Assessing scheduling policies in a permutation flowshop with common due dates *

Paz Perez-Gonzalez† Jose M. Framinan

Industrial Management, School of Engineering,

University of Seville. Camino de los Descubrimientos s/n. 41092 Seville, Spain

Abstract

This paper focuses onto a situation arising in most real-life manufacturing environments when scheduling has to be performed periodically. In such scenario, different scheduling policies can be adopted, being perhaps the most common to assume that, once a set of jobs has been scheduled, their schedule cannot be modified ('frozen' schedule). This implies that, when the next set of jobs is to be scheduled, the resources may not be fully available. Another option is assuming that the schedule of the previously scheduled jobs can be modified as long as it does not violate their due date, which has been already possibly committed to the customer. This policy leads to a so-called multi-agent scheduling problem. The goal of this paper is to discern when each policy is more suitable for the case of a permutation flowshop with common due dates. To do so, we carry out an extensive computational study in a testbed specifically designed to control the main factors affecting the policies, so we analyze the solution space of the underlying scheduling problems. The results indicate that, when the due date of the committed jobs is tight, the multi-agent approach does not pay off in view of the difficulty of finding feasible

*This is an Accepted Manuscript of an article published by Taylor and Francis in International Journal of Production Research, in 2015 (in Press). available online: http://dx.doi.org/10.1080/00207543.2014.994077

†Corresponding author. Tel. +3495 448 7327; fax: +3495 448 7329. E-mail address: pazperez@esi.us.es

solutions. Moreover, in such cases, the policy of 'freezing' the schedule of the jobs leads to a very simple scheduling problem with many good/acceptable solutions. In contrast, when the due date has a medium/high slack, the multi-agent approach is substantially better. Nevertheless, in this latter case, in order to perceive the full advantage of this policy, powerful solution procedures have to be designed, as the structure of the solution space of the latter problem makes extremely hard to find optimal/good solutions.

1 Introduction

In this paper, we address the problem of scheduling jobs on a permutation flowshop in a cyclic manner. More specifically, a set of jobs has been scheduled in a previous decision interval when a new set of jobs has to be scheduled in the current decision interval. In this situation, two main scheduling policies can be adopted. The first policy is to assume that the schedule of the jobs in the first set cannot be modified (i.e. the jobs in the first set are said to be 'frozen'), so in order to schedule the jobs in the second set it has to be assumed that not all resources are available from the beginning of the scheduling interval, as some of them may be busy with jobs belonging to the first set. Thus, the scheduling problem that corresponds to this policy is denoted Availability Scheduling Problem (ASP in the following), see e.g. Lee (1997); Perez-Gonzalez and Framinan (2009).

A second scheduling policy is to assume that the schedule of the first set can be modified as long as it does not violate their already established due date, and therefore jobs in both sets are scheduled together, although the objective is different for each set. In such case, the corresponding scheduling problem is a Multi-agent Scheduling Problem (MSP in the following), see e.g. Agnetis et al. (2014).

Clearly, both policies have their advantages: while the second policy may lead to a better utilization of resources and a potentially higher performance, the greater computational burden to solve these scheduling problems may not pay off, also taking into account that, from a practical viewpoint, changing an existing schedule introduces a higher nervousness in the shop floor that might have negative implications.

Therefore, the aim of our research is to establish the relative advantages of each policy, both in terms of the performance of the schedule, and of the solution procedures for each one of the resulting scheduling problems. More specifically, we wish to investigate the conditions that makes more convenient to address the scheduling of existing jobs in a multi-agent context instead of 'freezing' their schedule. The problem addressed here is related to order management in production companies, and to the quality of the service offered to the customer, since it considers different sets of jobs (for example different orders belonging to different customers) each one with its own objective (see Agnetis et al., 2014; Perez-Gonzalez and Framinan, 2014).

The decision problem considered may appear in a great diversity of productive layouts. Here we focus onto the permutation flowshop layout, as it is a popular setting both in practise and research. The flowshop implies a natural ordering of the machines in the shop in such a way that the jobs go through the same machines in the same order. In general, there are $(n!)^m$ schedules to be considered, with n the number of jobs and m the number of machines. However, there is a simplified version of the problem applicable to many situations in which it can be assumed that the processing sequence of the jobs is the same for all machines (i.e. permutation flowshop) and hence only $(n!)$ schedules have to be considered. In addition, we also consider that the jobs in the first set have a common due date.

Also let us note that, in theory, there is (at least) a third scheduling policy at hand, i.e. not to start processing the second set of jobs until the jobs in first one have been completed. Although this might seem unrealistic, this policy represents indeed the classical assumption of most scheduling literature, in which all resources are free from the beginning of the scheduling interval. In our paper we will also investigate this policy (which leads to the Classical Scheduling Problem, denoted in the following as CSP) in order to have a base case to compare the other policies.

It is clear that, when adopting the multi-agent approach, the common due date for the jobs in the first set is a parameter that may influence the analysis of the problems, as a tighter common due date implies a higher number of unfeasible schedules for the jobs in the second set. Therefore, we analyze existing methods in the literature for setting a

common due date and propose a new one that will allow us to control the slack in the due dates so to perform an exhaustive analysis of the different problems. With this method, a design of the experiments is conducted. The results show that, when the slack of the jobs in the first set is sufficiently small, the multi-agent approach does not pay off. Indeed, in such cases, the ASP approach may be very interesting also because the solution space is rather 'flat' and most schedules are of good quality, thus making this decision problem very easy. In contrast, higher slacks make the MSP approach more convenient, although the solution space of this problem is such that extremely good algorithms are required to fully grasp their advantages.

The remainder of the paper is as follows: The policies and their corresponding scheduling problems are presented in Section 2 together with the notation employed. Section 3 reviews existing methods for common due date generation, and explains the common due date generation method used in the experiments. Section 4 analyzes the structure of solutions of the problems in order to compare them, and finally, conclusions are summarized in Section 5.

2 Problem statement

In our problem, we consider a permutation flowshop where jobs must be scheduled in a periodical manner, i.e.: at time T, the Decision Maker should schedule orders (jobs) that entered the system from $T - H$ to T, being H the decision period. This procedure is repeated every H periods. Note that H does not have to be fixed and may be different for each decision interval. For each period, there are new jobs entering into the system. We model a situation in which the jobs belong to a single customer, or are produced in a single batch, distinguishing two sets of jobs: the old jobs belong to previously scheduled orders, denoted as the set J_O with n_O jobs, and a set of new jobs, J_N, with n_N jobs.

As already discussed in Section 1, different scheduling policies can be considered:

- To set the starting times of jobs in J_N once all jobs in J_O are processed, so all machines are available because there are not jobs in J_O scheduled and the system is empty.

- To start scheduling jobs in J_N as soon as possible, so some jobs in J_O are not completed and machines may be busy processing these jobs. In this case at least two options can be considered:
  - The schedule of jobs in J_O cannot change. We denote this option as 'frozen' jobs, since their schedule remains the same (see e.g. Akkan, 1997, and Frederix, 2001). Machine i is then not available until the availability instant a_i, $i \in \{1, \ldots, m\}$, defined by the completion times of jobs in J_O on each machine.
  - The schedule of jobs in J_O can change and these jobs can be merged and scheduled together with the new set of jobs. In this case, the set $J = J_O \bigcup J_N$ with $n = n_O + n_N$ jobs is scheduled.

