Scheduling flowshops with initial availability constraint: analysis of solutions and constructive heuristics [*]

Paz Perez Gonzalez[†] Jose M. Framinan

Departamento de Organización Industrial y Gestión de Empresas, Escuela Superior de Ingenieros, University of Seville. Camino de los descubrimientos s/n. 41092 Seville, Spain

Abstract

In this paper, we address the problem of scheduling a set of jobs in a flowshop with makespan objective. In contrast to the usual assumption of machine availability presented in most research, we consider that machines may not be available at the beginning of the planning period, due to processing of previously scheduled jobs. We first formulate the problem, analyse the structure of solutions depending on a number of factors (such as machines, jobs, structure of the processing times, availability vectors, etc.), and compare it with its classical counterpart. Results indicate that the problem under consideration presents a different structure of solutions, and that it is easier than the classical permutation flowshop problem. In view of these results, we propose and test a number of fast heuristics for the problem.

1 Introduction

Scheduling literature usually assumes that machines are available during the whole planning horizon [4]. However, there is a wide range of realistic situations where machines may not

[*]This is an Accepted Manuscript of an article published by Elsevier in Computers and Operations Research Volume 36, Issue 10, October 2009, Pages 2866-2876, available online: http://dx.doi.org/10.1016/j.cor.2008.12.018

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

be completely available, such as breakdowns (stochastic unavailability) [5], or if preventive maintenance activities are necessary (deterministic unavailability) [28]. In this paper, we tackle a different type of machine unavailability, in which we assume that the machines may not be immediately available for processing the set of jobs to be scheduled, but only from a date a_i that we denote as *availability instant*. This problem corresponds to a common real-life scenario where jobs must be scheduled in a periodical, dynamic 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. As a result, jobs scheduled in the previous period may not be completed at this point. These jobs either have started their processing, or wait until they can start according to the previous schedule. In the first case, the jobs cause the machines not to be available from the beginning of the planning period. We denote this type of jobs as 'frozen' jobs, since their schedule remains the same. In the second case, these jobs can be merged with the new set and rescheduled.

Among the different shop floor layouts, we focus on the flowshop, which is the most popular setting both in practice and research (some recent references are e.g. [12, 29]). The flowshop layout 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, but we assume that the processing sequence of the jobs is the same for all the machines (i.e. permutation flowshop) and hence only $(n!)$ schedules have to be considered. A representation of the problem studied in the scenario described previously by frozen jobs is displayed in Figure 1.

******* Insert Figure 1 about here *********

Regarding the different scheduling objectives that can be sought, we consider the minimization of the makespan or maximum completion time. Aside to the maximization of the throughput of the shop, makespan minimisation is connected to the problem of setting a common due date for the incoming jobs. If we consider that the set of jobs constitutes a single order, or a batch to be processed together in a subsequent operation, then the optimal makespan value serves to set the tightest due date for the order (batch). Note that, if each job belongs to a different order, then a suitable criterion for setting tight due dates would be the minimisation of the sum of the flow times.

Therefore, our problem consists on scheduling jobs in a flowshop where machines are not initially available with the objective of minimising makespan. This problem (denoted as P_A in the following) is related to two classes of scheduling problems:

- Scheduling with non availability of the machines, and
- 'Classical' flowshop scheduling without machine availability constraints and makespan objective (denoted as P_O in the following).

Machine scheduling problems with availability constraints have received increasing attention from researchers in the last decade [36], but most references do not deal with flowshops (see e.g. [21, 22, 31, 32, 35] for some references on the problem for different settings). Most of the works on flowshops are restricted to the 2-machine case (see [5, 7, 8, 10, 11, 24, 36, 37]), and only [2–4, 28] consider the case for more than 3 machines. Aside, in these works, availability constraints are represented by windows or holes, i.e. given time intervals for which each machine may be unavailable for processing jobs. These intervals are situated in any place on the planning horizon, which is not the case for the P_A problem. Only Lee [20] considers the 2-machine flowshop problem taking into account the special case where the unavailability happens due to unfinished jobs that were scheduled in the previous planning period. He shows that the problem is solvable in polynomial time.

On the other hand, the well known permutation flowshop problem with the makespan criterion (P_O) can be considered as a special case of P_A. P_O has been intensively addressed during the last 40 years [12] and it is known to be solvable to optimality in polynomial time for $m = 2$ by Jonhson's algorithm [18] and NP-complete in the strong sense when $m > 2$ [15]. For this reason, there is a number of heuristic procedures providing good approximate solutions to the problem [12, 29].

Given the similarity among both problems and the abundance of solution procedures for P_O, it is worth to investigate under which circumstances they differ. This analysis would give clues to devise solution procedures P_A, based (or not) on solution procedures adapted from problem P_O.

The rest of the paper is organized as follows: In Section 2 we present the notation employed, and analyze the structure of P_A by studying the main factors that may have influence on it, and by establishing its relation with P_O. This section is the basis of Section 3, where we propose and evaluate several heuristics that take into account the obtained results of previous section. Finally, we present the conclusions of the work and the future research lines.

2 Analysis of the problem

The following notation is adopted in the problem under consideration:

n : number of jobs

m : number of machines

j : job index ($j = 1, \dots, n$)

i : machine index ($i = 1, \dots, m$)

p_{ij} : processing time of job j in machine i

a_i : instant which define the time from machine i is available, $a_i \geq 0$, $\forall i = 1, \dots, m$

Without loss of generality we can assume that $a_i \leq a_{i+1}$, $\forall i = 1, \dots, m$. If it exists some i with $a_i \geq a_{i+1}$, then a_{i+1} does not have influence in the problem (see Figure 2). In fact, each a_{i+1} has influence if it is greater than $a_i + min_j\{p_{ij}\}$ for $i = 1, \dots, m$. In addition we can suppose that $a_1 = 0$. In other case, we can do a reference change $a'_i = a_i - a_1$, for all $i = 1, \dots, m$. Let $a = [a_1, a_2, \dots, a_m]$ the availability vector.

******* Insert Figure 2 about here *********

Scheduling problems are usually represented by the three-parameter notation $\alpha|\beta|\gamma$ introduced by Graham et al. [16]. According to this notation, problem P_O is denoted $Fm|prmu|C_{max}$, where Fm indicates that it is a flowshop problem with m machines, $prmu$ implies that it is a permutation case, and finally, C_{max} is the objective to minimize, i.e the makespan criterion. Following this notation, we denote problem P_A as $Fm|prmu, a_i|C_{max}$.

Johnson's rule is optimal for the P_A problem when $m = 2$ [20]. However, it can be shown that the $Fm|prmu, a_i|C_{max}$ problem is NP-hard in the strong sense for $m > 2$. The proof is based on the fact that $F3||C_{max}$ is strongly NP-hard [15, 26], since this complexity proof can be applied to P_O with $m > 2$.

As the differences between P_O and P_A are given by the availability instants a_i, it is worth investigating under which values of a_i are both problems different and, if so, which form takes the structure of solutions of P_A. This information would be very useful in order to develop approximate algorithms for the problem. In order to answer these questions we employ two approaches: first a design of experiments (which will allow us to determine the similarity between characteristic factors of each problem), and secondly an analysis of the structure of the space of solutions (which will allow us to analyse the complexity of P_A in order to develop efficient solution procedures for the problem). Both approaches are described in the next subsections.

