Proceedings of the 2016 Winter Simulation Conference
T. M. K. Roeder, P. I. Frazier, R. Szechtman, E. Zhou, T. Huschka, and S. E. Chick, eds.

COMBINING SIMULATION WITH METAHEURISTICS IN DISTRIBUTED SCHEDULING PROBLEMS WITH STOCHASTIC PROCESSING TIMES

Laura Calvet
Angel A. Juan

Computer Science Department
Open University of Catalonia - IN3
Carl Friedrich Gauss Avenue
Castelldefels 08860, SPAIN

Victor Fernandez-Viagas
Jose M. Framinan

School of Engineering
University of Seville
Descubrimientos Avenue
Seville 41092, SPAIN

ABSTRACT

In this paper, we focus on a scenario in which a company or a set of companies conforming a supply network must deliver a complex product (service) composed of several components (tasks) to be processed on a set of parallel flow-shops with a common deadline. Each flow-shop represents the manufacturing of an independent component of the product, or the set of activities of the service. We assume that the processing times are random variables following a given probability distribution. In this scenario, the product (service) is required to be finished by the deadline with a user-specified probability, and the decision-maker must decide about the starting times of each component/task while minimizing one of the following alternative goals: *(a)* the maximum completion time; or *(b)* the accumulated deviations with respect to the deadline. A simheuristic-based methodology is proposed for solving this problem, and a series of computational experiments are performed.

1 INTRODUCTION

The complexity of manufacturing has become increasingly higher due to the rise of supply chains where different companies co-operate to manufacture a product for a final customer in a distributed manner. These supply networks naturally appear as companies, in their fierce global competition, try to identify their core competences and outsource/purchase those activities/services in which they do not excel. As a result, these products (or services, as these words will be used interchangeably throughout the paper) can be decomposed into a set of independent components/tasks –each one to be manufactured in a different facility– with a common due date or deadline (Figure 1). In this paper, we assume that each of these components/tasks has to be processed in a factory, which can be modeled as a permutation flow-shop problem with random or stochastic processing times (PFSPST).

In each factory k of a set F (with $k \in \{1,...,f\}$), a set J_k of n jobs has to be processed by a set M of m machines, being T_{ijk} (abbreviated to T_{ij} when it does not lead to confusion) the random variable representing the time it takes for job i of factory k to be processed by machine j (Figure 2). The PFSPST goal is to find a sequence (permutation) of jobs that optimizes a given criterion (taking into account that all jobs are processed by all machines in the order defined by the selected permutation). The most employed criterion in the scientific literature is the minimization of the expected maximum completion time or expected makespan, i.e., the average time it requires to process all the jobs throughout all the machines for the selected permutation.

Additionally, the global product is required to be finished by the deadline with a user-specified probability, p_0 (from now on, we will assume that the global product is finished if, and only if, all its components/tasks are finished). In this context, the decision-maker must decide about the starting times of each component, while

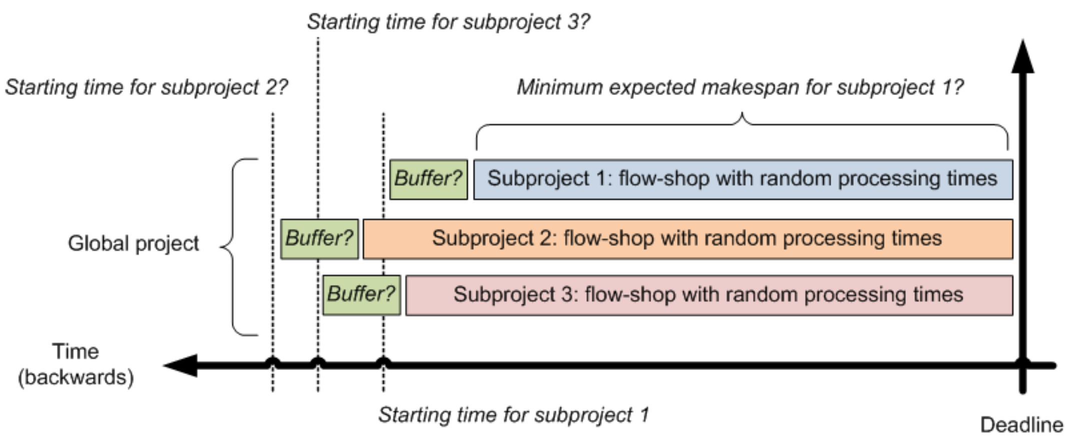
Starting time for subproject 3?
Starting time for subproject 2?
Minimum expected makespan for subproject 1?
Buffer?
Subproject 1: flow-shop with random processing times
Global project
Buffer?
Subproject 2: flow-shop with random processing times
Buffer?
Subproject 3: flow-shop with random processing times
Time (backwards)
Starting time for subproject 1
Deadline

Figure 1: A product requiring 3 independent flow-shops with a common deadline.