The adoption of each scheduling policy leads to different scheduling scenarios. Their features are summarised in Table 1.

Machines	**Processing of J_O**	**Scheduling policy**	**Scenario**
Available	Finished	$\Rightarrow$ Empty	Classical Scheduling
Unavailable	Unfinished	$\Rightarrow$ Frozen	Availability Scheduling
		$\Rightarrow$ Modifiable	Multi-agent Scheduling

Table 1: Identified scenarios depending on the scheduling policies

The three scenarios identified in Table 1 lead to the need to solve three problems. For each scenario, we consider a classical objective for the jobs in the second set, i.e. the minimization of the makespan or maximum completion time and, depending on the scheduling policy, not to violate the already established due date of jobs in the first set. More specifically, the objective considered is to minimize the makespan of jobs in J_N, denoted as $C_{max}^{J_N}$ (in order to provide a tight due date for these jobs), and that the common due date of jobs in J_O cannot be violated (according to the delivery reliability). This consideration implies that the completion times of jobs in J_O should be lesser or equal than their common due date. This is equivalent, for example, to state that the maximum tardiness for jobs in J_O should be zero, i.e $T_{max}^{J_O} = \max\{T_j^{J_O}\} = 0$ with $T_j^{J_O} = \max\{0, C_j^{J_O} -$

$d\}$ the tardiness of the job $j \in J_O$, and $C_j^{J_O}$ the completion time of the job $j \in J_O$ in the last machine. Note that it is also equivalent to state that the total tardiness for jobs in J_O is zero, i.e. $T^{J_O} = \sum_{j \in J_O} T_j^{J_O} = 0$. The scheduling problems corresponding to each scenario are summarised in Table 2, indicating the scheduling scenario, the influence of the common due date of the old jobs, the set of jobs to be scheduled, the objective of the problem as consequence of each scenario, and the corresponding notation according to Graham et al. (1979).

Scenario	**Due date of J_O**	**Jobs to schedule**	**Objective**	**Identified problem**
Classical Sch.	$\times$	J_N	$C_{max}^{J_N}$	$CSP : Fm\|prmu\|C_{max}$
Availability Sch.	$\times$	J_N	$C_{max}^{J_N}$	$ASP : Fm\|prmu, a_i\|C_{max}$
Multi-Agent Sch.	Deadline	$J_O \bigcup J_N$	$C_{max}^{J_N}/T_{max}^{J_O} = 0$	$MSP : Fm\|prmu, d_j = d\|0(C_{max}^{J_N}/T_{max}^{J_O})$

Table 2: Problems identified for the three scenarios

The classical scheduling scenario, where the old jobs have been processed (J_O is considered as empty), corresponds to the classical permutation flowshop problem or $Fm|prmu|C_{max}$, denoted as CSP in this paper. This well-known problem has been intensively addressed during the last 50 years (see e.g. Gupta and Stafford, 2006). It is solvable to optimality in polynomial time when there are two machines by Johnson's Algorithm (JA), or three machines under specific constraints on job processing times (Johnson, 1954). However, it is NP-complete in the strong sense when there are more than two machines (Garey et al., 1976), so the search for an optimal solution is of more theoretical than practical importance (Nagano et al., 2008). CSP has been analysed in many references (some recent works are Ribas et al., 2010; Tzeng and Chen, 2012; Fernandez-Viagas and Framinan, 2014).

In the availability scheduling scenario, jobs in J_O are scheduled, and their schedule is considered as frozen. Then we have a machine availability constraint problem or $Fm|prmu, a_i|C_{max}$, denoted as ASP, where a_i is the availability instant for each machine $i \in \{1, \ldots, m\}$ given by the completion times of jobs in J_O on each machine. As jobs in J_O are frozen, their common due date is fulfilled and it does not have influence on the

objective. This problem is also solvable in polynomial time by JA for two machines (Lee, 1997). The case for more than two machines is shown to be strongly NP-hard and it has been analyzed by Perez-Gonzalez and Framinan (2009), where fast heuristic methods were proposed to solve it.

Finally, if the schedule of jobs in J_O can be changed, then we have a multi-agent scheduling problem. The problem $Fm|prmu, d_j = d|0(C^{J_N}_{max}/T^{J_O}_{max})$, denoted MSP, follows the notation by T'kindt and Billaut (2002) for multi-criteria problems, where $d_j = d$ specifies the use of the common due date. This problem is strongly NP-hard for more than two machines, since if we consider $J_O = \varnothing$, then it is reduced to the CSP, which is known to be strongly NP-hard. For the case with two machines it is NP-hard too (see Luo et al 2012). Other works considering multi-agent scheduling problems in permutation flowshop are Huynh-Tuong and Soukhal (2009); Khelifati and Bouzid-Sitayeb (2011a,b); Lee et al. (2011); Luo et al. (2011); Mor and Mosheiov (2014); Xu and Lei (2014).

Since the common due date of jobs in J_O is a key parameter for MSP, Section 3 analyses the different methods available in the literature to generate the common due date of jobs in J_O, in order to provide a realistic due date for our problem.

3 Common due date generation

As mentioned before, in order to analyse and compare CSP, ASP and MSP we need a realistic common due date for jobs in J_O for MSP, since in our problem formulation, the due date is considered a parameter. Thus, we investigate different methods to provide a suitable common due date for jobs in J_O. A tight common due date with respect to the makespan of J_O will not allow rescheduling them together with J_N, so the problem would be more similar to ASP than to MSP. On the other hand, a loose common due date for J_O would not be realistic, and the due date will be verified for any schedule, so the problem will turn into a classical permutation flowshop problem CSP.

As a consequence, we need realistic common due dates generation methods in order to provide due date reasonableness, which is defined as a measure of the due date performance reflected on the capability of the system to achieve successfully an arbitrary set of due dates

(Vig and Dooley, 1991). Achieving reasonableness of due dates is not a trivial issue. In the order capture process (Framinan and Leisten, 2010), it is possible to identify different ways to obtain due dates. On one hand, they can be set by the customer; and, on the other hand, they can be determined according to different mechanism: a) taking into account scheduling decisions (due date assignment and scheduling); or b) considering certain job- and workload related parameters (due date assignment). In our problem, a due date lower than the optimal makespan is no suitable, since it will be violated for any schedule. Therefore, we opt for obtaining realistic common due dates by due date assignment and scheduling, setting due dates taking into account the makespan value obtained by solving the CSP for the jobs in J_O. As the CSP is NP-hard when the number of machines is greater or equal than three, we cannot obtain optimal values in all the cases. In order to guarantee feasibility, the makespan value may be either the optimal makespan or the best value determined by a given algorithm applied to the jobs in J_O. For this reason, we will consider due dates higher than the optimal/best makespan obtained as realistic due dates.