2.1 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 P_A. To carry out this analysis, we first identify a number of factors that can affect the structure of the space of solutions. Then, for each combination of these factors, we build a high number of problem instances and obtain all possible schedules and the corresponding solution values. These results are then summarised in order to extract conclusions on the distribution of the space of solutions. This approach is similar to the one employed by [33] and by [6] in their analysis of the $F_m|prmu|C_{max}$ and $F_m|prmu|\Sigma T_j$ problems respectively (with ΣT_j the sum of the flow times).

The factors considered in the design of experiments are the following: the number of jobs (factor N), the number of machines (factor M), the processing times (factor P), and the values of a_i (factor K).

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 results in a reasonable time. Therefore, levels of N and M are $n \in \{5, 10\}$ and $m \in \{5, 10\}$ respectively.

Furthermore we consider three different ways to generate processing times. Levels p of the factor P can be:

- Random instances ($p = R$). In this level, processing times are generated according to a uniform distribution. This is the most popular distribution to generate processing times in the literature (see e.g. [13, 25, 29, 34]), as it is known to provide most difficult problem instances, although it is hardly found in practice [38]. In our experiments, we consider a uniform distribution between 1 and 99, which is widely used in the literature.

- Structured instances ($p = S$). The so-called structured instances (i.e. with correlation of processing times across jobs and/or machines) are considered to be better representative of realistic flowshops [38]. Watson et al. [38] present three types of structured problems: job-correlated (n distributions, one for each job); machine-correlated (m distributions, one for each machine); and mixed-correlated (are similar to machine-correlated problems, with the exception that the relative ranks of jobs processing times are generally consistent across the machines, for example, if a job has the largest processing time on machine 1, then it is likely to have the largest processing time on machines 2 through m). To control the expected problem correlation the distribution means are randomly sampling from a fraction α of the processing times interval (by varying α from 0 to 1 we can gradually select from random to more structured problems). In our experiment, we have generated structured instances by the generator provided at [1], selecting mixed-correlated problems with processing times in the interval between 1 and 99 and $\alpha = 0.5$, which corresponds to an intermediate level of correlation across jobs and machines.

- Instances with missing operations ($p = M$). The existence of missing operations is known to be another feature of real-life environments [27]. In this type of problem instances, each job has a percentage of operations whose processing times are equal to zero, while the rest of the processing times are drawn from a random distribution [9, 17, 27]. In our experiments, we select a 10% of missing operations and the usual random interval between 1 and 99.

Finally, we define a factor K to control the availability vector size. We calculate $C_i(S_{ini})$ the completion time of sequence $S_{ini} = [1, \ldots, n]$, verifying that $C_i(S_{ini}) < C_j(S_{ini})$ for all $i < j = 1, \ldots, m$, so the initial vector can be calculated from these values doing the following reference change defined as $\overline{a_j} = C_j(S_{ini}) - C_1(S_{ini})$ for all $j = 1, \ldots, m$, i.e $\overline{a} = [\overline{a_1}, \ldots, \overline{a_m}] = [0, C_2(S_{ini}) - C_1(S_{ini}), \ldots, C_m(S_{ini}) - C_1(S_{ini})]$. For different values of $k \in K$ we consider the problem with the availability vector $a = k * (\overline{a})$. Selected values are $k = 0$ (implying problem without availability constraint, i.e. P_O) and $k = 0.5$, $k = 1$ and $k = 2$ (implying some P_A cases with different sizes of a).

Taking into account the aforementioned factors, we have developed a full factorial design which is efficient for evaluating the effects and possible interactions of several factors (independent variables). An equireplicate design is carried out with 100 runs for each treatment, i.e. 100 problem instances are generated for each combination. Thus we have $2\times 2\times 3\times 4 = 48$ combinations of the levels of all factors with 100 runs, i.e. 4800 runs in total. Table 1 summarizes the considered factors, levels and runs per treatment.

******* Insert Table 1 about here *********

The convention in most research is to use a significance level of 0.05, i.e. there is a 5 percent chance of rejecting the null hypothesis, even if it is true. So we employ that significance level for all statistical tests developed.

In order to check the similarity of P_O and P_A depending on the considered factors, we initially intended to perform an Analysis of Variance [23], with the following null hypotheses H_O: 'There are not differences between the means of the samples'. The main assumptions to carry out this test are: independency (the data points must be independent from each other), normality (the distributions must be normal), and homoscedasticity (the variances of the samples must not be different).

The first condition is verified since each data is the optimal makespan obtained from the resolution of an independent generated problem. As we replicate each treatment with 100 runs, the Central Limit Theorem states that, as sample size increases, the sampling distribution of sample means approaches to that of a normal distribution, so the second condition is verified too. To check the last condition, it is necessary to carry out the Levene test. The p-values obtained from Levene test are lower than 0.05 for all cases, so we reject the null hypotheses about homogeneity of variance.

Therefore, applying Analysis of Variance is not suitable, and we must consider an alternative nonparametric test, i.e. the Kruskal-Wallis test [23], which 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 (p-values are equal to 0.000 for all cases). Mann-Whitney test [23] is another non-parametric test which allows us to study differences between levels for factors P and K. In the case of factor P, there are three possible pairs ($R-S$, $R-M$ and $S-M$). In this study the significance must be divided by the number of possible pairs (Bonferroni's correction), i.e. 0.05/3. The comparison between $p = R$ and $p = S$ (i.e. random and structured generation of processing times of factor P) gives a p-value equal to 0.180, which indicates that there are not significative differences between them. However the results indicate rejection of the hypothesis of similarity between the rest of the levels (p-values lower than 0.05/3). As results are dependent on factor P, the analysis of factor K is developed for each level of P. The p-values obtained from each Kruskal-Wallis tests are lower than 0.05, so Mann-Whitney tests are carried out to determine the differences between levels of K for each case of P. Results for both factors are represented in Table 2, showing the means for each case and grouping in the same set those without significative differences.

******* Insert Table 2 about here *********

The similarity between the optimal makespan values of random and structured problems is shown in Table 2, and the similarity of results in theses cases for $k = 0$ and $k = 0.5$. Figure 3 shows these conclusions, both figures, a) and b) present the same result in two ways. In Figure 3 a) it can be observed that marginal means increase while factor K increases for all problem types. Figure 3 b) reveals that random and structured problems are more similar when $k = 0$ and $k = 0.5$.

******* Insert Figure 3 about here *********

By performing a sensitivity analysis for different values of factor K ($0 \leq k \leq 1$) in the cases of structured and random problems, we can conclude that the problems are different when $k > 0.5$ for both cases.

In summary, the analysis carried out indicates the difference between problems P_O and P_A even for moderate values of k. If we recall that $k = 1$ would represent the 'typical' non availability of machines after processing a set of jobs identical to the one that is to be scheduled, we can conclude that problem P_O would be different from P_A for most dynamic environments, at least with respect to the makespan values. Now the issue is to establish whether these differences in makespan values are approximately the same for all schedules due to the non availability of machines (and therefore the wealth of solution procedures available for P_O could be applied in an straightforward manner to P_A), or the non availability of machines induces a different structure of the solution space with respect to that of P_O. This analysis is performed in the next subsection.