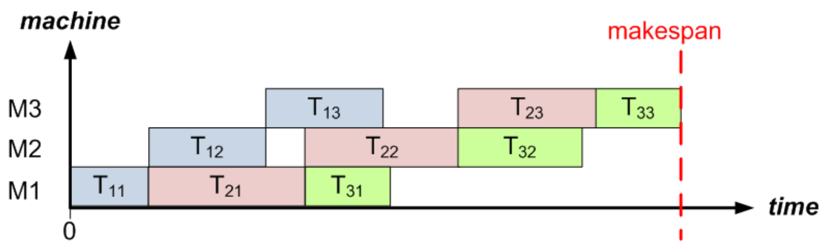

Figure 2: Example of a PFSP with 3 jobs and 3 machines.

minimizing one of the following alternative goals: *(a)* the maximum completion time of all components (machine-booking goal), i.e., the sum of all expected makespans and buffers in the product; or *(b)* the accumulated deviations with respect to the deadline (synchronization goal).

Our approach for solving this stochastic combinatorial optimization problem is based on the use of a simheuristic algorithm combining simulation (Monte Carlo in this case) with a metaheuristic framework. Some of the main benefits of our approach are the following ones: *(i)* it does not make any assumption on the size of the instances, i.e., being based on a metaheuristic framework it can solve large-scale instances in reasonable computing times; and *(ii)* it does not make any assumption either on the probability distributions employed to model the random processing times, i.e., being based on simulation there is no need to assume normality of processing times –any probability distribution can be used instead.

The rest of the paper is organized as follows: Section 2 provides a review of related work. Section 3 describes the main ideas behind our simulation-optimization approach. Some numerical experiments are carried out in Section 4, and analyzed in Section 5. Finally, we conclude this paper by summarizing its main findings and discussing future work in Section 6.

2 RELATED WORK

Different streams of the literature are related to the problem under consideration, namely assembly scheduling, distributed flow-shop scheduling, and permutation flow-shop scheduling with stochastic processing times. These are discussed in the next subsections.

2.1 Assembly scheduling

This problem is also denoted *n-stage assembly* or *assembly flow-shop scheduling*. In this problem, *m* tandem lines are arranged prior to a single assembly station which is fed by the tandem lines. Using this layout, *n* different products (jobs) have to be manufactured, each one consisting of *m* components manufactured in the tandem lines. The processing time of each component in each line is different. Some authors distinguish among the *fixed* case (i.e., each component can be processed only in a given tandem line), and the *unfixed* case (i.e., each component can be processed in different factories).

For these problems, different objectives are sought, such as makespan minimization (Sung and Juhn 2009), total flow-time (Al-Anzi and Allahverdi 2013, Sung and Kim 2008), due date fulfillment (Al-Anzi and Allahverdi 2007), or the combination of several indicators (Seidgar et al. 2014).

Most references refer to the 2-stage case (production followed by assembly), so they assume that each tandem line consists of a single machine. The underlying hypothesis is that there is a single processing time for each component before the assembly process. For this problem, different exact and approximate methods have been proposed, and some variants of the original problem have been tackled by Sung and Juhn (2009), where two types of components –manufactured and imported– are considered, and by Liao et al. (2015), where assembly batches are assumed.

Several other variants of the problem for three stages have been addressed in the literature (see e.g. Koulamas and Kyparisis 2001), but in none of the different versions of the problem the processing times have been assumed to be stochastic.

2.2 Distributed Flow-shop Scheduling Problem (DFSP)

This problem has several similarities with the one presented in this communication: there are *m* identical permutation flow-shops where *n* jobs have to be processed. However, in the DFSP the jobs have not been assigned to each flow-shop, so this assignment becomes part of the decision problem. Furthermore, the DFSP has not been addressed with objectives related with due dates. In fact, the only objective addressed so far refers to makespan minimization (Naderi and Ruiz 2010), for which the best available heuristic is due to Fernandez-Viagas and Framinan (2015).

A particular case of the DFSP refers to the so-called *Distributed Assembly Flow-shop Scheduling Problem*, which combines the DFSP with assembly scheduling. In this problem, a distributed flow-shop composed of *k* identical flow-shops is followed by a single assembly operation. *n* jobs consisting each one of *k* components have to be assembled after each component has been manufactured in one of the flow-shops. This decision problem includes job assignment plus the scheduling of jobs in the assembly line. The main references for this problem are Hatami et al. (2015) and Hatami et al. (2013). In the first reference, the authors consider the objective of makespan minimisation, while in the second sequence-dependent setup times are assumed. In both cases, the problem is addressed using approximate algorithms, and, as in the assembly scheduling problems, we are not aware of references dealing with stochastic processing times.

2.3 Permutation Flow-shop Problem with Stochastic Processing Times

While the PFSP has been intensively studied during the last few decades, the PFSPST has received less attention. Baker and Trietsch (2011) designed heuristics for addressing the 2-machine PFSPST, where the processing times are independent random variables following specific probability distributions. Later, Baker and Altheimer (2012) presented a methodology for the *m*-machine version. In addition, several variations of the PFSPST have been analyzed. For instance, Allaoui et al. (2006) and Choi and Wang (2012) worked on the stochastic hybrid flow-shop scheduling problem, aiming to minimize the expected makespan. The same problem was tackled by Kianfar et al. (2012) with the goal of minimizing the average tardiness of jobs. A novel approach is applied in Zhou and Cui (2008) for tackling the multi-objective stochastic PFSP, where both the flow-time and delay time of jobs are minimized.