In the literature about flowshop scheduling problems with due date related objectives, we have identified a number of methods to generate due dates in order to use them as parameters for our problem. The methods available to set a common due date in the flowshop scheduling literature are due to Blazewicz et al. (2004, 2008); Sarper (1995); Sakuraba et al. (2009) and Della Croce et al. (2000). Among them, we will not consider the proposals by Blazewicz et al. (2004, 2008) and Della Croce et al. (2000), since these methods provide due dates lower than the makespan of the considered jobs. Therefore, the method SAR proposed by Sarper (1995) for loose common due date and SRS by Sakuraba et al. (2009) for unrestricted common due date will be tested, in order to determine if they will be useful for our problem or the due dates are too loose. In addition, Framinan and Leisten (2008) review different procedures for establishing due dates in permutation flowshop. These methods are: AR by Armentano and Ronconi (1999), GS by Gelders and Sambandam (1978) and HR by Hasija and Rajendran (2004). Note that these methods do not generate a common due date, but they can be adapted in a straightforward manner. More specifically:

- The method AR proposed by Armentano and Ronconi (1999) is based on the uniform

distribution $U[P\cdot(1-T-\frac{r}{2}), P\cdot(1-T+\frac{r}{2})]$. The tardiness factor T is a constant in the original method that has been eliminated in our experiments since tardiness is not allowed for our problem. Moreover, the factor P which depends on the processing times has been replaced by the makespan of jobs in J_O in order to provide due dates closer to the makespan value of jobs in J_O as mentioned before. With the original lower bound $(1-\frac{r}{2})$ we would obtain values lower than the makespan, so it has been replaced by $(1+\frac{r}{4})$, which provides values higher than the makespan when multiplied by $C_{max}^{J_O}$. Then, as the due date does not depend on the jobs j, the distribution obtained can be applied for the common due date case, being drawn from a $U[C_{max}^{J_O}(1+\frac{r}{4}), C_{max}^{J_O}(1+\frac{r}{2})]$ distribution. We have selected the same values for r than in Armentano and Ronconi (1999), i.e.: $r = 0.6$ and $r = 1.2$.

- The method proposed by Gelders and Sambandam (1978) is $d_j \sim U\left[\sum_{i=1}^{m} p_{ij}, \sum_{i=1}^{m} p_{ij} + 0.5\cdot p_{m,\bullet}\right]$ with $p_{m,\bullet}$ the mean processing time of jobs in the machine m. The sum of the processing times for each job has been replaced in order to provide a due date closer to the makespan of jobs in J_O. Therefore, the due date is generated according to the distribution $U[C_{max}^{J_O}, C_{max}^{J_O} + 0,5\cdot p_{m,\bullet}]$.

- Finally, in the proposal by Hasija and Rajendran (2004), $d_j = [1+3u]\cdot\sum_{i=1}^{m} p_{ij}$ with $u \sim U[0,1]$. In our adaptation, the sum of the processing times for each job has been replaced in the same way that in the previous case. As $[1+3u]$ provides values between 1 and 4, this factor multiplied by $C_{max}^{J_O}$ provides values in the interval $[C_{max}^{J_O}, 4\cdot C_{max}^{J_O}]$, being too loose. Therefore, the factor 3 multiplying to the random number u has been replaced by 1, resulting $[1+u]C_{max}^{J_O}$. This distribution is more realistic since it provides values in the interval $[C_{max}^{J_O}, 2\cdot C_{max}^{J_O}]$.

Finally, a new method is proposed based on the idea suggested by Unal et al. (1997) of multiplying the completion time of the job in the revised schedule by a slack factor greater than one, thus providing a time of reaction in the case that unforeseen disruptions in the production process, and to allow rescheduling the jobs on the case of new order arrival. This slack-depending method, labelled SD, is also similar to the one proposed by Della Croce et al. (2000) and to the adaptation of AR, having a uniform distribution

depending on the factor r with $r = 0.2, 0.4, 0.6, 0.8, 1$.

Summarizing, the considered methods are:

- SAR from Sarper (1995): $d \sim U\left[\sum_{j=1}^{n} p_{1j}, \sum_{j=1}^{n}\sum_{i=1}^{m} p_{ij}\right]$
- SRS from Sakuraba et al. (2009): $d = \sum_{j=1}^{n}\sum_{i=1}^{m} p_{ij}$
- AR adapted from Armentano and Ronconi (1999), for $r = 0.6$ and $r = 1.2$: $d \sim U[C_{max}^{J_O}(1 + \frac{r}{4}), C_{max}^{J_O}(1 + \frac{r}{2})]$
- GS adapted from Gelders and Sambandam (1978): $d \sim U[C_{max}^{J_O}, C_{max}^{J_O} + 0,5 \cdot p_{m,\bullet}]$
- HR adapted from Hasija and Rajendran (2004): $d = [1 + u]C_{max}^{J_O}$
- SD for $r = 0.2, 0.4, 0.6, 0.8, 1$: $d \sim U[C_{max}^{J_O}, C_{max}^{J_O}(1 + r)]$

To select the most suitable method to generate realistic common due dates, we compute the relative deviation (in percentage) of the due dates from the makespan of old jobs $C_{max}^{J_O}$. The objective is to determines if a common due date is tight or loose. The relative deviation RD is compute as follows:

$$RD = \frac{duedate - C_{max}^{J_O}}{C_{max}^{J_O}} \cdot 100$$

Then, if this percentage is too small, then the due date is very tight and it will not allow to apply rescheduling. However, if the percentage deviation is too large, then the due date is loose and it would not be realistic.

Since the previously presented methods (except SRS) use the uniform distribution, the RD is computed using the expected due date for these cases. The expected due date of the methods AR, GS, HR and SD depends on $C_{max}^{J_O}$, so in these cases RD is the same regardless $C_{max}^{J_O}$. However, for SRS and SAR we generate common due dates according to these methods and compare them with the makespan value obtained for each problem instance.

Then, we test all methods using Taillard's test-bed (Taillard, 1993) considering all jobs belonging to J_O. Taillard's test-bed consists of 120 instances of various sizes $n \times m$, with

n	m	**SAR**	**SRS**	**GS**	**HR**	**AR**		**SD**				
r						0.6	1.2	0.2	0.4	0.6	0.8	1
20	5	180.52	316.47	1.16	287.95	21.69	43.18	7.71	22.48	27.41	43.86	49.55
	10	196.52	550.42	0.77	287.96	22.91	45.68	7.56	24.90	38.70	41.39	58.17
	20	367.24	788.99	0.57	287.98	21.02	41.44	11.70	25.69	28.05	46.45	49.22
50	5	260.18	356.19	0.50	287.98	21.78	43.37	9.49	13.40	29.71	37.96	56.07
	10	184.11	733.56	0.51	287.98	22.74	47.16	11.02	23.12	33.41	47.69	65.77
	20	728.28	1.238.31	0.46	287.99	20.06	43.45	10.19	16.25	38.60	44.80	40.31
100	5	152.01	376.48	0.28	291.59	23.15	48.32	10.82	20.93	25.12	53.34	66.47
	10	492.59	791.56	0.28	296.99	22.60	44.76	10.18	16.36	24.51	24.74	28.16
	20	429.05	1483.90	0.25	296.99	20.90	41.66	10.12	10.20	30.36	50.52	71.43
200	10	260.42	838.24	0.10	297.00	21.08	46.46	7.70	25.76	25.87	25.96	78.33
	20	241.60	1673.21	0.15	64.20	26.68	41.56	11.77	11.86	11.95	12.06	63.77
500	20	102.92	1795.70	0.07	12.00	17.47	47.41	7.50	27.54	47.59	27.63	12.64
Average		299.62	911.92	0.42	248.88	21.84	44.54	9.65	19.87	30.11	38.03	53.32

Table 3: Comparison between methods for generating common due dates: RD from Taillard's bounds

10 instances for each considered size and $n \in \{20, 50, 100, 200, 500\}$ and $m \in \{5, 10, 20\}$, and processing times uniformly generated in [1,99].