2.2 Distribution of the space of solutions

In Figure 4 we show the empirical distribution for the three levels of factor P, representing all possible makespan values obtained by complete enumeration of 100 problems with 10 jobs and 10 machines for all levels of factor K. The distribution of the space of solution is given relatively to the optimal solution, i.e. we calculate the relative makespan $C^r_{max}(S)$ to obtain the approximation percentage to the optimal solution of each makespan value.

$$C^r_{max}(S) = \frac{C_{max}(S)}{C_{max}*} - 1$$

******* Insert Figure 4 about here *********

From Figure 4 it can be observed that while k is increasing, the frequency of solutions with a small approximation percentage is larger, that is, the problem becomes easier (in statistical terms) as k increases, being P_O the most difficult problem for all problem types. This pattern is consistent for all levels of factor P (random, structured, and missing-operations). Structured problems are the easiest type of problems, because a high percentage of solutions are found at less than 5% above the optimal. In particular, when $k = 2$ almost a hundred of them (99.6%) have an approximation percentage to the optimal makespan less than 2%. Finally, it can be noted that the distribution of solutions in random and missing operations cases are very similar.

The complete results are presented in Table 3, which shows the mean of the approximation percentage to the optimal solution for each problem, and the upper bound of approximation percentage to the optimal for 95% of solutions. These results are classified by the generation of processing times factor, P, for all values of K and for all combinations of number of jobs and number of machines (NxM). For example, in random problems with 5 jobs and 5 machines in the case of $k = 2$ the mean of approximation percentage to the optimal solution is 2.51, i.e. any solution is (on average) below 2.51% of the optimal makespan. Furthermore, the 95% of solutions in this case are below 12% of the optimal makespan. However, the mean for $k = 0$ is 16.57 and the upper bound for the 95% of solutions is 39%, showing in a clear way that problems are easier while k increases. As we conclude previously from Figure 4, this table shows that results for missing operations problems are similar than random, although the former are slightly more difficult than the latter. A remarkable issue are the results for structured problems. As shown in Figure 4 and confirmed in Table 3, this type of problems is easier to solve, with an approximation percentage less than 6% for almost all solutions (95%), decreasing while k increases until less than 1% for $k = 2$ when NxM are equal to 5x10 and 10x10.

******* Insert Table 3 about here *********

The results of the analysis raise an important conclusion of practical application: since for most real-life environments scheduling is performed on a periodical basis and this would naturally lead to unavailability of machines at the starting of the scheduling period, this scheduling decision problem becomes easier than its 'classical' (i.e. without machine unavailability) counterpart. Besides, if the processing times of the jobs are correlated (which is the situation in most real-life environments), the problem becomes even easier. Also note that k is somehow related to the congestion of the shop floor, so this indicates that the highest the workload of the shop, the easier is to find a good solution for the problem. By no means these results indicate that scheduling is not relevant for P_A, as there are notable differences among the best and worst makespan values. In addition, we have no way to carry out such analysis for problems of realistic size, as the computation time required to evaluate all possible schedules is simply unacceptable. Another consequence of the results is that the simplicity of problem P_A suggests that fast heuristics could be very effective, as they may provide very good solutions in nearly real-time. Recall that, since our interest is on due date setting, the possibility of providing the customer with an accurate due date in real time is a highly desirable feature. Therefore, our efforts would concentrate in building very fast heuristics for the problem. We present this outcome in the next section.

3 Heuristics

In order to develop fast heuristics for the problem, we adopt the framework presented by Framinan et al. [12]. This framework classifies heuristics depending on the applied phases [12]: In the first phase (phase I) named index development, jobs are arranged according to a certain indicator based on the data of the problem instance. In the second phase or phase II (solution construction), the solution is constructed in a recursive manner trying one or more unscheduled jobs to be inserted in one or more positions of a partial schedule until the schedule is completed. The last phase III (solution improvement) improves the solution from phase II by means of a local search procedure, so the quality of the new solution is equal to or better than that of phase II.

Next subsection presents a set of heuristics collectively denoted as *Initial Sequence Heuristics*. These have been obtained by applying the first phase once or twice. In this first phase, we sort the jobs according to a tuple (p_j, C) with p_j the sorting indicator and C the sorting criterion. According to this notation, the tuple ($\Sigma_{i=1}^{m} p_{ij}$,INCR) denotes a heuristic solution given by sorting the jobs in ascending (increasing) order of the sum of their processing times. We have considered 15 indicators, which are the following:

(1) SUM: total processing times of the jobs

$$p_j = \sum_{i=1}^{m} p_{ij} \tag{1}$$

(2) WSUM: weighted total processing times of the jobs

$$p_j = \sum_{i=1}^{m} (m-i) * p_{ij} \tag{2}$$

(3) SQ: sum of the squared residuals

$$p_j = \sum_{i=1}^{m-1} (r_{ij})^2 \tag{3}$$

(4) WSQ: weighted sum of the squared residuals

$$p_j = \sum_{i=1}^{m-1} (m-i) * (r_{ij})^2 \tag{4}$$

(5) ABS: sum of the absolute residuals

$$p_j = \sum_{i=1}^{m-1} |r_{ij}| \tag{5}$$

(6) WABS: weighted sum of the absolute residuals

$$p_j = \sum_{i=1}^{m-1} (m-i) * |r_{ij}| \tag{6}$$

(7) FMACH ABS: First machine residual absolute value

$$p_j = |r_{1j}| \tag{7}$$

(8) SQ RR: sum of the squared real residuals

$$p_j = \sum_{i=2}^{m-1} (r^*_{ij})^2 \tag{8}$$

(9) WSQ RR: weighted sum of the squared real residuals

$$p_j = \sum_{i=2}^{m-1} (m-i) * (r^*_{ij})^2 \tag{9}$$

(10) ABS RR: sum of the absolute real residuals

$$p_j = \sum_{i=2}^{m-1} |r^*_{ij}| \tag{10}$$

(11) WABS RR: weighted sum of the absolute real residuals

$$p_j = \sum_{i=2}^{m-1} (m-i) * |r^*_{ij}| \tag{11}$$

(12) NEG RR: sum of the negative real residuals

$$p_j = \sum_{i=2}^{m-1} |min(r^*_{ij}, 0)| \tag{12}$$

(13) WNEG RR: weighted sum of the negative real residuals

$$p_j = \sum_{i=2}^{m-1} (m-i) * |min(r^*_{ij}, 0)| \tag{13}$$

(14) POS2NEG: sum of the positive real residuals with the negative real residuals multiplied by 2.

$$p_j = \sum_{i=2}^{m-1} (max(r^*_{ij}, 0) + 2 * |min(r^*_{ij}, 0)|) \tag{14}$$

(15) WPOS2NEG: weighted sum of the positive real residuals with the negative real residuals multiplied by 2.