An interesting line is related to uncertainty. Basically, there are two categories: proactive (or robust) scheduling and reactive scheduling. Works falling in the first category (Roy 2010) propose constructing an original predictive schedule. The basic aim is to find schedules that do not require new schedules (or significant changes) when confronting disruptions. These works may consider probability distributions or sets of scenarios. Al Kattan and Maragoud (2008), Ghezail et al. (2010) and Liu et al. (2011) addressed the PFSP with uncertainty implementing proactive scheduling strategies. On the other hand, reactive scheduling consists in revising and re-optimizing schedules when unexpected events take place. A classical option is to obtain a predictive scheduling and then try to repair it according to the actual state of the system. A comprehensive review on rescheduling under disruptions is provided by Katragjini et al. (2013).

Some authors employ exact methods for addressing the PFSPST. A disadvantage of many of these methods is that they only work with a specific set of probability distributions and relatively small instances. Moreover, it may be difficult to adapt them for handling dependencies among processing times. Simulation techniques enable researchers to deal with these situations in a natural way. An interesting example is the work of Baker and Altheimer (2012), which proposed a hybrid approach combining heuristics and simulation. The authors tested three heuristic methods: two relying on the CDS heuristic (Campbell et al. 1970) and one on the NEH heuristic (Nawaz et al. 1983).

3 PROPOSED METHODOLOGY

Our methodology is based on a simheuristic approach (Juan et al. 2015), which relies on the Iterated Local Search (ILS) metaheuristic (Lourenço et al. 2010) and Monte Carlo simulation (MCS) techniques. This metaheuristic has been successfully applied to a wide range of combinatorial optimization problems and is highly popular among researchers. Due to its modularity, it is relatively easy to implement. On the other hand, MCS techniques enable the assessment of solutions in a dynamic environment by taking the following steps: *(1)* simulate a number of scenarios, where each one is created by generating one value per random variable; *(2)* apply a specific solution in each scenario and compute a measure of performance; and *(3)* calculate the average performance.

The methodology is summarized in Algorithm 1 and described next. The inputs are the set F, the deadline (d_0), the probability p_0, and the processing time of each trio of job, machine, and component ($T_{ijk} \forall i \in J, j \in M$ and $k \in F$), which is assumed to follow a specific probability distribution (either theoretical or empirical). Initially, the number of components (*nComponents*) is obtained. Then, a set of instructions are performed for each component:

1. calculate the associated probability (p_{0k}) in order to reach p_0. p_{0k} is computed as $p_0^{1/\text{nComponents}}$ (note we assume that the components are independent);
2. solve the corresponding PFSPST (i.e., get the 'best' permutation);
3. employ MCS techniques to get a sample of makespans (of size r) for the solution obtained and use it to build an empirical cumulative distribution function;
4. identify the makespan below which p_{0k} percent of the observations may be found (i.e., the p_{0k} percentile); and
5. set the starting time as d_0 minus the p_{0k} percentile and store it.

Once all starting times are computed, the procedure returns them. In this work we define 'best' permutation (*bestSol*) as the one minimizing the expected makespan.

Regarding steps 3 and 4, we will provide more details. There is a sample of makespans $x_1, x_2, ..., x_r$, which are observations of independent and identically distributed real random variables with a common cumulative distribution function $F(t)$, which is unknown. The empirical cumulative distribution function is defined as:

$$\hat{F}_r(t) = \frac{\text{number of elements in the sample } \leq t}{r} = \frac{1}{r}\sum_{i=1}^{r} 1_{x_i \leq t} \qquad (1)$$

By the strong law of large numbers, the estimator $\hat{F}_r(t)$ converges to $F(t)$ as almost surely, for every value of t. As a consequence, the estimator $\hat{F}_r(t)$ is consistent. Thus, we can obtain the value t ensuring that the component k will have a makespan of t or lower with a probability of p_{0k} by computing: $t = \hat{F}_r^{-1}(p_{0k})$.