The makespan, C_{max}^{Jo}, is obtained by using the best solution (best-known) from the literature for these problems, i.e. the upper bounds provided by Taillard (2014). In fact, 92 of these 120 upper bounds are known to be optimal, and, for the remaining 28, the average gap between the best known solution and the highest known lower bound is just 0.94% (Vallada et al., 2015).

The results are presented in Table 3, indicating the average RD for each problem size. Additionally, the average RD are shown graphically in Figure 1. As it can be observed, the due dates provided by SAR, SRS and HR are too loose, with very high RD values. Note that the due date obtained by these methods does not depend on the makespan. In comparison with SAR, SRS and HR, the results provided by GS, AR and SD are tighter, implying more realistic due dates.

Figure 2 shows the means and confidence intervals (LSD intervals) for the methods providing tighter results, i.e. GS, AR and SD. In Table 3, as well as in Figure 2, it can be seen that the results for GS are too tight, with values lower than 1% for each size. The

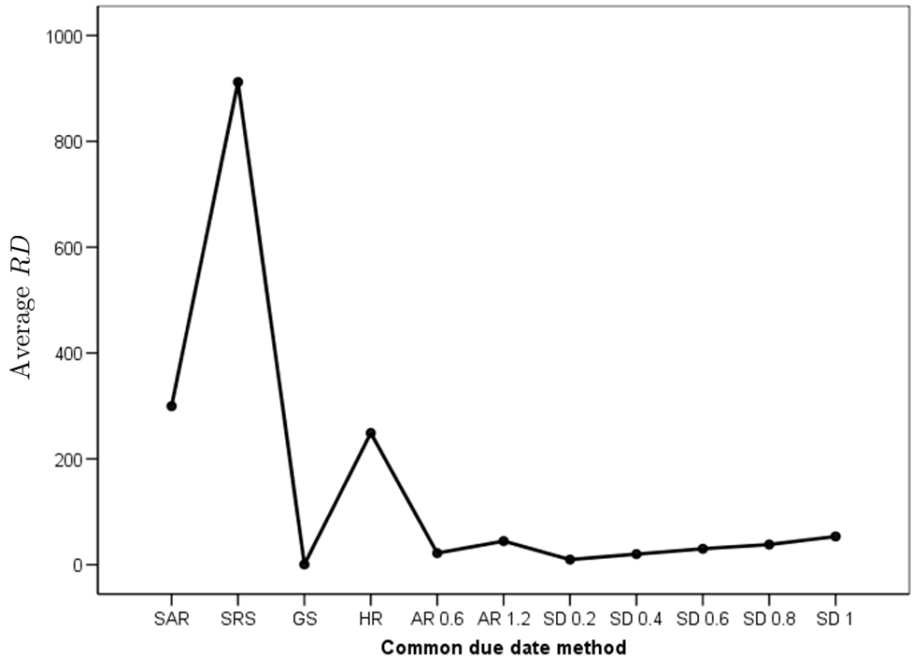

Figure 1: Average RD for common due date methods

common due dates provided by GS do not consider possible disruptions, thus increasing the possibility of violating the due date in case of disruption. Results for AR and SD provide more reasonable percentages for our purpose. They are controlled by the parameter r and depend on the makespan value employed. Between them, the most useful can be the SD method since the values of r give us an upper bound of the slack in a intuitive and easier-to-control manner.

Therefore, in the following section, where we analyse the scenarios identified in Section 1, we build a test bed for which the common due dates for J_O are generated by the SD method. Using this method, we obtain different (tight and loose) common due dates which represents more or less realistic problems for the multi-agent scheduling scenario, which is compared to the classical and availability scheduling scenarios.

4 Analysis of the scenarios

As commented in Section 1, we have identified two main scenarios according to the scheduling policy adopted when two sets of jobs, J_O and J_N compete for the resources: In the first

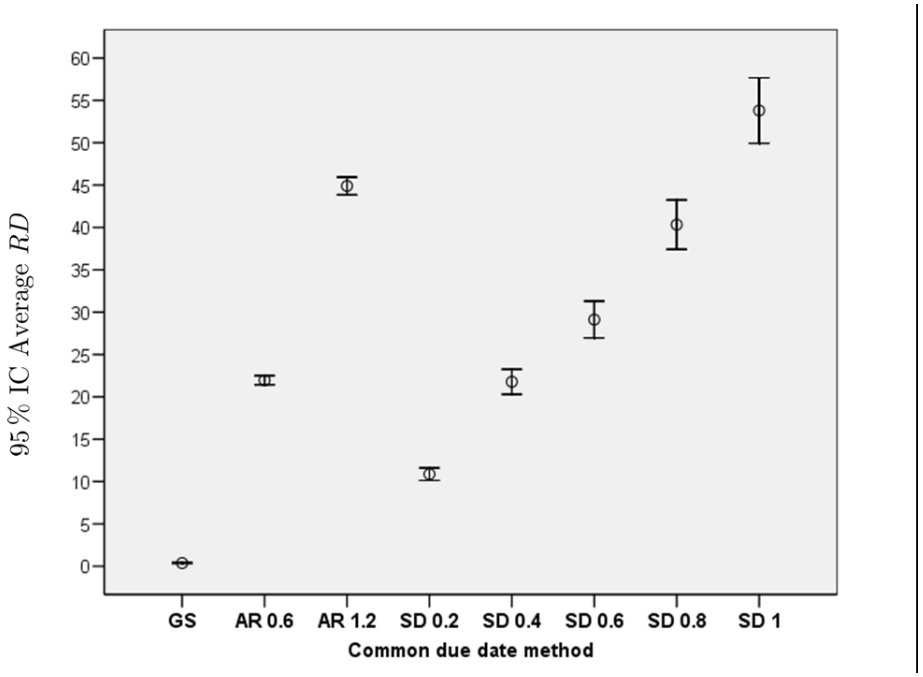

Figure 2: Means and 95% LSD intervals for GS, AR and SD methods

scenario, J_O is considered as frozen, and J_N are scheduled considering a machine availability constraint (ASP). In the second scenario, J_O and J_N are scheduled together, each set with its own objective (MSP). In this Section we compare both policies in order to provide a decision tool according to the slack of the common due date of jobs in J_O, which must not be violated. Additionally, we consider the base case (CSP) where the option is to wait until J_O finishes and J_N is scheduled with the resources completely available, see Table 1 in Section 2. As mentioned earlier, this scenario does not seem realistic, but it provides a reference for the comparison.

It is clear that, for any sequence, the makespan value obtained for the new jobs by the CSP will be greater than or equal to the makespan provided by the ASP. Moreover, any solution provided by the ASP is obviously included in the set of solutions of the MSP, so we can conclude that the multi-agent scheduling scenario dominates the other two scenarios. However, this is not sufficient to state that this is the best scenario: On one hand, we do not know if there are significant differences among the makespan values obtained by this scenario and the others. On the other hand, we do not know the difficulty degree of the problem MSP as compared to CSP and ASP, and this is a key aspect in

order to determine the methods which will be applied to solve it.