$$p_j = \sum_{i=2}^{m-1} (m-i) * (max(r^*_{ij}, 0) + 2 * |min(r^*_{ij}, 0)|) \qquad (15)$$

The first two indicators have been extracted directly from [14]. The rest of indicators have been adapted taking into account the features of the problem under consideration. Considering the similarity between the availability instant of machine i, a_i plus the processing time of job j in this machine p_{ij}, and the availability instant of machine a_{i+1}, we would like $a_i + p_{ij}$ to be as closest as possible to a_{i+1} for machines $i = 1, \dots, m-1$, in order to avoid machine idle times. More specifically, we search for the job whose processing times adapt best to the steps determined by the availability vector, if scheduled in the first position. We thus define the availability step as $s_i = a_{i+1} - a_i$. In this line, indicators (3) to (6) depend on $p_{ij} - s_i$. If $p_{ij} - s_i > 0$ then machine $i+1$ has an idle time equal to $p_{ij} - s_i$ or allowing that other job could be sequenced on the step s_i if $p_{ij} - s_i < 0$. Therefore, $r_{ij} = p_{ij} - s_i$ is the residual value used to create indicators (3) to (6). In addition, the first job in the first machine has a high importance since it may imply idle times for the rest of machines. In order to capture this circumstance, we design indicator (7). Finally, another concept is the real residual, $r^*_{ij} = p_{ij} - s_i + max(p_{i-1j} - s_{i-1}, 0) = r_{ij} + max(r_{i-1j}, 0)$ with $i = 2, \dots, m-1$. This expression allow us to compare each availability step with the processing time plus the machine idle time. Indicators (8) to (15) use this concept.

Regarding the sorting criteria C, we consider the following:

- INCR: sort the jobs according to increasing indicator values.
- DECR: sort the jobs according to decreasing indicator values.

Combining indicators with criteria (p_j, C) we can construct 15×2 different approaches for the phase I. We define this most basic class of algorithm as *Simple Sorting*. In addition, we develop two classes of algorithms based on applying phase I twice. We label these algorithms *Simple-Job Fitting* and *Multi-Job Fitting* respectively. The idea is based on the adaptation of jobs to the profile defined by the unavailability constraint. Intuitively, the better the adaptation, the higher the reduction in machine idle time and, consequently, the reduction in the makespan. So, the algorithms consist on the following steps:

Simple-Job Fitting algorithm sorts the first job according to a tuple (p_j,INCR), i.e. the first job on the sequence has the minimum value of p_j, in order to find the job which achieves the best adaptation to the profile defined by the unavailability constraint. The others jobs are arranged according to a different tuple (p'_j, C'), sorting them by simple sorting.

Multi-Job Fitting algorithm pretends to adapt the greatest number of jobs on all availability steps, according to a tuple (p_j,INCR). The rest of the jobs are arranged according to a new selected pair (p'_j, C').

The pseudocodes of these three types of algorithms are presented on Figure 5.

******* Insert Figure 5 about here *********

Note that all possibilities combining each algorithm type with indicators and criteria have not been used. *Simple-Job Fitting* and *Multi-Job Fitting* depend on *Simple Sorting* algorithm result (it is shown in the following subsection). Table 4 indicates for all cases the algorithm type, indicators and criteria, heuristic name, phase and complexity. In total 60 heuristics have been developed. The evaluation of these heuristics is conducted in the next subsection.

******* Insert Table 4 about here *********

3.1 Evaluation of Initial Sequence Heuristics

For testing the proposed Initial Sequence Heuristics, we adapt to our problem the well known Taillard's test bed [34]. This testbed consists of 120 instances of various sizes, $n \in \{20, 50, 100, 200, 500\}$ and $m \in \{5, 10, 20\}$. There are 10 instances for each size: 20x5, 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x10, 200x20 and 500x20. These instances are extremely difficult to solve for the makespan objective and therefore constitute an excellent benchmark to test different solution procedures for the P_O problem [29], being widely used in this context (see e.g. [13, 19, 29, 30]). We have adapted the benchmark to our problem by creating an availability vector a for each instance from the completion time of a random sequence in each machine as described in section 2.1. Furthermore, we have considered $k = 1$. Each problem has been solved by the Initial Sequence Heuristics to calculate the corresponding relative percentage deviation (RPD) from the objective function value of the best approach or from the optimal solution, respectively (%). The following expression has been employed for RPD:

$$RPD = \frac{C_{max}(HEUR) - C_{max}(BEST)}{C_{max}(BEST)} \cdot 100$$

where $C_{max}(HEUR)$ is the makespan obtained by heuristic $HEUR$ and $C_{max}(BEST)$ the best known makespan for each instance. Since there is no benchmark for our problem, this solution is the best among all heuristics tested. These results are then analysed by using statistical methods following three steps presented below (for all cases the significance level is the usual 0.05):

1. Analysis of *Simple Sorting* algorithm: First, we have obtained the corresponding RPD results for all possible heuristics created from the *Simple Sorting* class of algorithms. The objective is to determine the pair indicator-criterion which provides the best result by *Simple Sorting* in order to embed it into *Simple-Job Fitting* and *Multi-Job Fitting* classes of algorithms.

   To check the influence of indicators and criteria on the results, we have considered RPD as dependent variable, and two factors: Indicator I (with 15 levels) and Criterion C (with 2 levels). As the data do not verify the homoscedasticity condition (p-values are lower than 0.05 in the Levene tests for both factors), non-parametric tests have to be used to study the influence of the factors in the dependent variable. These tests reveal that there are differences between levels of C and I. Therefore, it can be concluded that RPD results are different depending on these factors. The best result of RPD is obtained for the heuristic with indicator (2), WSUM, and criterion INCR. There are not significative differences between it and other two heuristics ((WNEG RR,DECR) and (WPOS2NEG RR, DECR)), but they are different of the rest 27 heuristics developed by *Simple Sorting* algorithm.

2. Complete analysis: With the previous result, *Simple-Job Fitting* and *Multi-Job Fitting* algorithms have been applied to solve all adapted Taillard instances. As it is shown in Table 4, in both cases, the first pair (p_j, C) is formed by the indicators from (1) to (15) combined with the criteria INCR. The second tupla used in both cases is the best

obtained from *Simple Sorting*, i.e. the tupla (WSUM, INCR). It can be observed that we have the same algorithm three times when the first application of the tupla (p_j, C) fit with the second application on *Simple-Job Fitting* and *Multi-Job Fitting*, implying the same result than the heuristic with (WSUM, INCR) for the *Simple Sorting* algorithm.

The ARPD (Average RPD) values obtained for each heuristic have been represented in the Figure 6.

******* Insert Figure 6 about here *********

In Figure 6 it can be observed that, in general, the best results are obtained in the case of SF. Result for the indicator WSUM is the same for SS, SF and MF (as mentioned previously), being a good result too. For SF and MF there are not great differences between indicators (it can be observed that lines belonging to them are almost horizontal). However, results from *Simple Sorting* have a high variability.

We have developed a new analysis to study these results from a statistical viewpoint. The design considers RPD values as dependent variable, factor Algorithm Type T with 4 levels (SS, SS*, SF and MF), and factor Indicator I with 15 levels. Results have been analyzed by the same non-parametric tests used before (since Levene test for two factors gives a p-value lower than 0.05, implying that the homoscedasticity condition is not verified).