Algorithm 1

```
1:  procedure DistributedFlowshopScheduling($F, d_0, p_0, T_{ijk}$)
        $d_0$: product deadline (common due date for all components)
        $p_0$: user-specified probability that the product finishes on or before $d_0$
        $T_{ijk}$: random processing time of job $i$ in machine $j$ for component $k$
2:      $nComponents \leftarrow$ getNumberComponents($F$)
3:      for each component $k$ in $F$ do
4:          $p_{0k} \leftarrow$ calcProbabilityComponent($p_0, k, nComponents$)
5:          $bestSol(k) \leftarrow$ solvePFSPST($k, T_{ijk}$)          ▷ use a simulation-optimization algorithm
6:          $distFunction(k) \leftarrow$ calcDistFuntion($bestSol(k)$)          ▷ use observations from simulation
7:          $percentile(p_{0k}) \leftarrow$ calcPercentil($distFunction(k), p_{0k}$)
8:          $startingTime(k) \leftarrow d_0 - percentil(p_{0k})$
9:          $startingTimes \leftarrow$ add($startingTime(k)$)
10:     end for
11:     return $startingTimes$
12: end procedure
```

In order to solve each of the PFSPST (i.e., step 4), we present a simheuristic methodology similar to the one described in Juan et al. (2014). It relies on two components: an ILS metaheuristic that searches promising solutions, and MCS techniques for assessing them. Figure 3 shows the basic scheme. The first steps are to transform the original problem instance into a PFSP instance replacing random variables by their means, and apply the aforementioned NEH heuristic for finding a solution (*baseSol*). Then, MCS techniques are used to compute the expected makespan for that solution considering the stochastic environment described by the original instance. A new solution (*bestSol*) is created to store the best solution found, which is a copy of *baseSol* at this stage. Afterwards, the loop is started, which will execute instructions during *maxTime* seconds. The first step is building a new solution (*newSol*) by perturbing *baseSol*, which implies selecting randomly two jobs and interchanging their positions. A local search based on the classical shift-to-left movement is applied to each job in a random order. The next step checks whether the makespan of *newSol* is lower than or equal to that of *baseSol*. If this is not satisfied, the solution is discarded and another iteration starts. Otherwise, the expected makespan (obtained using MCS techniques) of *newSol* is computed. A variable *delta* contains the difference between the expected makespan of *newSol* and *baseSol*. If *delta* is negative or 0 (i.e., there is an improvement) *newSol* replaces *baseSol* and the variable *credit* (which is initially 0) is reset to $-delta$. Additionally, *bestSol* is updated if its expected makespan is higher than the one of *baseSol*. If *delta* is positive but equal to or lower than *credit* (i.e., it is only ‘slightly’ worse), *baseSol* is updated and *credit* is reset to 0. This acceptance criterion that chooses whether *baseSol* is updated is known as Demon-like process (Talbi 2009) and is employed to help avoiding local minima during the execution of the algorithm. Finally, *bestSol* is returned. It is important to note the difference between the methodology described in Juan et al. (2014) and ours: while the former is deterministic-driven (i.e., the replacement of the base solution depends on the makespan of the transformed instance), the latter is stochastic-driven (i.e., stochastic makespans are employed). This modification provides better solutions without requiring more computational time.

4 COMPUTATIONAL EXPERIMENTS

The methodology presented has been implemented as a Java application. A standard personal computer, Intel Core i5 CPU at 3.2 GHz and 4 GB RAM with Windows 7 has been employed. We have experimented with 4 instances with the parameters $f \in \{2,4\}$ and $m \in \{5,10\}$, and with 20 jobs assigned to each line $k \in F$. The processing times on each machine are taken from the PFSP instances introduced in Taillard (1993). Table 1 describes the composition of the new instances and their main characteristics (d_0, m, and

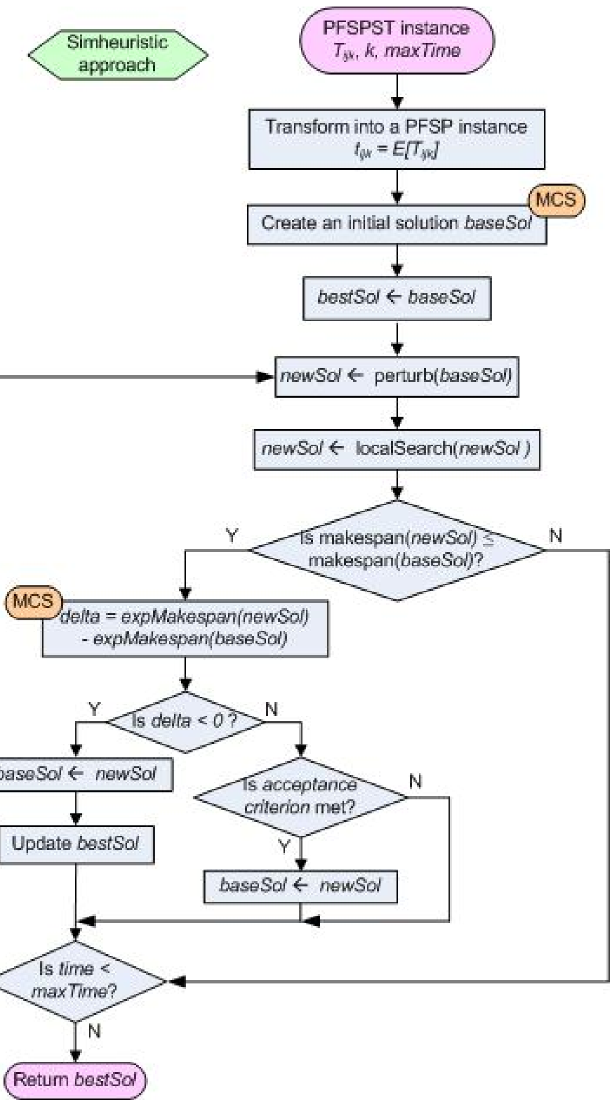
Simheuristic approach
PFSPST instance
T_{ijk}, k, *maxTime*
Transform into a PFSP instance
$t_{ijk} = E[T_{ijk}]$
MCS
Create an initial solution *baseSol*
bestSol ← *baseSol*
newSol ← perturb(*baseSol*)
newSol ← localSearch(*newSol*)
Is makespan(*newSol*) ≤ makespan(*baseSol*)?
Y
N
MCS
delta = *expMakespan*(*newSol*) - *expMakespan*(*baseSol*)
Is *delta* < 0 ?
Y
N
baseSol ← *newSol*
Update *bestSol*
Is acceptance criterion met?
N
Y
baseSol ← *newSol*
Is *time* < *maxTime*?
Y
N
Return *bestSol*