Taking into account these observations, the analysis presented in this section has the following objectives:

- To identify the advantages and disadvantages of the three scenarios previously presented in Table 1 in Section 2.

- To check the differences between the problems identified in Table 2 in Section 2 according to the structure of solutions, and to the objective function values. If MSP is similar to CSP or ASP, then existing methods to solve the latter problems could be applied to MSP. Otherwise it would be necessary to develop specific solution methods for MSP.

- To determine the difficulty degree of the three problems. For instance, if MSP is statistically more difficult than CSP and than ASP, then it will be necessary to develop sophisticated algorithms in order to solve it. Otherwise, fast methods and existing constructive heuristics would be sufficient to obtain good solutions for the problem.

In order to answer these questions and to achieve these objectives, we have adopted two approaches:

- A design of experiments, presented in Subsection 4.1, which will allow us to determine the similarity between specific factors of each problem.

- In practice, there are instances of some NP-hard scheduling problems for which is easy to find a good or even the optimal solution, since most solutions yield values close to the optimum and thus even a random solution would be a 'good' solution. An analysis of the structure of the space of solution is presented in Subsection 4.2, by using the concept of "empirical distribution" which allows to study the difficulty of finding a good solution for each problem. The empirical distribution is obtained by considering the frequencies of the values of the objective function for each possible sequence for each instance, in terms of their percentage deviation from the optimal

value. The structures of solutions of CSP and ASP have been analyzed using this concept by Taillard (1990) and by Perez-Gonzalez and Framinan (2009) respectively. To compare the structures of solutions, including the empirical distributions for different problems MSP, we will consider different values of the slack of the common due date of the jobs in J_O as presented in the previous section.

4.1 Design of experiments

The aim of the design of experiments is to analyse the different factors influencing the structure of the space of solutions in problem MSP, solving to optimality different instances of the problem. Therefore, for each instance of the problem, all possible feasible schedules are evaluated, i.e. for all sequences S_J formed by jobs belonging to $J = J_O \bigcup J_N$ verifying that $T_{max}^{J_O}(S_J) = 0$, their makespan is computed so the optimal sequence S_J^* and the optimal makespan $C_{max}^{J_N}(S_J^*)$ of the instance are obtained.

The factors that can affect the structure of the space of solutions are: the number of old jobs (factor n_O), the number of new jobs (factor n_N), the number of machines (factor m) and the slack of the due date with respect to the makespan of the problem (factor r). Regarding the number of jobs and the number of machines, they should be restricted to small values in order to obtain all possible schedules and makespan values in a reasonable time. We will schedule $n = n_O + n_N$ jobs, so n_O and n_N cannot be too large. Therefore, the levels selected are $n_O = \{3, 4, 5\}$, $n_N = \{3, 4, 5\}$, and $m = \{5, 10\}$. Moreover, r controls the width of the interval where the due date will be generated. According to Section 3, the due date will be drawn from the $U[C_{max}^{J_O}, C_{max}^{J_O}(1 + r)]$ distribution, with r the slack of the common due date. Furthermore, for factor r we have selected the levels evaluated in Section 3, i.e.:

- $r = 0$: in this case the due date for J_O is equal to the optimal makespan of this set of jobs, so the old jobs cannot be rescheduled since a change would imply a tardy job. This includes two problems, CSP and ASP depending on the option considered.

- $r \in \{0.2, 0.4, 0.6, 0.8, 1\}$: these levels represent different slacks for the due date with respect to the makespan, so it is possible to reschedule the jobs in J_O, implying

different cases of MSP.

We have developed a full factorial design, which is efficient for evaluating the effects and possible interactions of aforementioned factors (independent variables). An equireplicate design is carried out with 100 runs for each treatment, i.e. 100 problem instances (with the processing times $U[1, 99]$) are generated for each combination. The dependent variable is the optimal makespan obtained after solving each problem instance. Thus we have $3 \times 3 \times 2 \times 7 = 126$ combinations of the levels of all factors with 100 runs, i.e. 12,600 runs in total. Following the convention in most research works, the significance level employed for all statistical tests is 0.05, i.e. there is a 5% chance of rejecting the null hypothesis, even if it is true.

The null hypotheses for the analysis of variance (see e.g. Montgomery, 2005) is H_O: 'There are not differences between the means of the samples'. The two main assumptions to carry out this test (i.e. independency and normality) are verified since each data is the optimal makespan obtained from the resolution of a problem generated independently, and we replicate each treatment with 100 runs, guaranteeing the verification of the central limit theorem. To check the third condition (homoscedasticity) it is necessary to carry out the Levene test. The p-values obtained from Levene test are lower than 0.05 for all cases, except for n_N, so we reject the null hypotheses about homogeneity of variance. Therefore, applying analysis of variance is not suitable and we must consider an alternative non-parametric test.

The Kruskal-Wallis test determines the equality between the levels of the factors. The results indicate that the mean ranks of makespan per run are significantly different among the four factors, since all p-values obtained are lower than 0.05. Moreover, the Mann-Whitney test allows us to study the differences between levels for factors with more than two levels: n_O, n_N and r. For factors n_O and n_N, there are three possible pairs ($3-4$, $3-5$ and $4-5$). Furthermore, the significance must be divided by the number of possible pairs (Bonferroni's correction), i.e. $0.05/3 \simeq 0.016$. In both cases the p-values are lower than 0.016, implying that the problem is different for each level of the number of jobs. Finally, the analysis of the differences between the levels of the factor r implies 21 possible pairs, and all p-values obtained are lower than $0.05/21 \simeq 0.002$, implying differences among all

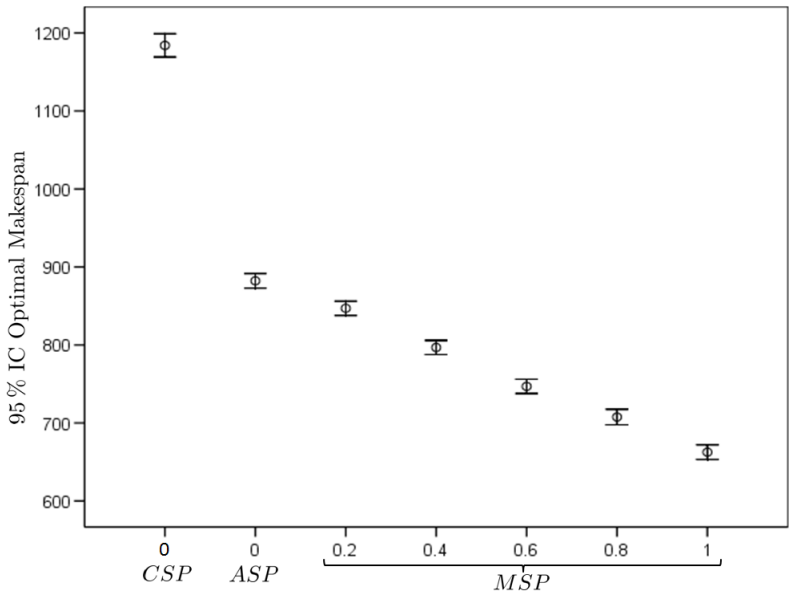

Figure 3: Means and 95% LSD intervals for factor r

levels. Figure 3 shows the means and confidence intervals (LSD intervals) for the levels of the factor r, being the differences between all levels statistically significant.

