On heuristic solutions for the stochastic flowshop scheduling problem

Article *in* European Journal of Operational Research · May 2015

DOI: 10.1016/j.ejor.2015.05.006

Jose M. Framinan
Universidad de Sevilla

Paz Perez-Gonzalez
Universidad de Sevilla

On heuristic solutions for the stochastic flowshop scheduling problem*

Jose M. Framinan, Paz Perez-Gonzalez

May 21, 2015

Abstract

We address the problem of scheduling jobs in a permutation flowshop when their processing times adopt a given distribution (stochastic flowshop scheduling problem) with the objective of minimisation of the expected makespan. For this problem, optimal solutions exist only for very specific cases. Consequently, some heuristics have been proposed, all of them with similar performance. In our paper, we first focus on the critical issue of estimating the expected makespan of a sequence and found that, for instances with a medium/large variability (expressed as the coefficient of variation of the processing times of the jobs), the number of samples or simulation runs used in the literature may not be sufficient to derive robust conclusions. We thus propose a procedure with a variable number of iterations that ensures that the percentual error in the estimation of the expected makespan is bounded with a very high probability. Using this procedure, we test the main heuristics proposed in the literature and find significant differences in their performance, in contrast with existing studies. We also find that the deterministic counterpart of the most efficient heuristic for the stochastic problem performs extremely well for most settings, which indicates that (at least within the limitations of our study), a practical way to solve the stochastic problem may be to simplify it to its deterministic version.

Keywords: Scheduling, Flowshop, Stochastic, Makespan Objective, Heuristics

*Preprint submitted to European Journal of Operational Research

1 Introduction

The flowshop scheduling problem with makespan objective (usually denoted as $Fm|prmu|C_{max}$) has been subject of research for more than 60 years, being one of the most comprehensively studied problems in Operations Research (see in this regard the reviews by Framinan et al., 2004, Reza Hejazi and Saghafian, 2005 and Ruiz and Maroto, 2005). This decision problem consists of how to schedule jobs in a permutation flowshop in order to minimize the maximum completion time or makespan. A classical assumption is that the processing times of each job in each machine are considered different, but known in advance (deterministic). In contrast, our paper deals with the problem of scheduling n jobs in a permutation flowshop consisting of m machines where the processing times are not deterministic, but follow some known distribution. The objective considered is that of minimizing the expected makespan. This problem is considered to be more realistic that their deterministic counterpart, as it allows capturing part of the inherent variability present in many real-life manufacturing environments (see e.g. Hopp and Spearman, 2008). In the following, we will denote our problem as $Fm|prmu|E[C_{max}]$.

The $Fm|prmu|E[C_{max}]$ problem has been much less studied than its deterministic counterpart, and it is clearly much more complex. In fact, apart from a dominance rule obtained by Makino (1965) for the case of two jobs, no exact solution is available without assumptions on the distribution of the processing times. For $m = 2$ and exponential distribution of the processing times, Talwar (1967) conjectured an exact solution for the problem that was later proved to be optimal by Cunningham and Dutta (1973), and is currently known as Talwar's rule.

Despite these advances, for the rest of the cases, no optimal procedure has been found. For the two-machine case, three approximate solutions have been proposed by Baker and Trietsch (2011) based both in Talwar's rule and in Johnson's rule (Johnson, 1954) for the deterministic flowshop, all of them with similar (near optimal) performance. For the general m machine case, Baker and Altheimer (2012) suggest three heuristics based on adaptations of the CDS (Campbell et al., 1970) and NEH (Nawaz et al., 1983) heuristics, again with similar and near optimal performance. Although it might seem that, from these results, the problem $Fm|prmu|E[C_{max}]$ is already solved, some issues have to be discussed:

- First of all, the evaluation of sequences in an stochastic flowshop is far from being a trivial task. Since the objective is to obtain the expected makespan of a given sequence, $E[C_{max}]$ has to be estimated by running N simulations using the sequence as a solution, from which a sample C^i_{max} ($i = 1, \ldots, N$) is obtained. Then, $E[C_{max}]$ is estimated by averaging the sample, i.e. $E[C_{max}] \approx \bar{C}_{max} = \frac{1}{N}\sum_{i=1}^{N} C^i_{max}$.

  Up to now, there is no standardised procedure to determine N, although the authors of related contributions use a large number in order to ensure the significance of the estimation. Thus, Baker and Altheimer (2012) use $N = 100,000$ whereas Gourgand et al. (2003) employ $N = 200,000$ regardless the size and characteristics of each instance, while Portougal and Trietsch (2006) set N to $10,000$ for the 2-machine case. In addition, there is no mechanism to establish the statistical significance of the so-obtained $\bar{C}_{max}$ and, consequently, to assess the differences in the performance among different heuristics.

- Due to the computational complexity of the stochastic problem, the experiments in the literature have been limited to very small problem sizes (up to $n = 10$ and $m = 6$ in the most recent studies). This makes the conclusions obtained so far to be restricted to very small problem sizes, and perhaps not valid for bigger problem sizes.

- Finally, to the best of our knowledge, the neccessity of heuristics specifically designed for the stochastic problem has not been yet determined. In other words, one may try to solve the stochastic flowshop scheduling problem by transforming it into its deterministic counterpart, i.e. by obtaining a flowshop with the same number of jobs and machines but with deterministic processing times equal e.g. to the means of those from the stochastic problem. Then, heuristics for the $Fm|prmu|C_{max}$ problem can be applied and a (possibly good) sequence for the deterministic problem can be obtained. If this sequence performs well when applied to the stochastic problem, then the need of specific stochastic heuristics can be questioned. However, such test has not been conducted so far. It is foreseeable that the so-obtained sequences perform worse that those specifically designed for the stochastic version, but maybe the differences in the quality of the results do not justify the much higher computation times required for the stochastic heuristics. Even if the deterministic heuristics are not valid for some cases, it would be interesting to quantify the degree of variability for which using them is still acceptable, as it is clear that an stochastic flowshop with low variability would resemble very much to a deterministic flowshop.