Figure 3: Flowchart of the proposed methodology for the PFSPST.

f) in the third, fourth and fifth columns, respectively. Second column indicates the instance of Taillard (1993) from which the processing times are considered. The common due date has been generated by doing $d_0 = h \cdot C$, following a similar procedure to the ones described by Della Croce et al. (2000) and Biskup and Feldmann (2001), where h is a parameter to indicate how loose/tight the common due date is and C is the makespan of the line, i.e., the highest makespan among all the factories. Note that the makespan of each line is the best known solution of the corresponding Taillard instance which is recorded in http://mistic.heig-vd.ch/. To avoid unfeasible instances, we generate loose common due dates according to a uniform distribution [1.2, 1.6] for the parameter h. All experiments have been run 5 times with different seeds and only the best values are stored. The computational time allowed for each instance is the sum of the time assigned to each production line, which is $0.03 \cdot m \cdot n$ seconds (as suggested in Juan et al. 2014). For introducing the stochasticity of processing times, each one is defined as an independent random variable following a logNormal distribution with a mean (μ) set to the processing time of the original instance and a standard deviation (σ) obtained from the coefficient of variation ($c = \sigma/\mu$). Three values of c are tested: 0.1, 0.5, and 1 (proposed in Framinan and Perez-Gonzalez 2015), which represent a low, medium, and high level of stochasticity, respectively. Regarding the user-given probabilities, three levels are established: 0.8, 0.9, and 0.95.

First, we compare the performance of our algorithm following a stochastic approach (the described in the previous section) with a deterministic one, which assumes zero variability. Table 2 displays the booking times (i.e., the sum of all expected makespans and buffers, or percentiles) for the first instance considering each one of the 9 scenarios defined by the probabilities and the levels of stochasticity. The mean gaps between the booking times of both approaches for each level of stochasticity (from low to high) are: -0.12%, -1.13%, and -1.77%, respectively. Similarly, for each level of probability (from low to high), the mean gaps are: -0.89%, -0.99%, and -1.16%, respectively. Secondly, we analyze the booking times and buffers for the same 9 scenarios studying all instances. Solutions are provided in Table 3. Figure 4 shows this information in boxplots. Finally, the relation between booking times and level of stochasticity is studied for a high number of probability values. The patterns identified for the first instance are presented in Figure 5.

Table 1: Description of the instances.

Instance	Taillard's instances	d_0	m	f
ins1	ta001 ($f=1$), ta002 ($f=2$)	1841	5	2
ins2	ta003 ($f=1$), ta004 ($f=2$), ta005 ($f=3$), ta006 ($f=4$)	1620	5	4
ins3	ta011 ($f=1$), ta012 ($f=2$)	2409	10	2
ins4	ta013 ($f=1$), ta014 ($f=2$), ta015 ($f=3$), ta016 ($f=4$)	2372	10	4

Table 2: Booking times for "ins1" following the stochastic and the deterministic approach.

		Stochastic approach			Deterministic approach			
	Stochasticity	Low	Medium	High	Low	Medium	High	Average
Probability	0.8	2671.96	2845.83	3086.30	2674.66	2876.29	3133.19	2881.37
	0.9	2688.03	2944.64	3327.98	2691.21	2977.94	3386.28	3002.68
	0.95	2701.60	3034.82	3571.91	2705.70	3072.52	3648.68	3122.54
	Average	2687.19	2941.76	3328.73	2690.52	2975.58	3389.38	

Table 3: Table of results.