Table 4 shows the results for each level of r (averaged across all instance sizes). As it can be observed, the optimal makespan for J_N decreases as the value of the factor r increases, i.e. while the slack for the due date for J_O is greater, the makespan for the new jobs decreases, the worst result being obtained for the cases $r = 0$, i.e. CSP and ASP. This is an expected result, since increasing the due date of jobs in J_O allows a higher level of rescheduling, increasing also the number of feasible sequences and the likelihood to find a better solution. Although the design of experiments concludes that there are differences between the levels of each factor, the behaviour is similar for all values of n_O, and the results according to n_N are similar since the estimated marginal means decrease as factor r increases.

As a conclusion, the design of experiments reveals that all factors have influence on the variable (the optimal makespan) and that there are differences between the levels for each factor. The worst results for the optimal makespan are obtained for CSP, which means that the worst option is to wait until jobs in J_O have finished their processing. As mentioned earlier, this was a foreseeable result. To schedule the new jobs taking into

			CSP	*ASP*			*MSP*		
m	n_N	n_O	0.0	0.0	0.2	0.4	0.6	0.8	1.0
5	3	3	770.42	588.14	568.00	539.89	510.48	432.17	399.24
		4	832.41	932.38	897.13	838.68	785.23	739.50	707.92
		5	887.49	649.90	624.88	616.18	591.25	510.45	478.32
	4	3	831.41	997.33	958.37	926.20	873.75	817.31	774.54
		4	889.76	702.38	677.12	669.94	657.02	593.66	553.72
		5	933.96	1056.38	1020.48	994.58	948.56	887.88	838.70
	5	3	886.70	651.71	627.24	581.52	532.92	481.60	439.80
		4	936.19	998.50	943.36	851.43	784.48	766.50	711.39
		5	988.22	706.64	681.11	655.07	620.80	577.73	521.25
10	3	3	1360.63	1062.16	1018.89	943.09	884.57	852.33	785.80
		4	1416.42	753.06	727.83	714.67	693.06	646.26	637.79
		5	1475.72	1124.56	1082.23	1034.38	973.00	941.51	920.24
	4	3	1420.32	713.34	682.01	619.95	550.41	501.12	488.33
		4	1483.48	1059.22	1000.76	847.00	779.42	802.17	735.66
		5	1553.12	760.65	733.41	702.63	649.73	589.97	566.51
	5	3	1479.64	1125.39	1076.06	965.81	891.94	913.78	823.48
		4	1554.72	815.54	789.08	772.94	741.76	686.59	641.50
		5	1612.70	1182.57	1138.48	1068.32	976.39	997.59	902.34
Average			1184.07	882.21	847.02	796.79	746.93	707.67	662.59

Table 4: Optimal makespan values averaged across instance sizes for levels of r

account that the machines are busy (ASP) is not a good option either, being the MSP the option providing the best results. On average, as r increases around 10%, the optimal makespan decreases approximately a 22%. This is an important result that highlights the interest of the multi-agent scheduling scenario, since we can satisfy the due date of old jobs and to achieve a big improvement in the makespan of the new jobs (which implies the ability to set a tight due date for them), even for tight due dates of jobs in J_O.

4.2 Distribution of the space of solutions

The distribution of the space of solutions has been generated for the problems studied in the design of experiment, obtaining all possible makespan values by complete enumeration (i.e. evaluating the $n!$ sequences for a problem with n jobs, checking the feasibility for each schedule, and discarding those unfeasible schedules). It has been applied to 100 problems combining the levels of the factors previously presented. The structure of the space of solutions is given with respect to the optimal solution, i.e. we calculate the relative makespan RM for each feasible solution S_J as follows:

$$RM = \frac{C_{max}^{J_N}(S_J)}{C_{max}^{J_N}(S_J^*)} - 1$$

RM is thus an indicator of the distance of each feasible solution S_J to the optimal solution for each problem instance, S_J^*.

Figure 4 shows the "empirical distribution" for the case $n_O \times n_N \times m = 5 \times 5 \times 10$, including CSP, ASP and MPS for all values of r. The rest of the cases provide similar results. This figure represents the empirical frequencies of RM for the feasible solutions. It can be seen that the ASP is the problem with the solutions closest to the optimal, followed by the case $r = 0.2$ of MSP, then CSP, and finally the rest of the cases of MSP for $r \geq 0.4$. The figure shows the high difficulty degree of the multi-agent scheduling problem in comparison to the other problems, although the feasible solutions of MSP for the special case $r = 0.2$ are closer to the optimal than in the classical problem.

In order to provide more information, and taking into account that Figure 4 considers only feasible solutions, we try to determine the difficulty degree for each problem by the mean of RM and the 95-percentile of RM, indicating the percentage of feasible solutions

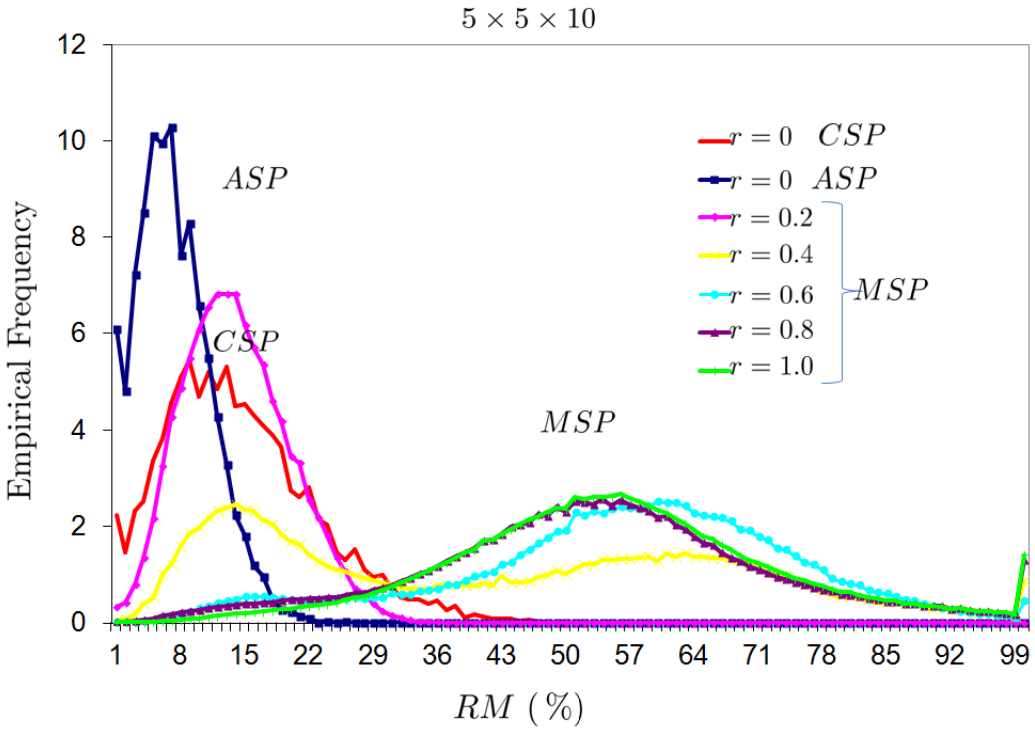

Figure 4: Distribution of feasible solutions for small problems: case $5 \times 5 \times 10$