For Algorithm Type factor, non-parametric test reveals that the p-value in the Kruskal-Wallis test is lower than 0.05. So the results are different depending on the selected algorithm type. Mann-Whitney tests reveals that there are not similarities between results since all of p-values are lower than the Bonferroni's corrected significance 0.008, except the case SF-MF which p-value is equal to 0.013 greater than 0.008. ARPD for groups in homogeneous subsets are displayed in Table 5, i.e. those pairs without significative differences have been represented in the same column.

******* Insert Table 5 about here *********

With respect to Indicator factor, we have developed a Kruskal-Wallis test for each Algorithm type. The p-values are lower than 0.05 in the cases SS and SS*, but is 1.000 and 0.930 for SF and MF respectively, so we can conclude that *Simple Sorting* implies different result depending on the indicator selected while indicators selected belong to different groups (see Table 6 for SS and Table 7 for SS*) but *Simple-Job Fitting* and *Multi-Job Fitting* provide similar results for all indicators.

******* Insert Table 6 about here *********

In Table 6 we study the similarity between results grouped by indicator in the case SS. It can be observed that results are different for SUM and WSUM, but there are not differences between the rest of indicators.

******* Insert Table 7 about here *********

In Table 7 presents the case SS*. There are four set to group results with overlapped results.

This analysis allows us to conclude that SF and MF are the best algorithm types (without significative differences between them), regardless the selected indicator since there are not differences between them for these algorithm types. *Simple Sorting* provides worse results, and it depends on the indicator selected.

3. Disaggregated analysis: A complete representation of the results for all heuristics is shown in Table 8, in terms of ARPD and ranks.

   ******* Insert Table 8 about here *********

   Furthermore, Table 8 presents the ARPD of Algorithm Types, as well as the ARPD of Indicators.

   The best results are obtained by heuristic MF combined with the indicator (1), i.e. SUM. This heuristic uses the *Multi-Job Fitting* algorithm, with the pair indicator-criterion (p_j, C) =(WSUM,INCR) in the first application of phase I, (p'_j, C') =(SUM,INCR) in the second application of this phase. This algorithm searches for the jobs better adapted to the availability profile according to the SUM indicator and schedules it first. The rest of jobs are scheduled according to the best heuristic from *Simple Sorting* algorithm, defined by WSUM indicator and INCR criterion.

   Moreover, the best ten results are remarked. Almost all of them belong to SF. It can be observed that there are very similar results. The four, five and six belong to the same heuristic as it was mentioned previously. In statistical terms (considering RPD as dependent variable and heuristics as factor with 60 levels), there are not significative differences between the thirty four best heuristics, most of them belong to SF and MF algorithm types (these thirty four best results can be observed in the Figure 6, the 30 results for SF and MF, and four points near to these results belonging to SS combined with WSUM and SS* combined with NEG RR, WNEG RR and WPOS2NEG). The high similarity between SF and MF, presented previously in Table 5, is corroborated in this disaggregated analysis.

In conclusion, the developed analysis reveals that there are a high similarity between the two best algorithm types, SF and MF, regardless the indicator, as there are not differences between them. The disaggregated analysis considering all developed heuristics shows that, as well as all heuristics belonging to SF and MF, there are another four without significative differences. SS combined with WSUM is one of them, it is the best obtained from *Simple Sorting* algorithm and the fourth best on the rank presented in Table 8. It uses an algorithm type with less complexity than SF and MF (see Table 4). So we can conclude that a fast method which orders jobs in ascending order of the indicator WSUM provides very good solutions for our problem. These results support the conclusion from the previous analysis presented in section 2, i.e.: As the problem is easier than P_O, a simple algorithm (*Simple Sorting* algorithm) can achieve a good makespan. In general, algorithms adapted to the problem (*Simple-Job Fitting* and *Multi-Job Fitting* algorithms) provide better results, but finally we have proved that an special case of the former, with (WSUM,INCR), gives a result as good as the latter.

4 Conclusions

This paper aims at a special case of machine availability constraint in permutation flowshops, which is motivated by the need of setting a common due date for a set of jobs while the machines are still busy processing another set of jobs previously scheduled. We label this problem as P_A. The problem is NP-hard in the strong sense for more than two machines. We analyze the problem to determine its differences and degree of difficulty as compared to the widely studied original permutation flowshop problem, denoted as P_O. To do so, a design of experiments was carried out to check the influence of a number of factors (machines, jobs, distribution of processing times, unavailability of machines, etc.) on the structure of solutions of P_A. The results show that P_A and P_O are different problems with respect to the values of the objective function and also with respect to the structure of the space of solutions. The results indicate that P_A is easier than P_O, particularly for structured problems, which are the most usual type found in real-world. This means that the 'classical' assumption of machine availability (which is not commonplace in real-life environments) generates a problem much more difficult than the initial availability constraint problem.

Given these results we develop fast methods to solve our problem in order to provide a nearly real-time response to due-date quotation request by customers. To do so, we build a number of fast heuristics based on the profile defined by the availability constraint, i.e.: *Simple Sorting*, *Simple-Job Fitting* and *Multi-Job Fitting* classes of algorithms. All of them have been combined with different criteria and indicators to sort the jobs. The developed analysis shows that there are not statistical differences between results obtained from *Simple-Job Fitting* and *Multi-Job Fitting* algorithms, and these results are independent of the indicator selected.

Future research could include the case where jobs already scheduled are not frozen, but can be rescheduled whenever their committed due dates are not violated. If it is a common due date with some slack, we can try to determine a sequence with old and new jobs, in such a way that already scheduled jobs fulfil their due date while new jobs minimize the makespan, thus guaranteing the best possible common due date for this new set of jobs.

References

[1] Exploiting problem structure in scheduling. Colorado State University, USA, 2003. Retrieved May 14, 2008 from http://www.cs.colostate.edu/sched.

[2] R. Aggoune. Minimizing the makespan for the flow shop scheduling problem with availability constraints. *European Journal of Operational Research*, 153(3):534–543, 2004.

[3] R. Aggoune, A. H. Mahdi, and M. C. Portmann. Genetic algorithms for the flow shop scheduling problem with availability constraints. volume 4 of *IEEE International Conference on Systems, Man, and Cybernetics 2001*, pages 2546–2551, 2001.

[4] R. Aggoune and M. C. Portmann. Flow shop scheduling problem with limited machine availability: A heuristic approach. *International Journal of Production Economics*, 99(1-2):4–15, 2006.

[5] H. Allaoui, A. Artiba, S. E. Elmaghraby, and F. Riane. Scheduling of a two-machine flowshop with availability constraints on the first machine. *International Journal of Production Economics*, 99(1-2):16–27, 2006.

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

[7] J. Breit. An improved approximation algorithm for two-machine flow shop scheduling with an availability constraint. *Information Processing Letters*, 90(6):273–278, 2004.

[8] J. Breit. A polynomial-time approximation scheme for the two-machine flow shop scheduling problem with an availability constraint. *Computers & Operations Research*, 33(8):2143–2153, 2006.