		Level of stochasticity					
		Low		Medium		High	
		Booking time	Buffer	Booking time	Buffer	Booking time	Buffer
ins1	0.8	2671.96	24.95	2845.83	104.23	3086.30	204.88
	0.9	2688.03	41.02	2944.64	203.04	3327.98	446.55
	0.95	2701.60	54.59	3034.82	293.22	3571.91	690.49
ins2	0.8	4867.96	38.18	5287.05	215.53	5821.46	445.30
	0.9	4896.62	66.85	5515.96	444.43	6346.50	970.34
	0.95	4923.97	94.20	5736.55	665.02	6879.40	1503.25
ins3	0.8	3270.41	13.66	3478.39	98.77	3786.69	221.03
	0.9	3282.55	25.80	3579.86	200.24	4041.78	476.12
	0.95	3293.67	36.91	3680.14	300.53	4308.61	742.95
ins4	0.8	5748.58	29.41	6069.68	163.80	6544.70	358.62
	0.9	5774.13	54.96	6251.65	345.78	7012.24	826.16
	0.95	5797.32	78.15	6427.30	521.42	7485.07	1298.99

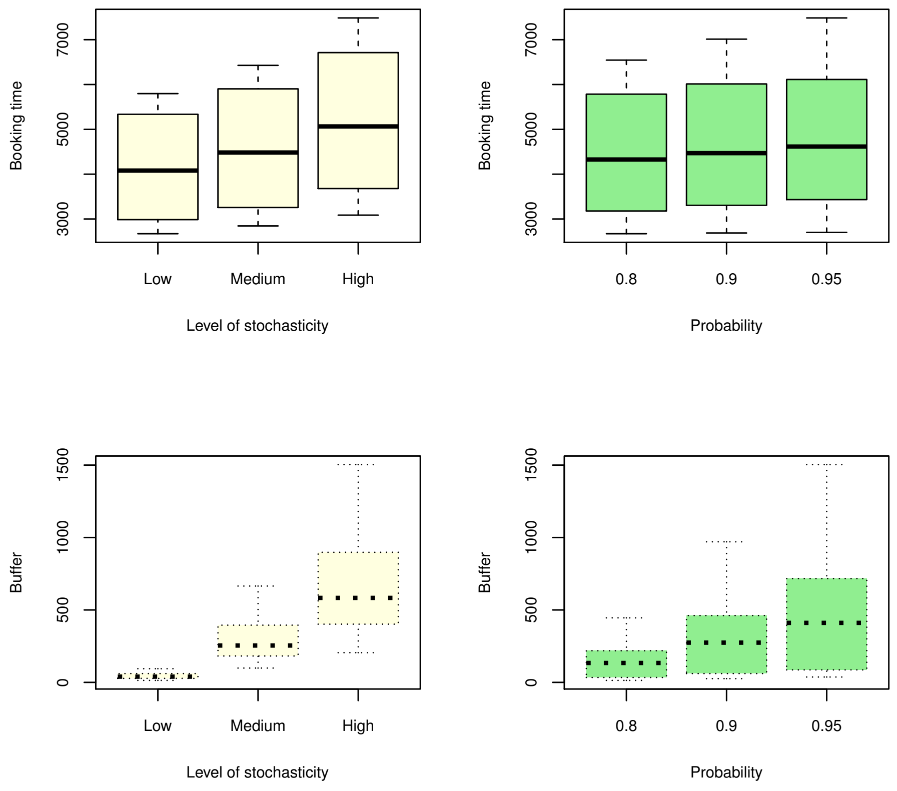

Figure 4: Boxplots of booking times and buffers for several scenarios.

5 ANALYSIS OF RESULTS

It is important to start with the comparison between the stochastic and the deterministic approach, in order to check whether introducing simulation in the solving methodology for the PFSPST is useful. The results (Table 2) indicate that the improvement in terms of booking times is highly significant, with gaps increasing with the levels of stochasticity and probability.

Focusing on the effect of these variables (stochasticity and probability) on booking times, it can be observed (Table 3 and Figure 5) that the booking times increase with both the stochasticity and the probability. While the differences among levels of stochasticity are relatively small for low probabilities (below 0.8), they rapidly increase for higher values. As expected, when users require a high probability, the booking time increases exponentially for all the levels of stochasticity considered.

According to the graphical results in Figure 4, which refer to all instances, both variables have a positive effect on the booking time, being the level of variability more relevant. Regarding the buffers required, they are much more volatile. For a low level of stochasticity, it is very small but rapidly increases as the level grows. The effect of the probability is a bit smaller but also positive and growing.

6 CONCLUSIONS AND FURTHER RESEARCH

In this paper we have discussed how simulation can be combined with metaheuristics in order to deal with stochastic multi-factory scheduling problems. In particular, we have studied a scheduling problem composed of parallel and independent components/tasks with a common due date, the processing of each