		$3 \times 3 \times 5$			$4 \times 4 \times 5$			$5 \times 5 \times 10$		
	r	Mean	95%	% feas.	Mean	95%	% feas.	Mean	95%	% feas.
CSP	0	11.76	32	100	14.62	35	100	13.51	28	100
ASP	0	4.49	14	100	6.07	15	100	6.28	13	100
MSP	0.2	12.68	48	4.84	12.63	25	1.15	12.99	22	0.35
	0.4	28.18	88	12.83	30.13	85	5.08	41.04	82	5.28
	0.6	46.99	98	23.76	48.51	98	12.35	55.19	83	14.48
	0.8	53.92	98	59.58	59.73	98	27.72	52.37	85	27.16
	1	56.15	98	78.68	60.99	98	52.32	54.27	85	52.56

Table 5: Mean and percentile 95 of *RM* and percentage of feasible solutions

for each problem (number of feasible solutions evaluated by complete enumeration divided by $n!$). Table 5 shows the degree of difficulty for some cases combining $n_O \times n_N \times m$, particularly the cases $3 \times 3 \times 5$, $4 \times 4 \times 5$ and $5 \times 5 \times 10$ representing small, medium and large problem sizes. It can be observed that the percentages of feasible solutions for CSP and ASP are 100%, while there are different percentages for MSP according to the value of r (since MSP is a constrained problem). Results show that the easiest problem is ASP, where the 95-percentile is close to 15 in all cases, and the mean is around to 6. For CSP, the values of the mean are less than 15 while the values of 95-percentile are around 30. The case $r = 0.2$ of MSP shows that the 95-percentile is lower than 25 for the cases $4 \times 4 \times 5$ and $5 \times 5 \times 10$, being worst (i.e. more difficult) for the smaller case. Moreover, the values of the mean for $r = 0.2$ are around 12, being this case easier than CPS. However, the percentage of feasible solutions is very small for $r = 0.2$, and decreases with the size of the problem. The difficulty degree increases with r in MSP according to the values of the mean and 95-percentiles, and, although the percentage of feasible solutions increases with r, it is lower than 55% even for the bigger sizes and $r = 1$.

Taking into account the results from Subsection 4.1, CSP and ASP do not provide the best values for the makespan of J_N, but the difficulty of the problem is, in general, lower than MSP according to the distribution of feasible solutions. However, for MSP we can state that we obtain better values of the optimal makespan while r increases, but the difficulty degree of the problem increases too, according to the percentages of feasible solutions, and their distances to the optimal solution. Therefore, the main issue in this problem is not only to find solutions close to the optimal, but to find any feasible solution.

5 Conclusions

This paper compares different scenarios depending on the scheduling policy considered when two sets of jobs (old and new jobs) are competing for the same resources in a permutation flowshop with different objectives. Old jobs have a common due date which must not be violated, and the makespan of the new jobs must be minimized. On one hand, the first policy consists on 'freezing' the set of old jobs, which leads to solving an

availability scheduling problem (ASP). Another policy is to schedule together old and new jobs, which implies solving a multi-agent scheduling problem (MSP). Finally, the policy of waiting until the old jobs have finished their processing, and machines are available –although it is not realistic– is also considered as a base case (CSP).

In order to compare the scenarios for a permutation flowshop problem with a common due date, we study the differences and structure of the solution space of ASP, MSP and CSP from the viewpoint of order management, i.e. guaranteeing delivery reliability and speed of the orders. We consider makespan minimization as the criteria for the new jobs, whereas the old jobs cannot violate their tardiness with respect to the common due date.

To carry out the experiments, we first analyze different methods to generate common due dates of the old jobs, as it is a key parameter in our experiments since the number of feasible schedules of the multi-agent problem depends on the slack of the due date. Thus, an analysis of the existing methods in the literature for setting a common due date, including a new method, has been carried out. Results show that the new method allows an easy manner of generate common due date for the old jobs with different slacks.

Then, in order to compare the scenarios previously proposed, a design of experiment has been carried out to check the influence of a number of factors on the structure of solutions of the problems. The number of machines and jobs have been considered to control the size of the problems. Moreover, the slack of the common due date of old jobs allows to distinguish different cases of the multi-agent scheduling problem MSP. From the analysis, we observe that the optimal makespan of the new jobs is better for the MSP than for CSP and for ASP. In addition, together with the size of the problem, the slack of the due date with respect to the makespan of the old jobs has a great influence on the structure of solutions of MSP, since as the slack factor increases around 10%, the optimal makespan of the new jobs decreases by more than 20% for all cases.

Furthermore, the distribution of the space of solutions gives us an important result about the difficulty degree of the MSP for different values of factor r, as compared to the ASP and the CSP. The feasible solutions for the MPS with $r = 0.2$ are closer to the optimal than the CSP for the larger sizes. However, the number of feasible solutions is too small, so in this case the difficulty may be to find these feasible solutions. Moreover, for

the rest of values of r, the percentage of feasible solutions for MSP increases as expected, but it is lower than 55% even for $r = 1$, and the distances to the optimal solutions increases making the problem more difficult.

In summary, considering the multi-agent scheduling scenario provides good makespan values of the new jobs when the slack of common due date for the old jobs is medium/high. However, this scenario implies a longer due date for the old jobs, reflecting a lower service level (Birman and Mosheiov, 2004). When the slack is low, perhaps the multi-agent approach does not pay off, since the improvement of the makespan for the new jobs may not compensate the difficulty of the problem to find feasible solutions, being the ASP approach more appropriated according to the good quality of most schedules obtained when solving this problem. However, this would change if we had a solution procedure able to find feasible solutions. This makes the multi-agent scenario to be the best policy, also taking into account that any feasible solution found by the method is close to the optimal.

The problem addressed in this paper presents some interesting implications to both practitioners and researchers. For the former, it gives some evidence to support the advantages of 'freezing' the schedules of existing jobs in the shop floor, a commonly used practice that finds justification not only from a managerial viewpoint (simplicity, minimization of disruptions, low nervousness, ...), but also from a performance viewpoint. For researchers, the challenge posed by the structure of the solution space of multi-agent scheduling problems may foster the investigations towards more accurate/near optimal methods, since substantial performance improvements can be found if these methods are available.

Finally, a future research line is to relax the MSP problem, allowing some tardiness for the old jobs, i.e. considering the problem $Fm|prmu|\epsilon(C^{J_N}_{max}/T^{J_O}_{max})$ and to perform the analysis for different values of ϵ.

Acknowledgements

The authors are sincerely grateful to the anonymous referees. This research has been funded by the Spanish Ministry of Science and Innovation, under the project "SCORE" with reference DPI2010-15573/DPI.

References

Agnetis, A., Billaut, J. C., Gawiejnowicz, S., Pacciarelli, D., and Soukhal, A. (2014). *Multiagent Scheduling: Models and Algorithms*. Springer.