[9] A. G. Celia, N. D. Gupta, and C. N. Potts. Two-machine no-wait flow shop scheduling with missing operations. *Mathematics of Operations Research*, 24(4):911, 1999.

[10] T. C. E. Cheng and Z. Liu. 3/2-approximation for two-machine no-wait flowshop scheduling with availability constraints. *Information Processing Letters*, 88(4):161–165, 2003.

[11] T. C. E. Cheng and Z. Liu. Approximability of two-machine no-wait flowshop scheduling with availability constraints. *Operations Research Letters*, 31(4):319–322, 2003.

[12] J. M. Framinan, J. N. D. Gupta, and R. Leisten. A review and classification of heuristics for permutation flow-shop scheduling with makespan objective. *Journal of the Operational Research Society*, 55(12):1243–1255, 2004.

[13] J. M. Framinan and R. Leisten. Total tardiness minimisation in permutations flow shops: a simple approach based on a variable greedy algorithm. *International Journal of Production Research*, In Press, 2007. To link to this article: http://dx.doi.org/10.1080/00207540701418960.

[14] J. M. Framinan, R. Leisten, and C. Rajendran. Different initial sequences for the heuristic of Nawaz, Enscore and Ham to minimize makespan, idletime or flowtime in the static permutation flowshop sequencing problem. *International Journal of Production Research*, 41(1):121–148, 2003.

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

[16] R. Graham, E. Lawer, J. Lenstra, and A. Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: A survey. *Annals of Discrete Mathematics*, 5:287–326, 1979.

[17] N. Hefetz and I. Adiri. A note on the influence of missing operations on scheduling problems. *Naval Research Logistics*, 29(3):535–539, 1982.

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

[19] P. J. Kalczynski and J. Kamburowski. An improved NEH heuristic to minimize makespan in permutation flow shops. *Computers & Operations Research*, 35(9):3001–3008, 2008.

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

[21] C. Y. Lee. Machine scheduling with availability constraints. In J. Y. T. Leung, editor, *Handbook of scheduling*, book chapter 22, pages 22.1–22.13. Boca Raton: CRC Press, 2004.

[22] C. Y. Lee, L. Lei, and M. Pinedo. Current trends in deterministic scheduling. *Annals of Operations Research*, 70(0):1–41, 1997.

[23] D. C. Montgomery. *Design and Analysis of Experiments.* John Wiley and Sons, New York (USA), 6th edition, 2005.

[24] C. T. Ng and M. Y. Kovalyov. An FPTAS for scheduling a two-machine flowshop with a one unavailability interval. *Naval Research Logistics*, 51(3):307–315, 2004.

[25] E. Nowicki and C. Smutnicki. A fast tabu search algorithm for the permutation flowshop problem. *European Journal of Operational Research*, 91(1):160–175, 1996.

[26] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems.* Prentice Hall International Series in Industrial and System Engineering. Prentice Hall, Englewood Cliffs, New Jersey (USA), 1995.

[27] C. Rajendran and H. Ziegler. A performance analysis of dispatching rules and a heuristic in static flowshops with missing operations of jobs. *European Journal of Operational Research*, 131(3):622–634, 2001.

[28] R. Ruiz, J. Carlos Garcia-Diaz, and C. Maroto. Considering scheduling and preventive maintenance in the flowshop sequencing problem. *Computers & Operations Research*, 34(11):3314–3330, 2007.

[29] R. Ruiz and C. Maroto. A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(3):479–494, 2005.

[30] R. Ruiz and T. Sttzle. A simple and effective iterated greedy algorithm for the permutation flowshop scheduling problem. *European Journal of Operational Research*, 117(3):2033–2049, 2007.

[31] E. Sanlaville and G. Schmidt. Machine scheduling with availability constraints. *Acta Informatica*, 35(9):795–811, 1998.

[32] G. Schmidt. Scheduling with limited machine availability. *European Journal of Operational Research*, 121(1):1–15, 2000.

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

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

[35] A. Turkcan. Machine scheduling with availability constraints. Technical report, Industrial Engineering Department, Bilkent University. Ankara (Turkey), 1999. Retrieved May 14,2008 from http://www.ie.bilkent.edu.tr/ ie672/docs/./present/turkcan.ps.

[36] X. Wang and T. C. E. Cheng. An approximation scheme for two-machine flowshop scheduling with setup times and an availability constraint. *Computers & Operations Research*, 34(10):2894–2901, 2007.

[37] X. Wang and T. C. E. Cheng. Heuristics for two-machine flowshop scheduling with setup times and an availability constraint. *Computers & Operations Research*, 34(1):152–162, 2007.

[38] J. P. Watson, L. Barbulescu, L. Darrell Whitley, and A. E. Howe. Constrasting structured and random permutation flowshop scheduling problems: search-space topology and algorithm performance. Technical report, The GENITOR Research Group in Genetic Algorithms and Evolutionary Computation, Colorado State University. Fort Collins, Colorado (USA), 2002. Retrieved May 14,2008 from http://www.cs.colostate.edu/ genitor/2002/joc02.pdf.