With these issues in mind, we first discuss and propose a procedure for estimating the expected makespan of a sequence in an stochastic flowshop, so the error in such estimation is bounded by a given percentage. In this way, the statistical significance of the results obtained by the different heuristics can be more clearly established. Interestingly, the results show that the sample sizes (N) obtained from our procedure proposed range from very small to very high values, thus supporting the conclusion that no predetermined value can be easily found regardless the variability of the instances and the error assumed in the estimation.

Next, we compare the main heuristics proposed in the literature as well as the expected makespan obtained from the application of purely deterministic procedures to the mean processing times of the stochastic problem. These heuristics are tested for problem sizes larger than those presented so far in the literature, so the conclusions from the results can be better supported. The results show that, in contrast to Baker and Altheimer (2012), there are significant differences in the performance of the heuristics, and that –perhaps not so surprisingly– the performance of the sequences obtained from purely deterministic methods in the stochastic flowshop do not differ greatly from that obtained from specific stochastic methods.

The remainder of the paper is organised as follows: First, the problem under consideration is formally described in Section 2, where the main contributions of the literature are discussed. Since our work is of computational nature, we devote Section 3 to discuss the key issue of the testbed in which the heuristics are to be compared, as we intend to capture different problem sizes and different degrees of variability of the flowshop. Next, we present in Section 4 the procedure to estimate the expected mean makespan of a given solution, and compare the number of iterations required with those employed in the literature. The comparison of the performance of different heuristics for the problem is done in Section 5, where the main results are also discussed. Finally, Section 6 present the conclusions and points out future research lines.

2 Background

A flowshop consists of n jobs that must be processed on m machines in the same order, where job i requires p_{ij} time units to be processed on machine j. The scheduling problem in flow shops is to find a sequence of jobs for each machine according to certain performance measure(s). Additionally, for many situations, it is assumed that the job sequences will be the same on every machine (permutation flowshops). Other hypotheses common in scheduling research are, e.g. the simultaneous availability of all jobs and of all machines, deterministic processing times, etc. For a complete list of these assumptions, see e.g. Framinan et al. (2004).

While the deterministic flowshop scheduling problem with makespan objective has been extensively studied (see the reviews mentioned above), the same cannot be said about its stochastic counterpart. For the two-jobs case and a general distribution of the processing times, a dominance rule is given by Makino (1965), but this result is extremely restrictive and with little applicability for most practical settings.

By making assumptions on the distribution of the processing times of the jobs, an important result is due to Talwar (1967). He conjectures that the expected makespan is minimized when the processing time of the jobs follows an exponential distribution by sequencing the jobs in non increasing order of $\frac{1}{\mu_{i1}} - \frac{1}{\mu_{i2}}$, where μ_{ij} is the mean processing times of job i on machine j. This order is proved to be optimal by Cunningham and Dutta (1973), and it is currently known as Talwar's rule. As an extension of this rule to other distributions, Kalczynski and Kamburowski (2006) heuristically adapt Talwar's rule for the Weibull distribution. For a general family of distributions, Portougal and Trietsch (2006) develop a heuristic named PSH which starts with the solution given by Johnson's rule (Johnson, 1954) for the deterministic flowshop, and applies an adjacent pairwise interchange (a reason why this heuristic is later renamed API by Baker and Trietsch, 2011). Finally, Baker and Trietsch (2011) test three different procedures for different families of distributions, i.e.: Talwar's rule, Johnson's rule, and the API heuristic. They conclude that the (deterministic) Johnson's rule could be better unless the coefficient of variation of the jobs is very high. For such cases, Talwar's rule or API may perform better.

For the general flowshop problem with m machines, Baker and Altheimer (2012) propose different heuristics. The first heuristic is called CDS/Johnson and consists of obtaining a set of $m-1$ 2-machine flowshop subproblems with the addition of the processing times of the jobs in the manner of the CDS heuristic by Campbell et al. (1970). More specifically, 2-machine flowshop subproblem k (with $k = 1, \ldots, m-1$) is constructed by obtaining the processing times of job i in the first (second) machine of this subproblem as $A_i = \sum_{j=1}^{j=k} p_{ij}$ ($B_i = \sum_{j=k+1}^{j=m} p_{ij}$). Then, Johnson's procedure is applied to each of the resulting $m-1$ subproblems, and $m-1$ sequences are obtained. The estimation of the expected makespan of each of this sequences is obtained (in their case by running 100,000 simulations and taking the average makespan value), and the one yielding the lowest value is selected.

The second tested heuristic is the CDS/Talwar heuristic. In a similar manner to the previous one, a set of $m-1$ 2-machine flowshop subproblems are obtained, and a sequence is obtained for each one by applying Talwar's rule. Out of these $m-1$ sequences, the one with the minimum estimation of the expected makespan is selected.

The third heuristic tested is based on the famous NEH heuristic proposed by Nawaz et al. (1983) for the deterministic case. This heuristic consists of two phases: First the jobs are ranked according to the descending sum of their mean processing times. In a second phase, a solution is constructed as follows: Starting from a partial sequence constructed by taking the first job of the rank, then, for $k = 2, \ldots, n$, k partial sequences are constructed by inserting the k-th job of the rank in all k slots of the partial sequence. These k partial sequences are evaluated with respect to their expected makespan (estimated, as in previous cases, by means of obtaining a sample via simulation and taking the average value), and the one obtaining the lowest value of the estimation of the expected makespan is retained as partial sequence for step $k+1$. The procedure is repeated until a full sequence is obtained.