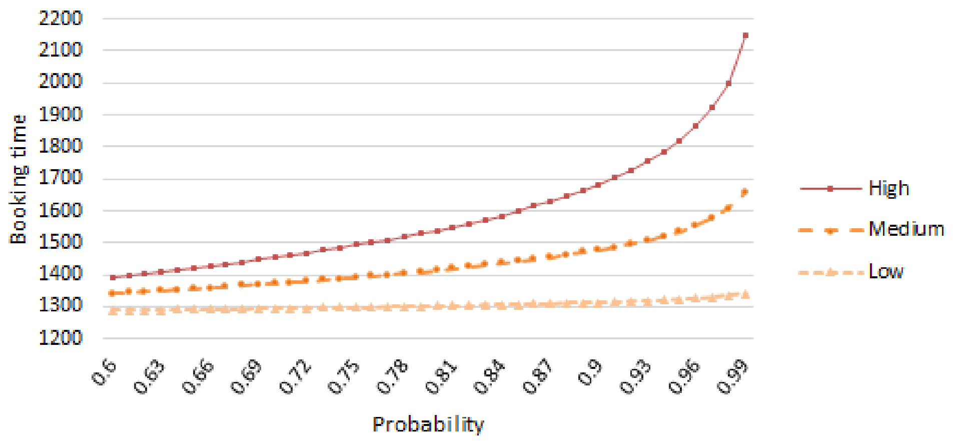

Figure 5: Relation between probability, booking time and level of stochasticity for "ins1".

of these components being modeled as a permutation flow-shop problem with stochastic processing times. In this context, a natural question arises: how to set starting times of each component in such a way that the total machine-occupancy time is minimized while ensuring a user-given probability of finishing all components in due time. The computational experiments carried out in this paper show how starting times vary according to factors such as the variance level in the random processing times or the user-required probability threshold for the product to finish on time. Other similar questions can be formulated, e.g., how to set starting times of each component in order to minimize total deviations from the common due date while respecting the aforementioned constraint, etc. These are open research lines that must require from similar approaches to the one introduced here.

ACKNOWLEDGMENTS

This work has been partially supported by the Spanish Ministry of Economy and Competitiveness (grants TRA2013-48180-C3-P, DPI2013-44461-P), FEDER, and the Catalan Government (2014-CTP-00001).

REFERENCES

Al-Anzi, F., and A. Allahverdi. 2007. "A Self-adaptive Differential Evolution Heuristic for Two-stage Assembly Scheduling Problem to Minimize Maximum Lateness with Setup Times". *European Journal of Operational Research* 182 (1): 80–94.

Al-Anzi, F., and A. Allahverdi. 2013. "An Artificial Immune System Heuristic for Two-stage Multi-machine Assembly Scheduling Problem to Minimize Total Completion Time". *Journal of Manufacturing Systems* 32 (4): 825–830.

Al Kattan, I., and R. Maragoud. 2008. "Performance Analysis of Flowshop Scheduling Using Genetic Algorithm Enhanced with Simulation". *International Journal of Industrial Engineering: Theory, Applications and Practice* 15 (1): 62–72.

Allaoui, H., S. Lamouri, and M. Lebbar. 2006. "A Robustness Framework for a Stochastic Hybrid Flow Shop to Minimize the Makespan". In *International Conference on Service Systems and Service Management*, Volume 2, 1097–1102. IEEE.

Baker, K. R., and D. Altheimer. 2012. "Heuristic Solution Methods for the Stochastic Flow Shop Problem". *European Journal of Operational Research* 216 (1): 172–177.

Baker, K. R., and D. Trietsch. 2011. "Three Heuristic Procedures for the Stochastic, Two-machine Flow Shop Problem". *Journal of Scheduling* 14 (5): 445–454.

Biskup, D., and M. Feldmann. 2001. “Benchmarks for Scheduling on a Single Machine Against Restrictive and Unrestrictive Common Due Dates”. *Computers & Operations Research* 28 (8): 787–801.

Campbell, H. G., R. A. Dudek, and M. L. Smith. 1970. “A Heuristic Algorithm for the n Job, m Machine Sequencing Problem”. *Management science* 16 (10): B630–B637.

Choi, S., and K. Wang. 2012. “Flexible Flow Shop Scheduling with Stochastic Processing Times: A Decomposition-based Approach”. *Computers & Industrial Engineering* 63 (2): 362–373.

Della Croce, F., J. N. Gupta, and R. Tadei. 2000. “Minimizing Tardy Jobs in a Flowshop with Common Due Date”. *European Journal of Operational Research* 120 (2): 375–381.

Fernandez-Viagas, V., and J. M. Framinan. 2015. “A Bounded-search Iterated Greedy Algorithm for the Distributed Permutation Flowshop Scheduling Problem”. *International Journal of Production Research* 53 (4): 1111–1123.

Framinan, J. M., and P. Perez-Gonzalez. 2015. “On Heuristic Solutions for the Stochastic Flowshop Scheduling Problem”. *Journal of Operational Research* 246 (2): 413–420.

Ghezail, F., H. Pierreval, and S. Hajri-Gabouj. 2010. “Analysis of Robustness in Proactive Scheduling: A Graphical Approach”. *Computers & Industrial Engineering* 58 (2): 193–198.

Hatami, S., R. Ruiz, and C. Andrés-Romano. 2013. “The Distributed Assembly Permutation Flowshop Scheduling Problem”. *International Journal of Production Research* 51 (17): 5292–5308.

Hatami, S., R. Ruiz, and C. Andrés-Romano. 2015. “Heuristics and Metaheuristics for the Distributed Assembly Permutation Flowshop Scheduling Problem with Sequence Dependent Setup Times”. *International Journal of Production Economics* 169:76–88.