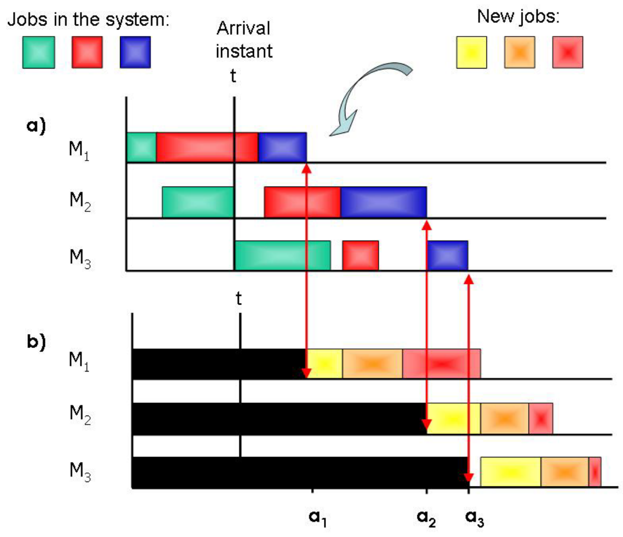

Figure 1: Feature of the initial availability constraint problem

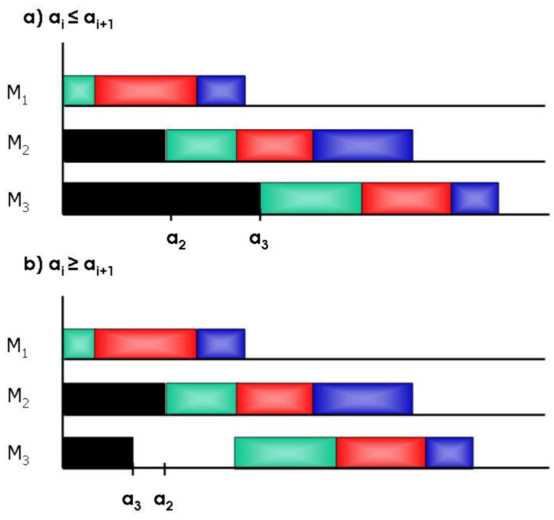

Figure 2: Representation of the initial availability constraint problem

Estimated Marginal Means

Dependent variable: Optimal Makespan

Number of Jobs = 5

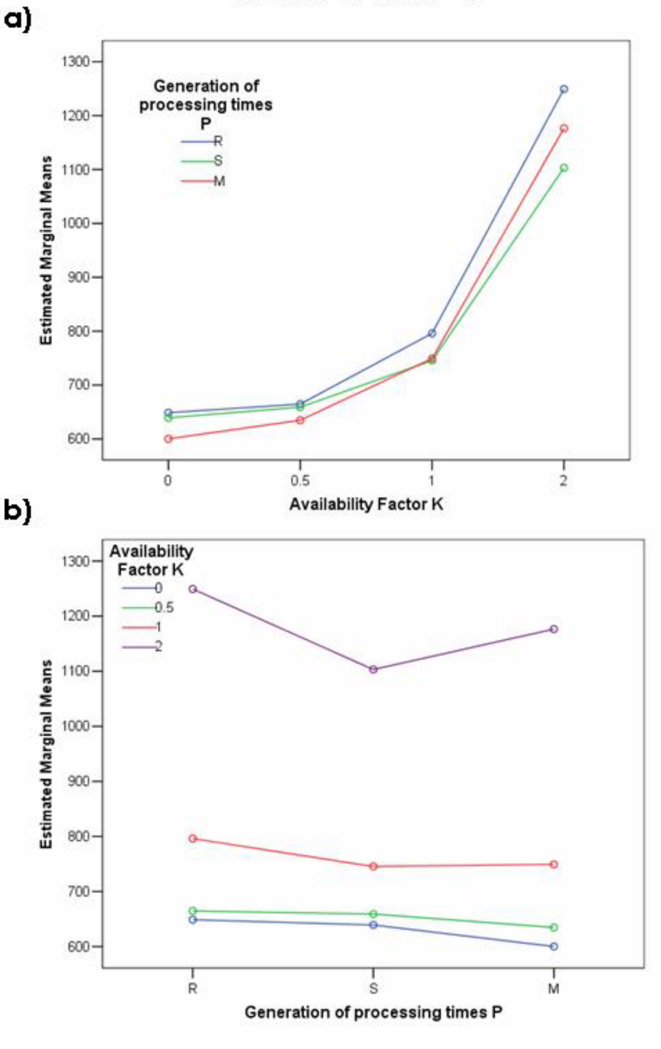

Figure 3: Estimated marginal means of Optimal Makespan

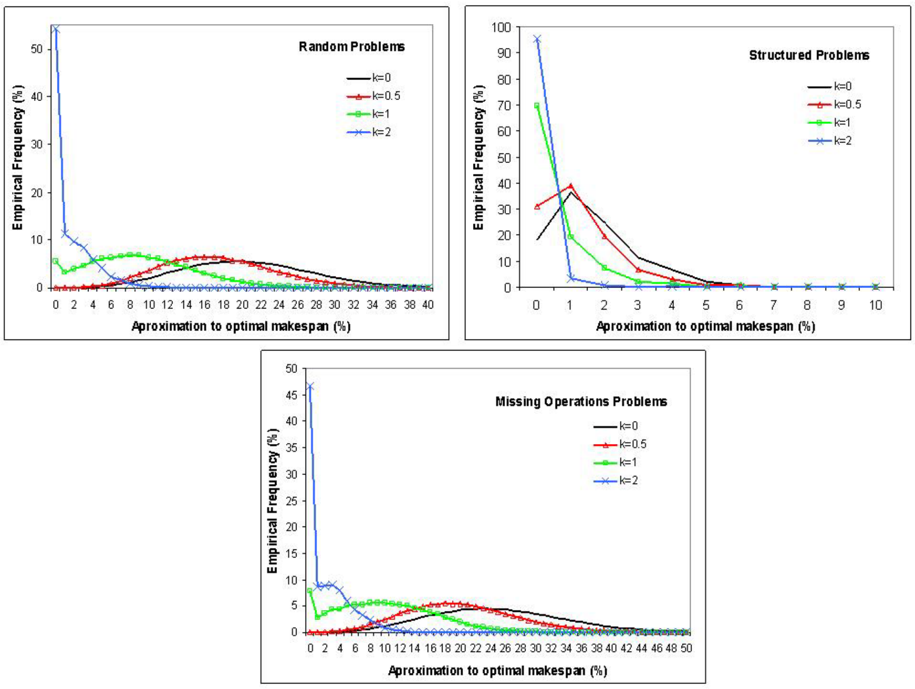

Figure 4: Distribution of solutions for small problems

Simple Sorting

Stage 1: $\forall j \in J$ are sorted according to p_j by using criterion C.

Simple-Job Fitting

Stage 1: $j_1 = \min_{j \in J} p_j$.

Stage 2: $\forall j \in J \backslash \{j_1\}$ are sorted according to p'_j by the criterion C

Multi-Job Fitting

Stage 1: $j_1 = \min_{j \in J} p_j$.

Stage 2: If $\exists i/\ s_i - p_{ij_1} < 0 \Rightarrow$ Stage 3. Else if $\forall i\ s_i - p_{ij_1} \geq 0 \Rightarrow a_i = a_i + p_{ij_1}$, then calculate new steps s_i and return to Stage 1 with $J = J \backslash \{j_1\}$.

Stage 3: $\forall j \in J \backslash \{j_1\}$ are sorted according to p'_j by the criterion C'.

Figure 5: Algorithms

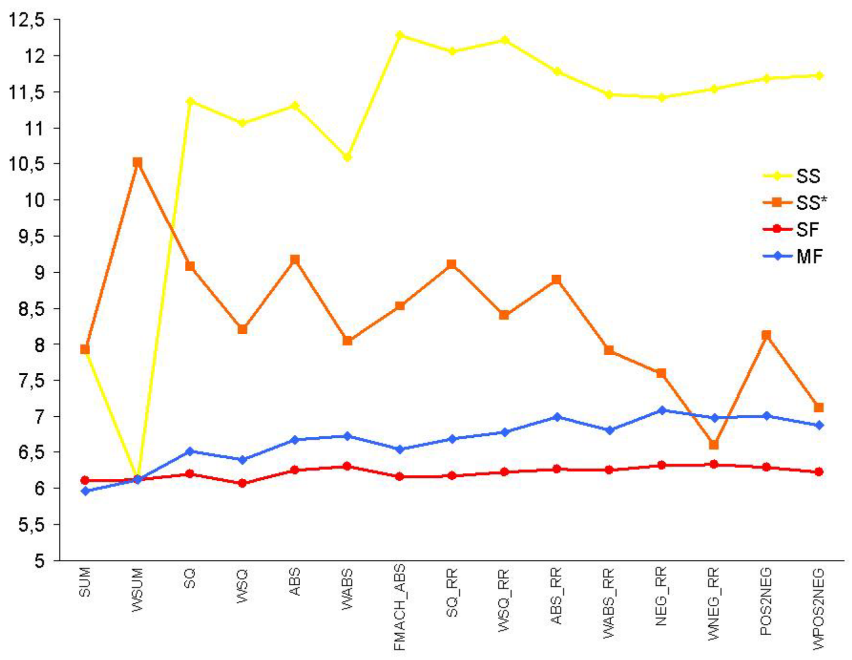

Figure 6: ARPD of heuristics

Description	**Factor**	**Level**	**Runs**
Number of jobs	N	5	2400
		10	2400
Number of machines	M	5	2400
		10	2400
Generation of processing times	P	R	1600
		S	1600
		M	1600
Availability factor	K	0	1200
		0.5	1200
		1	1200
		2	1200