Among the three heuristics, Baker and Altheimer (2012) find that the NEH adaptation is the best one, but none of the heuristic procedures dominates the others. Furthermore, their performance is compared against that of a genetic algorithm (assumed to find the optimal or near-optimal sequence for most instances), and the authors conclude that the three heuristics generate average suboptimalities of less than 1%, leaving little room for the development of new approximate algorithms.

Despite the advances reported, there are some issues already discussed in Section 1 that affect the existing results and deserve further research. A closer look on how to estimate the expected makespan is needed, in order to add statistical consistency to the results. Furthermore, the neccessity of special stochastic heuristics for the problem has not been established, particularly when, for the two-machine case, Portougal and Trietsch (2006) state the excellent performance of the Johnson's deterministic heuristic in the stochastic setting. To address all these issues, we first need a benchmark testbed to conduct the computational experience. The design of this testbed is presented in the next section.

3 Testbed design

Regarding the design of a testbed for flowshop scheduling, we first have to decide about the problem sizes, i.e. the number of jobs n and machines m of the different instances. Given the computational complexity of the stochastic model, problem sizes have to be much more reduced than those in the deterministic counterpart, however we want to subtantially increase the existing problem sizes in order to gain generality on the results. With these premises, we choose $n \in \{5, 10, 15, 20\}$ and $m \in \{2, 5, 10, 20\}$.

In order to ease the analysis, we assume that all distribution of the processing times belong to the same family, which is a common assumption almost universally made (see e.g. Gourgand et al., 2003, Baker and Trietsch, 2011 or Baker and Altheimer, 2012). Regarding to the family of distributions, there are several option, including the random distribution (Baker and Trietsch, 2011 or Baker and Altheimer, 2012), the normal distribution (Gourgand et al., 2003) the exponential distribution (Gourgand et al., 2003, Baker and Trietsch, 2011 or Baker and Altheimer, 2012), and the lognormal distribution (Baker and Trietsch, 2011 or Baker and Altheimer, 2012).

After reviewing the different distributions, we will assume a log normal distribution for our testbed. The log normal distribution is characterised by two parameters (mean μ and standard deviation σ), and it has a considerable practical value for modelling real-life processing times.

Additionally, in contrast e.g. to the exponential distribution, we can control its variability by means of the standard deviation and thus model different degrees of variability of the instances in the testbed.

Regarding the average processing times of the jobs (μ_{ij}), each one is drawn from a discrete uniform [1,99] distribution. In the deterministic counterpart, these values are assumed to generate difficult instances (see e.g. Dannenbring, 1977, or Campbell et al., 1970), so we expect the same for the stochastic case. For each mean processing time, we want to model different degrees of variability. Setting different values for the standard deviation in an straightforward manner could be rather tricky and produce non realistic processing times. Therefore, we use different values of the coefficient of variation $c = \frac{\sigma}{\mu}$ to generate several instances. More specifically, $c \in \{0.01, 0.1, 0.2, 0.5, 1\}$ to capture the different scenarios of variability. When $c = 0.01$, the processing times are close to be deterministic, but for $c = 1$ the processing times for a job suffer a great variation. Although bigger c values are naturally possible, we believe that these do not represent reasonably realistic environments, as very large coefficient of variations are not the norm in industry due to the substantial efforts done to reduce the variability of the processing times via process automation and standardisation procedures. Note that the case $c = 1.0$ yields the same variance as in the exponential distribution, and, in order to make sure that we also cover this case, in Section 4 we also include results assuming an exponential distribution of the processing times.

In summary, we build each instance of the testbed in the following manner: For a given coefficient of variation c, we select a number of jobs and a number of machines (problem size). Next, for each job on each machine, we generate their mean processing time μ_{ij} by using an uniform [1,99] distribution. The standard deviation of each job on each machine σ_{ij} is then set to $c \cdot \mu_{ij}$. This procedure is repeated in order to generate 20 instances for each problem size and coefficient of variation. In total, 1,600 instances (320 for each value of c) are generated.

4 A procedure for the estimation of the expected makespan

As mentioned before, a critical issue in stochastic scheduling is how to evaluate the solutions. Recall that here the objective function is the expected makespan, therefore, given a sequence, an expected makespan must be assigned to this sequence. However, such expected makespan has to be estimated via a sample mean of the makespans corresponding to this sequence. The usual way to obtain the sample mean is to conduct a very large number of simulations of the makespan. Gourgand et al. (2003) assesed the accuracy of such simulations by making comparisons of the simulation results against those obtained by a Markov chain, and found that sample sizes of 200,000 produced 95% confidence intervals of the order of 0.1%. In their experiments, Baker and Altheimer (2012) use a sample size of 100,000 for problem sizes where $m \in \{2, 3, 6\}$ and n is up to 10 jobs whereas, for the 2-machine case, Portougal and Trietsch (2006) use a sample size of 10,000. Nevertheless, it is clear that the sample size should depend –at least– on two factors, namely the confidence interval of the results, and the stochasticity of the problem. Note that the way the solutions are evaluated may determine the significance of the results, here the criticality of this issue.

