design and analysis of experiments*. New York: Wiley.

Naderi, B., & Salmasi, N. (2012). Permutation flowshops in group scheduling with sequence-dependent setup times. *European Journal of Industrial Engineering*, *6*(2), 177-198.

Pinedo, M. L. (2012). *Scheduling: theory, algorithms, and systems*. Springer Science & Business Media.

Kan, A. R. (1976). *Machine scheduling problems: classification, complexity and computations*. Springer Science & Business Media.

Schmidt, G. (2000). Scheduling with limited machine availability. *European Journal of Operational Research*, *121*(1), 1-15.

Tang, C. S., & Denardo, E. V. (1988). Models arising from a flexible manufacturing machine, part I: minimization of the number of tool switches. *Operations research*, *36*(5), 767-777.

Wang, J. B., & Wei, C. M. (2011). Parallel machine scheduling with a deteriorating maintenance activity and total absolute differences penalties. *Applied Mathematics and Computation*, *217*(20), 8093-8099.

Xu, D., Sun, K., & Li, H. (2008). Parallel machine scheduling with almost periodic maintenance and non-preemptive jobs to minimize makespan. *Computers & operations research*, *35*(4), 1344-1349.

Xu, D., Liu, M., Yin, Y., & Hao, J. (2013). Scheduling tool changes and special jobs on a single machine to minimize makespan. *Omega*, *41*(2), 299-304.

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

Zarook, Y., & Abedi, M. (2014). JIT-scheduling in unrelated parallel-machine environment with aging effect and multi-maintenance activities. *International Journal of Services and Operations Management*, *18*(1), 99-113.