Table 1: Design of Experiments Data

	Factor P			Factor K: Case $p = M$			
Subset	1	2	Subset	1	2	3	4
$p = M$	950.06		$k = 0$	713.48			
$p = S$		995.14	$k = 0.5$		764.45		
$p = R$		1008.15	$k = 1$			918.85	
			$k = 2$				1403.46

	Factor K: Case $p = R$				Factor K: Case $p = S$		
Subset	1	2	3	Subset	1	2	3
$k = 0$	777.64			$k = 0$	807.27		
$k = 0.5$	785.57			$k = 0.5$	845.39		
$k = 1$		978.09		$k = 1$		965.58	
$k = 2$			1491.3	$k = 2$			1362.33

Table 2: Homogeneous subsets for factors P and K based on Mann-Whitney Tests

P	K	5x5		5x10		10x5		10x10		Total	
		Mean	95%	Mean	95%	Mean	95%	Mean	95%	Mean	95%
$p = R$	$k = 0$	16.57	39	13.50	29	22.90	39	20.58	34	18.39	35.25
	$k = 0.5$	12.84	31	11.18	24	16.45	32	17.44	29	14.48	29.00
	$k = 1$	7.38	22	6.40	16	7.80	21	9.13	20	7.68	19.75
	$k = 2$	2.51	12	1.06	6	2.32	11	1.46	7	1.84	9.00
$p = S$	$k = 0$	1.03	4	1.56	6	0.78	3	1.62	5	1.25	4.50
	$k = 0.5$	0.83	4	1.35	6	0.49	3	1.15	4	0.95	4.25
	$k = 1$	0.35	3	0.65	4	0.19	2	0.47	3	0.41	3.00
	$k = 2$	0.08	2	0.06	1	0.08	2	0.07	1	0.07	1.50
$p = M$	$k = 0$	19.14	44	15.63	33	28.26	49	24.78	41	21.95	41.75
	$k = 0.5$	14.96	34	13.12	31	20.01	39	20.12	34	17.05	34.50
	$k = 1$	9.02	25	7.98	20	10.21	29	10.01	22	9.30	24.00
	$k = 2$	2.48	12	1.62	8	3.35	15	2.12	9	2.39	11.00

Table 3: Mean and 95% of approximation to the optimal makespan

Algorithm	(p_j, C)	(p'_j, C')	Name	Phase	Complexity	Heuristics
Simple Sorting	(p_j,INCR)	X	SS	I	$O(nlog(n))$	15
	(p_j,DECR)	X	SS*	I	$O(nlog(n))$	15
Simple-Job Fitting	(p_j,INCR)	(WSUM,INCR)	SF	I+I	$O(nlog(n))$	15
Multi-Job Fitting	(p_j,INCR)	(WSUM,INCR)	MF	I+I	$max(O(nlogn), O(nm))$	15
	p_j = (1) to (15)				Total heuristics	60

Table 4: Heuristics

Algorithm Type	Obs.	Subset 1	Subset 2	Subset 3
SF	1800	6.2219		
MF	1800	6.6783		
SS*	1800		8.3442	
SS	1800			10.964

Table 5: Homogeneous subsets for factor T based on Mann-Whitney Tests

Indicator	Obs.	Subset 1	Subset 2	Subset 3
WSUM	120	6.1218		
SUM	120		7.9147	
WABS	120			10.5885
WSQ	120			11.0562
ABS	120			11.2921
SQ	120			11.3587
NEG RR	120			11.4218
WABS RR	120			11.4542
WNEG RR	120			11.5392
POS2NEG	120			11.6827
WPOS2NEG	120			11.7234
ABS RR	120			11.7771
SQ RR	120			12.0468
WSQ RR	120			12.2079
FMACH	120			12.2751

Table 6: Homogeneous subsets for factor I based on Mann-Whitney Tests. Case: Algorithm Type SS

Indicator	Obs.	Subset			
		1	2	3	4
WNEG RR	120	6.5965			
WPOS2NEG	120	7.1064	7.1064		
NEG RR	120	7.5850	7.5850	7.5850	
WABS RR	120	7.8996	7.8996	7.8996	
SUM	120	7.9242	7.9242	7.9242	
WABS	120	8.0408	8.0408	8.0408	
POS2NEG	120	8.1153	8.1153	8.1153	
WSQ	120	8.2012	8.2012	8.2012	
WSQ RR	120	8.3946	8.3946	8.3946	
FMACH	120		8.5271	8.5271	
ABS RR	120		8.8998	8.8998	8.8998
SQ	120			9.0806	9.0806
SQ RR	120			9.1037	9.1037
ABS	120			9.1700	9.1700
WSUM	120				10.5179

Table 7: Homogeneous subsets for factor I based on Mann-Whitney Tests. Case: Algorithm Type SS*

Indicator	SS		SS*		SF		MF		
	ARPD	Position	ARPD	Position	ARPD	Position	ARPD	Position	Average
SUM	7.9147	36	7.9242	37	6.1109	**3**	5.9676	**1**	6.9793
WSUM	6.1218	**4**	10.5179	47	6.1218	**5**	6.1218	**6**	7.2208
SQ	11.3587	51	9.0806	44	6.2035	**9**	6.5197	20	8.2906
WSQ	11.0562	49	8.2012	40	6.0691	**2**	6.3955	19	7.9305
ABS	11.2921	50	9.1700	46	6.2607	13	6.6780	23	8.3502
WABS	10.5885	48	8.0408	38	6.3058	16	6.7274	25	7.9156
FMACH ABS	12.2751	60	8.5271	42	6.1628	**7**	6.5394	21	8.3761
SQ RR	12.0468	58	9.1037	45	6.1772	**8**	6.6871	24	8.5037
WSQ RR	12.2079	59	8.3946	41	6.2214	**10**	6.7861	26	8.4025
ABS RR	11.7771	57	8.8998	43	6.2638	14	6.9893	30	8.4825
WABS RR	11.4542	53	7.8996	35	6.2607	12	6.8143	27	8.1072
NEG RR	11.4218	52	7.5850	34	6.3183	17	7.0830	32	8.1020
WNEG RR	11.5392	54	6.5965	22	6.3312	18	6.9836	29	7.8626
POS2NEG	11.6827	55	8.1153	39	6.2941	15	7.0034	31	8.2739
WPOS2NEG	11.7234	56	7.1064	33	6.2265	11	6.8778	28	7.9835
Average	10.9640		8.3442		6.2219		6.6783		

Table 8: ARPD and Ranks for Heuristic results