In this paper, we propose a method to estimate the expected makespan of a sequence based on the maximum percentual error accepted for the estimation of $E[C_{max}]$. More specifically, the half-width of a confidence interval for $E[C_{max}]$ of $1-\alpha$ confidence level is given by $t_{\alpha/2,N-1}\frac{s}{\sqrt{N}}$, where s is the sample standard deviation of the makespan, and $\alpha/2$ is the area of a Student's t-distribution with $N-1$ degrees of freedom left in the interval $(-\infty, t_{\alpha/2,N-1}]$ (see e.g. Vijay and Saleh, 2011). We intend that $t_{N-1,\alpha/2}\frac{s}{\sqrt{N}} \leq \bar{C}_{max} \cdot p$, where p is a (small) percentage. By doing so, $E[C_{max}]$ is confined in the interval $[\bar{C}_{max}(1-p), \bar{C}_{max}(1+p)]$ with a $1-\alpha$ confidence level. If we set α to a very low value (in our experiments, $\alpha = 0.001$), we can be almost sure (statistically speaking) that p represents the maximum relative error of the estimation of $E[C_{max}]$ and thus use p to check the significance of different results obtained by several heuristics. Note that the normality assumption of the sum of the sample values of makespan required to use this confidence interval seems a very loose restriction given the high number of samples (simulations) that would be run in practice, and the fact that the result of each run is independent from the others.

More specifically, our procedure for estimating the expected makespan of a sequence S consists of the following steps:

1. Set the simulations counter to zero, i.e. $N := 0$
2. Set the sum of makespans to zero, i.e. $SumC_{max} := 0$
3. Set the sum of squares of makespans to zero, i.e. $SumSC_{max} := 0$
4. do:
   (a) Run a simulation to obtain a sample makespan C_{max} of S.
   (b) Update the number of simulations, i.e. $N := N + 1$.
   (c) Update the sum of makespans, i.e. $SumC_{max} := SumC_{max} + C_{max}$
   (d) Update the sum of squares of makespans, i.e. $SumSC_{max} := SumSC_{max} + C^2_{max}$
   (e) Calculate $\bar{C}_{max} := \frac{SumC_{max}}{N}$ and $s := \sqrt{\frac{SumSC_{max} - N \cdot (\bar{C}_{max})^2}{N-1}}$.

   while $\frac{s \cdot t_{N-1,\alpha/2}}{\bar{C}_{max}\sqrt{N}} > p$
5. Return $\bar{C}_{max}$.

Note that, technically, we have to force the procedure to run more than one simulation. From the second simulation on, the percentual error oscillates until it is below the desired quantity p. Note also that setting unrealistic values of p and α may cause the loop to enter into a deadlock. We can be there sure (with a confidence level of $1 - \alpha$) that the percentage error in the estimation of the expected makespan is below p. In our experiments, $\alpha = 0.001$, so that means that we can be quite confident (99,9%) on the bounds of the error in the estimation.

In order to check the number of simulation runs required by this procedure for different degrees of variability and percentage error p, we obtain the estimations of the expected makespan of a random sequence for each instance in the testbed. The results are shown in Table 1.

From the results, it can be seen that, as foreseable, the number of runs varies greatly depending on the percentage error accepted. Allowing a 5% error means that results can be obtained with

$N \longleftarrow 0$; \%Set the simulation counter to zero
$\pi \longleftarrow \pi_1^{'}$;
for $k = 2$ **to** n **do**
    $r \longleftarrow {\pi_k}'$;
    Determine the values of e_{ij}, q_{ij} and f_{il} from Taillard's acceleration (see equations **??**, **??**, and **??**);
    Determine minimal makespan resulting from inserting job r in all possible positions of π;
    bp $\longleftarrow$First position where the makespan is minimal;
    tb $\longleftarrow$ Number of positions with minimal makespan (i.e. number of ties);
    ptb $\longleftarrow$ Array (of length tb) with the positions where the makespan is minimal;
    it_{bp} is the idletime corresponding to the bp and set to a very large number;
    if $tb > 1$ **and** $k < n$ **then**
        for $l = 1$ **to** tb **do**
            $it^{''} \longleftarrow 0$;
            if $ptb[l] = k$ **then**
                for $i = 2$ **to** m **do**
                    $it^{''} \longleftarrow it^{''} + f_{i,k} - e_{i,k-1} - t_{i,r}$;
                end
            else
                $f^{'}_{1,ptb[l]} \longleftarrow f_{1,ptb[l]} + p_{1,ptb[l]}$;
                for $i = 2$ **to** m **do**
                    $it'' \longleftarrow it^{''} + f_{i,ptb[l]} - e_{i,ptb[l]} + p_{i,ptb[l]} - t_{i,r} + \max\{0, f^{'}_{i-1,ptb[l]} - f_{i,ptb[l]}\}$;
                    $f^{'}_{i,ptb[l]} \longleftarrow \max\{f^{'}_{i-1,ptb[l]}, f_{i,ptb[l]}\} + p_{i,ptb[l]}$;
                end
            end
            if $it_{bp} > it^{''}$ **then**
                $bp \longleftarrow ptb[l]$;
                $it_{bp} \longleftarrow it^{''}$;
            end
        end
    end
    $\pi \longleftarrow$ Array obtained by inserting job r in position bp of π;
end

Figure 1: Our Tie-Breaking Method

		$p = 0.005$, several c values				$p = 0.01$, several c values				$p = 0.05$, several c values			
n	m	0.01	0.1	0.2	0.5	0.01	0.1	0.2	0.5	0.01	0.1	0.2	0.5
5	2	431975	445822	496753	2023060	107999	111461	124159	500291	4327	4465	4981	17707
5	5	432423	440565	469612	1357063	108111	110146	117412	324274	4332	4413	4707	12420
5	10	432690	437698	455579	1013767	108178	109431	113899	247341	4334	4385	4559	9438
5	20	432856	435632	445717	787848	108220	108914	111445	197201	4336	4364	4464	8418
10	2	432495	440274	469076	1387634	108129	110069	117269	333995	4332	4411	4701	12204
10	5	432679	437646	455673	1001419	108175	109416	113923	244879	4334	4384	4561	9358
10	10	432818	436128	447910	804226	108210	109040	111982	201065	4336	4369	4487	7671
10	20	432911	434916	442260	691900	108233	108734	110568	173496	4336	4357	4430	6724
15	2	432686	437973	457365	1052151	108177	109495	114367	263858	4334	4387	4579	9512
15	5	432793	436369	449396	870760	108204	109097	112343	215040	4335	4371	4501	7987
15	10	432870	435250	444120	725289	108223	108816	111031	181546	4336	4360	4447	7498
15	20	432949	434544	440394	641138	108243	108642	110100	158538	4337	4353	4411	6360
20	2	432788	436753	451444	949730	108202	109194	112862	239675	4335	4375	4521	8842
20	5	432856	435628	445883	776576	108219	108912	111467	194368	4336	4364	4466	7177
20	10	432914	434841	442099	689996	108234	108716	110532	172757	4337	4356	4430	6858
20	20	432968	434284	439143	613362	108248	108577	109790	154052	4337	4350	4399	6195
Avg.		432729	437145	453276	961620	108188	109291	113322	237649	4335	4379	4540	9023