Juan, A., B. Barrios, E. Vallada, D. Riera, and J. Jorba. 2014. “A Simheuristic Algorithm for Solving the Permutation Flow-shop Problem with Stochastic Processing Times”. *Simulation Modelling Practice and Theory* 46:101–117.

Juan, A., J. Faulin, S. Grasman, M. Rabe, and G. Figueira. 2015. “A Review of Simheuristics: Extending Metaheuristics to Deal with Stochastic Optimization Problems”. *Operations Research Perspectives* 2:62–72.

Katragjini, K., E. Vallada, and R. Ruiz. 2013. “Flow Shop Rescheduling Under Different Types of Disruption”. *International Journal of Production Research* 51 (3): 780–797.

Kianfar, K., S. F. Ghomi, and A. O. Jadid. 2012. “Study of Stochastic Sequence-dependent Flexible Flow Shop via Developing a Dispatching Rule and a Hybrid GA”. *Engineering Applications of Artificial Intelligence* 25 (3): 494–506.

Koulamas, C., and G. J. Kyparisis. 2001. “The Three-stage Assembly Flowshop Scheduling Problem”. *Computers & Operations Research* 28 (7): 689–704.

Liao, C.-J., C.-H. Lee, and H.-C. Lee. 2015. “An Efficient Heuristic for a Two-stage Assembly Scheduling Problem with Batch Setup Times to Minimize Makespan”. *Computers & Industrial Engineering* 88:317–325.

Liu, Q., S. Ullah, and C. Zhang. 2011. “An Improved Genetic Algorithm for Robust Permutation Flowshop Scheduling”. *The International Journal of Advanced Manufacturing Technology* 56 (1-4): 345–354.

Lourenço, H., O. Martin, and T. Stützle. 2010. “Iterated Local Search: Framework and Applications”. In *Handbook of Metaheuristics*, edited by M. Gendreau and J.-Y. Potvin, Volume 146 of *International Series in Operations Research & Management Science*, 363–397. Springer US.

Naderi, B., and R. Ruiz. 2010. “The Distributed Permutation Flowshop Scheduling problem”. *Computers & Operations Research* 37 (4): 754–768.

Nawaz, M., J. Enscore, and I. Ham. 1983. “A Heuristic Algorithm for the m-machine, n-job Flowshop Sequencing Problem”. *Omega* 11:91–95.

Roy, B. 2010. “Robustness in Operational Research and Decision Aiding: A Multi-faceted Issue”. *European Journal of Operational Research* 200 (3): 629–638.

Seidgar, H., M. Kiani, M. Abedi, and H. Fazlollahtabar. 2014. "An Efficient Imperialist Competitive Algorithm for Scheduling in the Two-stage Assembly Flow Shop Problem". *International Journal of Production Research* 52 (4): 1240–1256.

Sung, C., and H. A. Kim. 2008. "A Two-stage Multiple-machine Assembly Scheduling Problem for Minimizing Sum of Completion Times". *International Journal of Production Economics* 113 (2): 1038–1048.

Sung, C. S., and J. Juhn. 2009. "Makespan Minimization for a 2-stage Assembly Scheduling Problem Subject to Component Available Time Constraint". *International Journal of Production Economics* 119 (2): 392–401.

Taillard, E. 1993. "Benchmarks for Basic Scheduling Problems". *European Journal of Operational Research* 64:278–285.

Talbi, E. 2009. *Metaheuristics: From Design to Implementation*. Wiley.

Zhou, Q., and X. Cui. 2008. "Research on Multiobjective Flow Shop Scheduling with Stochastic Processing Times and Machine Breakdowns". In *IEEE International Conference on Service Operations and Logistics, and Informatics*, Volume 2, 1718–1724. IEEE.

AUTHOR BIOGRAPHIES

LAURA CALVET is a PhD student at the IN3 - Open University of Catalonia. She holds a MSc in Statistics and Operations Research, one BSc in Economics and another in Applied Statistics. Her work is related to the combination of statistical learning and simheuristics for solving complex combinatorial optimization problems under uncertainty. Her email address is lcalvetl@uoc.edu.

VICTOR FERNANDEZ-VIAGAS is Researcher in the Industrial Management Research Group at the University of Seville. His research interests include production and project scheduling, and operating room planning and scheduling. His email address is vfernandezviagas@us.es.

JOSE M. FRAMINAN is a Professor of Industrial Engineering in the School of Engineering at the University of Seville. His research interests include decision support systems and procedures for operations and supply chain management. His email address is framinan@us.es.

ANGEL A. JUAN is Associate Professor at the Open University of Catalonia. He holds a PhD in Industrial Engineering and a MSc in Mathematics. He completed a pre-doctoral internship at Harvard University and a postdoctoral internship at the MIT Center for Transportation and Logistics. His research interests include applications of randomized algorithms and simheuristics in logistics, production, and Internet computing. He has published over 150 peer-reviewed papers in these fields. His website address is http://ajuanp.wordpress.com and his email address is ajuanp@uoc.edu.