Akkan, C. (1997). Finite-capacity scheduling-based planning for revenue-based capacity management. *European Journal of Operational Research*, 100(1):170–179.

Armentano, V. A. and Ronconi, D. P. (1999). Tabu search for total tardiness minimization in flowshop scheduling problems. *Computers & Operations Research*, 26(3):219–235.

Birman, M. and Mosheiov, G. (2004). A note on a due-date assignment on a two-machine flow-shop. *Computers & Operations Research*, 31(3):473–480.

Blazewicz, J., Pesch, E., Sterna, M., and Werner, F. (2004). Flow shop scheduling with late work criterion - choosing the best solution strategy. In *Applied Computing. Second Asian Applied Computing Conference, AACC 2004, Kathmandu, Nepal, October 29-31, 2004. Proceedings*, Lecture Notes in Computer Science, book chapter Volume 3285/2004, pages 68–75. Springer Berlin / Heidelberg.

Blazewicz, J., Pesch, E., Sterna, M., and Werner, F. (2008). Metaheuristic approaches for the two-machine flow-shop problem with weighted late work criterion and common due date. *Computers & Operations Research*, 35(2):574–599.

Della Croce, F., Gupta, J. N. D., and Tadei, R. (2000). Minimizing tardy jobs in a flowshop with common due date. *European Journal of Operational Research*, 120(2):375–381.

Fernandez-Viagas, V. and Framinan, J. M. (2014). On insertion tie-breaking rules in heuristics for the permutation flowshop scheduling problem. *Computers & Operations Research*, 45(0):60–67.

Framinan, J. M. and Leisten, R. (2008). Total tardiness minimisation in permutations flow shops: a simple approach based on a variable greedy algorithm. *International Journal of Production Research*, 46(22):6479–6498.

Framinan, J. M. and Leisten, R. (2010). Available-to-promise (ATP) systems: a classification and framework for analysis. *International Journal of Production Research*, 48(11):3079–3103.

Frederix, F. (2001). An extended enterprise planning methodology for the discrete manufacturing industry. *European Journal of Operational Research*, 129(2):317–325.

Garey, M. R., Johnson, D. S., and Sethi, R. (1976). The complexity of flowshop and jobshop scheduling. *Mathematics of Operations Research*, 1(2):117–129.

Gelders, L. F. and Sambandam, N. (1978). Four simple heuristics for scheduling a flowshop. *International Journal of Production Research*, 16(3):221–231.

Graham, R., Lawler, E. L., Lenstra, J., and Rinnooy Kan, A. H. G. (1979). Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326.

Gupta, J. N. D. and Stafford, E. F. (2006). Flowshop scheduling research after five decades. *European Journal of Operational Research*, 169(3):699–711.

Hasija, S. and Rajendran, C. (2004). Scheduling in flowshops to minimize total tardiness of jobs. *International Journal of Production Research*, 42(11):2289–2301.

Huynh-Tuong, N. and Soukhal, A. (2009). Interfering job set scheduling on two-operation three-machine flowshop. 2009 IEEE-RIVF International Conference on Computing and Communication Technologies: Research, Innovation and Vision for the Future, RIVF 2009.

Johnson, S. M. (1954). Optimal two-stages and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1:61–67.

Khelifati, S. L. and Bouzid-Sitayeb, F. (2011a). A multi-agent scheduling approach for the joint scheduling of jobs and maintenance operations in the flow shop sequencing problem. *Lecture Notes in Computer Science*, 6923 LNAI(PART 2):60–69.

Khelifati, S. L. and Bouzid-Sitayeb, F. (2011b). A sequantial distributed approach for the joint scheduling of jobs and maintenance operations in the flowshop sequencing problem. In *International Conference on Intelligent Systems Design and Applications, ISDA*, pages 420–425.

Lee, C. Y. (1997). Minimizing the makespan in the two-machine flowshop scheduling problem with an availability constraint. *Operations Research Letters*, 20(3):129–139.

Lee, W. C., Chen, S. K., Chen, C. W., and Wu, C. C. (2011). A two-machine flowshop problem with two agents. *Computers and Operations Research*, 38(1):98–104.

Luo, W., Chen, L., and Zhang, G. (2011). Approximation schemes for two-machine flow shop scheduling with two agents. *Journal of Combinatorial Optimization*, 24(3):229–239.

Montgomery, D. C. (2005). *Design and analysis of experiments*. John Wiley and Sons, New York (USA), 6th edition.

Mor, B. and Mosheiov, G. (2014). Polynomial time solutions for scheduling problems on a proportionate flowshop with two competing agents. *Journal of the Operational Research Society*, 65(1):151–157.

Nagano, M. S., Ruiz, R., and Nogueira Lorena, L. A. (2008). A constructive genetic algorithm for permutation flowshop scheduling. *Computers & Industrial Engineering*, 55(1):195–207.

Perez-Gonzalez, P. and Framinan, J. M. (2009). Scheduling permutation flowshops with initial availability constraint: Analysis of solutions and constructive heuristics. *Computers & Operations Research*, 36(10):2866–2876.

Perez-Gonzalez, P. and Framinan, J. M. (2014). A common framework and taxonomy for multicriteria scheduling problems with interfering and competing jobs: Multi-agent scheduling problems. *European Journal of Operational Research*, 235(1):1–16.

Ribas, I., Companys, R., and Tort-Martorell, X. (2010). Comparing three-step heuristics for the permutation flow shop problem. 37(12):2062–2070.

Sakuraba, S., Ronconi, D. P., and Sourd, F. (2009). Scheduling in a two-machine flowshop for the minimization of the mean absolute deviation from a common due date. *Computers & Operations Research*, 36(1):60–72.

Sarper, H. (1995). Minimizing the sum of absolute deviations about a common due date for the two-machine flow shop problem. *Applied Mathematical Modelling*, 19(3):153–161.

Taillard, E. D. (1990). Some efficient heuristic methods for the flow shop sequencing problem. *European Journal of Operational Research*, 47(1):65–74.

Taillard, E. D. (1993). Benchmarks for basic scheduling problems. *European Journal of Operational Research*, 64(2):278–285.

Taillard, E. D. (2014). Scheduling instances: Summary of best known lower and upper bounds of taillard's instances. Available from http://mistic.heig-vd.ch/taillard.

T'kindt, V. and Billaut, J. C. (2002). *Multicriteria scheduling: Theory, models and algorithms*. Springer, Berlin (Germany), second edition.

Tzeng, Y. R. and Chen, C. L. (2012). A hybrid eda with acs for solving permutation flow shop scheduling. 60(9-12):1139–1147.

Unal, A. T., Uzsoy, R., and Kiran, A. S. (1997). Rescheduling on a single machine with part-type dependent setup times and deadlines. *Annals of Operations Research*, 70:93–113.

Vallada, E., Ruiz, R., and Framinan, J. M. (2015). New hard benchmark for flowshop scheduling problems minimising makespan. *European Journal of Operational Research*, 240(3):666–677.

Vig, M. M. and Dooley, K. J. (1991). Dynamic rules for due-date assignment. *International Journal of Production Research*, 29(7):1361.

Xu, X. Q. and Lei, D. M. (2014). A parallel iterated local search for two-agent flow shop scheduling. *Advanced Materials Research*, 926-930:3476–3484.