Table 1: Estimation of the expected makespan for the lognormal testbed: Number of simulation runs required

		p=0.005	p=0.01	p=0.05
5	2	499372	124843	5016
5	5	466236	116973	4667
5	10	452409	113241	4533
5	20	443656	111020	4443
10	2	468839	116955	4696
10	5	452568	113256	4535
10	10	444945	111386	4457
10	20	440093	110088	4408
15	2	457536	114205	4591
15	5	446817	111795	4478
15	10	441605	110480	4424
15	20	438473	109658	4391
20	2	451430	112693	4520
20	5	443921	111069	4445
20	10	439961	110056	4407
20	20	437425	109399	4382
Average		451580	112945	4525

Table 2: Estimation of the expected makespan for the exponential testbed: Number of simulation runs required

less than 10,000 runs, but a 0.5% error requires more than 400,000 runs even for instances with small variability, and around 1,000,000 for $c = 0.5$.

In general, the need of more runs decreases with the number of machines, but remains relatively stable with respect to the number of jobs. This may speak for a compensation of the processing times of a job across the machines.

Additional experiments were carried out to establish the percentage error (p) induced when the number of simulation runs is considered fixed. To do so, we estimate $\bar{C}_{max}$ for a random sequence using 100,000 simulation runs, and calculate a confidence interval for $\alpha = 0.001$. By doing so, we are 'almost' sure than the makespan is contained in this interval, and then use the extreme values of this interval to obtain p. The results are shown in Table 3 and make clear that, while 100,000 simulation runs are an acceptable number for low variability, for moderate/high variability, this number of simulations leads to an average error over 10%, and that, in some instances, this error is as high as 80%.

The results show that it is difficult to be confident in the results obtained for the number of simulation runs employed in the literature. Estimation errors of less than %1 require more than

		CV=0.01	CV=0.1	CV=0.2	CV=0.5	CV=1.0
5	2	0.010	0.011	0.011	0.022	0.142
5	5	0.010	0.010	0.011	0.018	0.138
5	10	0.010	0.010	0.011	0.016	0.098
5	20	0.010	0.010	0.011	0.014	0.079
10	2	0.010	0.010	0.011	0.019	0.195
10	5	0.010	0.010	0.011	0.016	0.102
10	10	0.010	0.010	0.011	0.014	0.072
10	20	0.010	0.010	0.011	0.013	0.074
15	2	0.010	0.010	0.011	0.017	0.105
15	5	0.010	0.010	0.011	0.015	0.149
15	10	0.010	0.010	0.011	0.014	0.089
15	20	0.010	0.010	0.010	0.013	0.062
20	2	0.010	0.010	0.011	0.015	0.106
20	5	0.010	0.010	0.011	0.014	0.091
20	10	0.010	0.010	0.011	0.013	0.065
20	20	0.010	0.010	0.010	0.012	0.063
Avg.		0.010	0.010	0.011	0.015	0.102
Max.		0.010	0.011	0.011	0.026	0.815
Min.		0.010	0.010	0.010	0.012	0.037

Table 3: Percentual error in makespan estimation for N =100,000

200,000 runs for scenarios with medium/high variability. In addition, our method allows to know the accepted error of the estimations (in percentual terms) and therefore to assert the significance of the differences in the performance of solution procedures.

5 Comparison of heuristics

In this section, we carry out a computational study to establish the performance of different heuristics for the problem according to the procedure for the estimation of the expected makespan presented in the previous section. The heuristics tested are the following:

- The stochastic version of the NEH heuristic as described in Baker and Altheimer (2012). This heuristic is labelled SNEH.
- The stochastic version of the CDS/Talwar heuristic as described in Baker and Altheimer (2012). This heuristic is labelled SCDS/Talwar.

- The deterministic NEH heuristic applied using as data the mean processing times of the instances.

- The deterministic CDS/Talwar heuristic applied using as data the mean processing times of the instances.

- The deterministic NEH heuristic applied using as data the mean processing times of the instances, but using as initial order that given by the deterministic CDS/Talwar heuristic. This heuristic is labelles NEH-Talwar.

Note that, although the procedure for SNEH and SCDS/Talwar are identical to that of Baker and Altheimer (2012), the estimation of the expected makespan of the subsequences and that of the final sequence is carried out according to the procedure presented in Section 4 for $p = 0.01$. Analogously, in order to estimate the expected makespan given by the deterministic heuristics (NEH, CDS/Talwar, and NEH-Talwar), the sequence obtained is evaluated following the aforementioned procedure.

The testbed presented in Section 3 is solved using the five heuristics presented above. Tables 4 to 7 show the results obtained for different values of c. Apart from the average values obtained by the estimated makespan for each one of the heuristic (labelled as Avg. in the tables), the average percentage increase of the makespan of each heuristic with respect to that of SNEH is presented (labelled as Δ in the tables).

In these tables, we do not give information on the time required for each heuristic. Note that the deterministic heuristic are nearly instantaneous for the problem sizes tested (e.g. NEH is less than 0.01 seconds for the biggest instances), although for our purposes, we have to evaluate the so-obtained sequence (something not required when applying it for a real problem). Regarding the times required for the stochastic heuristic, they depend obviously on the problem size and on the value of c, ranging from 1300 seconds for $c = 0.01, n = 20, m = 20$ to 2000 seconds for $c = 0.5, n = 20, m = 20$. In total, the computational workload of the experiments contained in the tables can be measured in weeks of CPU time.

Several comments can be done on the results obtained:

- With respect to the heuristics specifically designed for the stochastic problem, there are significant differences in performance. This result contradicts those obtained by Baker and Altheimer (2012), who did not detect significant differences among them. It has to be noted that their way to estimate $E[C_{max}]$ the solution is based on a fixed number of simulation runs and that the number that they employed (100,000) has been proved to be unsufficient to establish consistent results for medium/large coefficient of variations. In addition, our testbed is of bigger size, a fact that may also explain some differences.

- The differences in performance between SNEH and SCDS/Talwar decrease with the variability of the testbed (from roughly 15% for $c = 0.01$ to about 6% for $c = 0.5$), but still are substantial for relatively large coefficient of variations. The explanation may lie in the fact that Talwar's rule is optimal for the exponential distribution, whose coefficient of variation is 1, and therefore its performance improves for instances where c is closer to that value.

- With respect to the deterministic heuristic tested, the best performance correspond to the NEH. It is interesting to note that CDS/Talwar is suppose to incorporate some stochastic considerations in their ordering, however these do not pay off, either as a simple heuristic, or as a starting order for the NEH heuristic.

- When comparing SNEH and NEH it may be seen that the differences are very small. In some cases, these differences are below the error accepted for p (1%). The conclusion is that, for a realistic ranges of variability in the flowshop, the application of the NEH to the mean processing times gives an extremely good estimation of the performance of their stochastic counterpart. If we note that SNEH is highly CPU-intensive, requiring a high number of simulations of all subsequences for all steps, it has to be questioned whether the effort in running SNEH pays off.

- Although there is no clear pattern, it seems that the differences in performance of all the heuristics decrease with the number of machines, a fact for which we do not have at the moment a fully convincing explanation.

Table 4: Comparative results of the different heuristics for $c = 0.01$.

		SNEH	SCDS/Talwar		NEH		CDS/Talwar		NEH-Talwar	
n	m	Avg.	Avg.	Δ	Avg.	Δ	Avg.	Δ	Avg.	Δ
5	2	309.848	348.360	13.056	309.978	0.041	348.367	13.059	310.171	0.103
5	5	477.448	550.888	15.905	477.552	0.024	550.878	15.903	477.373	-0.086
5	10	795.812	891.632	12.574	795.833	0.004	891.644	12.576	798.258	0.301
5	20	1372.339	1482.957	8.063	1374.269	0.148	1482.933	8.061	1374.786	0.167
10	2	571.929	632.638	11.423	572.181	0.046	632.649	11.426	572.615	0.136
10	5	720.947	866.235	20.268	721.875	0.119	866.224	20.266	716.499	-0.575
10	10	1049.066	1246.054	18.884	1053.062	0.384	1246.057	18.885	1052.709	0.348
10	20	1673.640	1902.705	13.662	1676.495	0.169	1902.713	13.662	1681.827	0.490
15	2	801.950	857.986	6.943	802.028	0.010	857.982	6.943	805.056	0.406
15	5	1008.644	1206.445	19.737	1008.840	0.017	1206.441	19.736	1013.747	0.536
15	10	1309.681	1574.132	20.336	1312.544	0.223	1574.124	20.335	1324.418	1.134
15	20	1965.180	2261.390	15.063	1981.237	0.833	2261.389	15.063	1972.786	0.375
20	2	1085.429	1155.937	6.582	1085.508	0.008	1155.929	6.582	1088.635	0.309
20	5	1224.466	1486.417	21.419	1228.891	0.368	1486.406	21.418	1240.408	1.305
20	10	1601.085	1939.413	21.186	1609.659	0.548	1939.435	21.187	1615.548	0.926
20	20	2258.262	2658.262	17.759	2269.505	0.501	2658.259	17.759	2274.953	0.746
				15.179		0.215		15.179		0.414

6 Conclusions

In this paper, we have addressed the problem of scheduling jobs in a flowshop when their processing times adopt a given distribution. Existing literature for the problem reveals that optimal solutions can be found only for very specific cases, so some heuristics with similar performance have been proposed for the general case. We first focus on the critical issue of estimating the expected makespan of a sequence, and found that, for instances with a medium/large variability (expressed as the coefficient of variation of the processing times of the jobs), the number of samples (simulation runs) used in the literature to estimate the expected makespan of a sequence may not be sufficient to derive conclusive results. We propose a procedure with a variable number of iterations that ensures that the error in the estimation of the expected makespan is bounded (with a very high probability) within a small percentage. Using this procedure, we test the main

Table 5: Comparative results of the different heuristics for $c = 0.1$.

		SNEH	SCDS/Talwar		NEH		CDS/Talwar		NEH-Talwar	
n	m	Avg.	Avg.	Δ	Avg.	Δ	Avg.	Δ	Avg.	Δ
5	2	291.782	325.107	11.687	293.834	0.772	325.137	11.698	295.252	1.175
5	5	475.155	532.005	12.307	478.174	0.631	532.057	12.317	476.358	0.225
5	10	782.803	860.982	10.386	788.699	0.763	862.039	10.538	790.445	0.971
5	20	1348.199	1439.980	6.817	1358.467	0.787	1440.195	6.832	1355.911	0.557
10	2	529.584	587.962	11.609	539.093	1.782	587.953	11.609	539.663	1.955
10	5	722.250	843.684	16.890	733.248	1.535	843.613	16.879	733.328	1.524
10	10	1071.549	1233.335	15.202	1086.796	1.465	1233.383	15.206	1087.909	1.536
10	20	1697.238	1873.897	10.362	1715.366	1.079	1873.586	10.344	1722.355	1.489
15	2	746.468	806.670	7.981	754.396	1.071	806.746	7.992	764.054	2.332
15	5	993.123	1171.510	18.038	1010.630	1.783	1171.513	18.041	1019.869	2.717
15	10	1352.968	1555.070	15.059	1371.496	1.386	1555.117	15.062	1378.176	1.874
15	20	2033.541	2269.854	11.609	2069.860	1.795	2271.566	11.697	2061.274	1.359
20	2	1005.958	1079.875	7.376	1016.039	0.998	1079.903	7.378	1024.590	1.865
20	5	1221.532	1440.011	17.865	1245.929	1.994	1440.785	17.929	1261.137	3.217
20	10	1647.583	1919.185	16.478	1682.258	2.108	1919.258	16.482	1687.571	2.416
20	20	2351.119	2686.962	14.339	2394.679	1.851	2686.998	14.341	2397.252	1.959
				12.750		1.362		12.771		1.698

Table 6: Comparative results of the different heuristics for $c = 0.2$.

		SNEH	SCDS/Talwar		NEH		CDS/Talwar		NEH-Talwar	
n	m	Avg.	Avg.	Δ	Avg.	Δ	Avg.	Δ	Avg.	Δ
5	2	322.603	352.795	9.441	325.613	1.004	352.802	9.441	327.488	1.520
5	5	562.439	613.089	9.259	566.252	0.688	613.133	9.264	564.659	0.384
5	10	934.795	1007.518	8.112	941.541	0.743	1009.800	8.384	941.668	0.746
5	20	1601.737	1697.290	5.986	1615.790	0.895	1697.146	5.978	1612.897	0.686
10	2	581.957	641.934	10.700	599.805	3.022	641.875	10.688	599.746	3.102
10	5	871.709	988.245	13.382	886.699	1.719	988.071	13.360	887.356	1.752
10	10	1334.107	1495.436	12.168	1352.889	1.447	1495.574	12.177	1354.049	1.503
10	20	2116.182	2291.745	8.248	2138.487	1.065	2291.817	8.252	2146.237	1.431
15	2	817.669	888.258	8.541	836.351	2.297	888.129	8.528	849.235	3.815
15	5	1197.086	1373.755	14.815	1222.018	2.101	1373.915	14.828	1231.843	2.933
15	10	1693.012	1899.637	12.271	1723.999	1.839	1899.451	12.259	1730.821	2.240
15	20	2589.667	2834.702	9.461	2636.059	1.798	2836.892	9.551	2627.271	1.450
20	2	1093.098	1185.651	8.490	1119.084	2.378	1185.546	8.480	1134.384	3.796
20	5	1467.477	1688.517	15.032	1502.591	2.381	1689.583	15.106	1521.990	3.696
20	10	2072.442	2350.040	13.380	2123.447	2.468	2350.523	13.403	2128.186	2.681
20	20	3025.734	3381.442	11.808	3082.084	1.859	3381.093	11.796	3084.835	1.949
Avg				10.693		1.731		10.718		2.105

Table 7: Comparative results of the different heuristics for $c = 0.5$.

n	m	SNEH Avg.	SCDS/Talwar Avg.	Δ	NEH Avg.	Δ	CDS/Talwar Avg.	Δ	NEH-Talwar Avg.	Δ
5	2	744.176	776.915	4.380	748.874	0.657	776.746	4.365	752.611	1.146
5	5	1529.894	1595.708	4.422	1537.664	0.520	1595.128	4.392	1533.510	0.224
5	10	2726.438	2842.900	4.478	2741.956	0.596	2848.047	4.689	2739.926	0.513
5	20	4807.027	4982.471	3.657	4836.312	0.613	4986.449	3.744	4834.410	0.561
10	2	1411.348	1496.158	6.206	1446.847	2.448	1496.660	6.243	1446.062	2.461
10	5	2579.011	2764.617	7.170	2610.984	1.251	2763.582	7.132	2613.878	1.326
10	10	4384.309	4678.766	6.734	4433.741	1.153	4684.903	6.870	4433.034	1.118
10	20	7312.968	7656.029	4.647	7357.016	0.616	7659.549	4.700	7373.893	0.843
15	2	1979.146	2095.786	5.837	2021.776	2.165	2095.632	5.814	2042.227	3.156
15	5	3683.235	3989.336	8.347	3737.412	1.467	3990.114	8.363	3756.730	2.026
15	10	5793.048	6186.216	6.805	5861.646	1.184	6189.823	6.863	5878.701	1.479
15	20	9561.606	10073.656	5.356	9661.007	1.039	10078.809	5.412	9646.360	0.881
20	2	2647.731	2821.413	6.605	2711.862	2.447	2819.596	6.535	2743.518	3.665
20	5	4527.154	4910.429	8.443	4590.192	1.394	4914.857	8.541	4632.402	2.332
20	10	7330.081	7883.050	7.532	7445.274	1.580	7882.278	7.528	7446.577	1.593
20	20	11624.512	12411.446	6.818	11730.277	0.911	12403.771	6.757	11745.989	1.044
Avg.				6.090		1.252		6.122		1.523

Table 8: Comparative results of the different heuristics for $c = 1.0$.

n	m	SNEH Avg.	SCDS/Talwar Avg.	Δ	NEH Avg.	Δ	CDS/Talwar Avg.	Δ	NEH-Talwar Avg.	
5	2	6266.040	6341.156	1.201	6290.683	0.417	6350.017	1.321	6303.186	0.5
5	5	14794.404	14998.899	1.432	14825.357	0.213	15011.184	1.529	14822.093	0.1
5	10	29495.484	29905.531	1.464	29539.868	0.158	29919.130	1.506	29527.008	0.1
5	20	55811.357	56727.477	1.631	56006.588	0.345	56705.085	1.604	56013.200	0.3
10	2	13280.157	13505.036	1.777	13399.913	0.841	13497.332	1.710	13391.054	0.8
10	5	27912.319	28581.464	2.392	28074.882	0.575	28571.105	2.334	28096.549	0.6
10	10	53943.962	55067.908	2.081	54019.570	0.141	55114.819	2.151	54034.985	0.1
10	20	98731.447	100432.899	1.710	99116.154	0.407	100587.218	1.857	99022.914	0.2
15	2	18628.113	18968.050	1.798	18732.248	0.565	18958.325	1.754	18819.655	1.0
15	5	43253.615	44324.220	2.494	43325.129	0.174	44268.685	2.381	43416.154	0.3
15	10	75215.521	77090.704	2.488	75648.411	0.556	77170.600	2.581	75667.356	0.5
15	20	139988.391	142683.498	1.916	140496.048	0.353	142720.286	1.949	140481.886	0.3
20	2	25754.094	26255.813	1.981	25897.427	0.584	26234.195	1.891	26025.904	1.0
20	5	53797.160	55212.357	2.618	53938.933	0.273	55249.351	2.688	54102.808	0.5
20	10	101187.069	103972.650	2.752	101715.555	0.530	103915.201	2.684	101785.109	0.5
20	20	179560.039	183853.265	2.433	180249.784	0.402	184245.565	2.638	180325.990	0.4
Avg.				2.010		0.408		2.036		0.5

heuristic proposed in the literature in a larger testbed and found significant differences in their performance, in contrast with existing studies. We also found that the deterministic counterpart of the most efficient heuristic for the stochastic problem performs extremely well for most settings, which indicates that (at least within the limitations of our study), a practical way to solve the $Fm|prmu|E[C_{max}]$ is to simplify it to its deterministic version.

Several issues lie ahead as future research lines. First of all, the computational analysis can be enhanced by introducing different families of distribution, higher coefficients of variation for the lognormal distribution, and bigger problem instances. However, it is to note that some families of distributions widely applied in theory due to their good properties (such as e.g. the exponential) do not match well with the processing times distributions found in practice. Analogously, very large coefficient of variations are not the norm in industry, as discussed before. Finally, there are severe computational limitations for using bigger testbeds, as the experiments carried out in this paper already amount for weeks of CPU time.

A more fruitful research line may be to check whether the results obtained by the best stochastic heuristic (SNEH) is close to the optimal values, or not. Although this neccessarily has to be done for small instances, it would be interesting to confirm the results by Baker and Altheimer (2012), who indicate that SNEH obtains close-to-optimal solutions. If this was not the case, ground for new approximate solutions for the problem may be re-opened.

References

Baker, K. and Altheimer, D. (2012). Heuristic solution methods for the stochastic flow shop problem. *European Journal of Operational Research*, 216(1):172–177.

Baker, K. and Trietsch, D. (2011). Three heuristic procedures for the stochastic, two-machine flow shop problem. *Journal of Scheduling*, 14(5):445–454.

Campbell, H., Dudek, R., and Smith, M. (1970). Heuristic algorithm for the n job, m machine sequencing problem. *Management Science*, 16(10):630–637.

Cunningham, A. and Dutta, S. (1973). Scheduling jobs with exponentially distributed processing times on two machines of a flow shop. *Naval Research Logistics Quarterly*, 16(1):69–81.

Dannenbring, D. G. (1977). Evaluation of flow-shop sequencing heuristic. *Management Science*, 23(11):1174–1182.

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

Gourgand, M., Grangeon, N., and Norre, S. (2003). A contribution to the stochastic flow shop scheduling problem. *European Journal of Operational Research*, 151(2):415–433.

Hopp, W. and Spearman, M. (2008). *Factory Physics (Third Edition)*. Irwin, Chicago.

Johnson, S. (1954). Optimal two- and three-stage production schedules with setup times included. *Naval Research Logistics Quarterly*, 1(1):61–68.

Kalczynski, P. and Kamburowski, J. (2006). A heuristic for minimizing the expected makespan in two-machine flow shops with consistent coefficients of variation. *European Journal of Operational Research*, 169(3):742–750.

Makino, T. (1965). On a scheduling problem. *Journal of the Operations Research Society of Japan*, 8:32–44.

Nawaz, M., Enscore Jr., E., and Ham, I. (1983). A heuristic algorithm for the m-machine, n-job flow-shop sequencing problem. *Omega*, 11(1):91–95.

Portougal, V. and Trietsch, D. (2006). Johnson's problem with stochastic processing times and optimal service level. *European Journal of Operational Research*, 169(3):751–760.

Reza Hejazi, S. and Saghafian, S. (2005). Flowshop-scheduling problems with makespan criterion: A review. *International Journal of Production Research*, 43(14):2895–2929.

Ruiz, R. and Maroto, C. (2005). A comprehensive review and evaluation of permutation flowshop heuristics. *European Journal of Operational Research*, 165(2):479–494.

Talwar, P. (1967). A note on sequencing problems with uncertain job times. *Journal of Operations Research Society of Japan*, 9:93–97.

Vijay, K. R. and Saleh, A. E. (2011). *An Introduction to Probability and Statistics (Second Edition)*. John Wiley and Sons.