ARTICLE IN PRESS

Simulation Modelling Practice and Theory xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times

Angel A. Juan [a,*], Barry B. Barrios [a], Eva Vallada [b], Daniel Riera [a], Josep Jorba [a]

[a] Computer Science Department, IN3-Open University of Catalonia, Barcelona, Spain
[b] Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Universitat Politècnica de València, Valencia, Spain

ARTICLE INFO

Article history:
Available online xxxx

Keywords:
Permutation flow shop problem
Simulation–optimization
Stochastic times
Randomized algorithms
Iterated local search
Monte-Carlo simulation
Reliability analysis

ABSTRACT

This paper describes a simulation–optimization algorithm for the Permutation Flow shop Problem with Stochastic processing Times (PFSPST). The proposed algorithm combines Monte Carlo simulation with an Iterated Local Search metaheuristic in order to deal with the stochastic behavior of the problem. Using the expected makespan as initial minimization criterion, our simheuristic approach is based on the assumption that high-quality solutions (permutations of jobs) for the deterministic version of the problem are likely to be high-quality solutions for the stochastic version – i.e., a correlation will exist between both sets of solutions, at least for moderate levels of variability in the stochastic processing times. No particular assumption is made on the probability distributions modeling each job-machine processing times. Our approach is able to solve, in just a few minutes or even less, PFSPST instances with hundreds of jobs and dozens of machines. Also, the paper proposes the use of reliability analysis techniques to analyze simulation outcomes or historical observations on the random variable representing the makespan associated with a given solution. This way, criteria other than the expected makespan can be considered by the decision maker when comparing different alternative solutions. A set of classical benchmarks for the deterministic version of the problem are adapted and tested under several scenarios, each of them characterized by a different level of uncertainty – variance level of job-machine processing times.

1. Introduction

The Flow Shop Scheduling Problem (FSP) is a well-known scheduling problem that can be described as follows: a set J of n jobs has to be processed by a set M of m machines. Each job $i \in J$ is composed by an ordered set of m operations, O_{ij}, that must be sequentially performed by the m machines (one operation per machine). A special case of the FSP is the Permutation Flow Shop Scheduling Problem (PFSP), where the processing order of operations in machines is the same for all jobs – i.e., all jobs are processed by all machines in the same order. Each operation O_{ij} requires a processing time. In the PFSP, the goal is to find a sequence (permutation) of jobs so that a given criterion is optimized. The most commonly and studied criterion is the minimization of the completion time or makespan, i.e., the time it requires to process all the jobs throughout all the machines (Fig. 1). Although the usual goal is to minimize the makespan, other goals can also be considered, e.g., the

* Corresponding author. Tel.: +34 933263627.
E-mail addresses: ajuanp@uoc.edu (A.A. Juan), barry.brian.barrios@gmail.com (B.B. Barrios), evallada@eio.upv.es (E. Vallada), drierat@uoc.edu (D. Riera), jjorbae@uoc.edu (J. Jorba).

http://dx.doi.org/10.1016/j.simpat.2014.02.005
1569-190X/

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

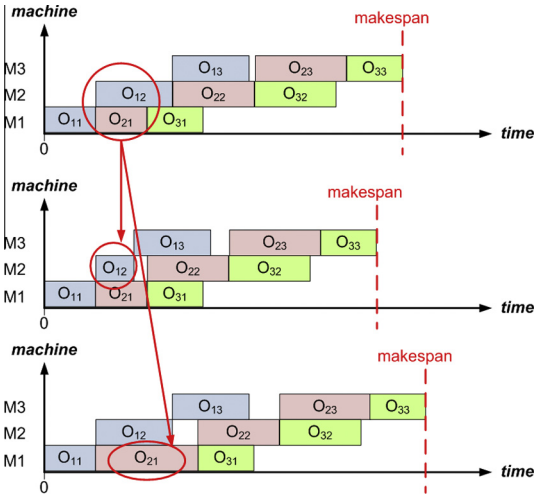

Fig. 1. Effect of random job processing times over makespan.

minimization of the total processing time – i.e., the sum of the individual processing times of each job in each machine, or the total tardiness of jobs with respect to their scheduled deadlines. In Framinan et al. [22] and Ruiz and Maroto [50] surveys about the PFSP with the objective to minimize the makespan are provided. Similarly, Vallada et al. [58] contains a survey for the PFSP with the objective of minimizing total tardiness. In addition, it is usually assumed that: (i) there is unlimited storage buffer between machines, and preemption (task interruption) is not considered; (ii) machines are always available for processing jobs, but each machine can process only one job at a time; (iii) a job cannot be processed more than once for each machine; and (iv) job processing times are independent. In most of the existing literature, the processing time of each job i in each machine j is considered to be a known constant value, p_{ij}. This problem, which is NP-hard [24], has many practical applications in both economic and industrial fields as it exists in many real-life scenarios, including, among others: the manufacturing and production industry, the services sector, the biochemical and pharmaceutical industries, the computing and telecommunication sector, etc. [52].

The PFSP with Stochastic Times (PFSPST) can be seen as a generalization of the PFSP in which the processing time of each job i in each machine j is not a constant value, but instead it is a random variable, P_{ij}, following a non-negative probability distribution – e.g., Log-Normal, Exponential, Weibull, Gamma, etc. Since uncertainty is present in most real-life processes and systems, considering random processing times represents a more realistic scenario than simply considering deterministic times. In effect, unforeseen circumstances can lead to sudden changes in the processing time of certain jobs in certain machines, which is likely to have noticeable effects on the predicted makespan. Therefore, one goal that can be considered when dealing with the PFSPST is to determine a sequence (permutation) of jobs that minimizes the expected makespan or mean time to completion of all jobs.

The study of the PFSPST is within the practice of introducing randomness into combinatorial optimization problems as a way of describing more realistic problems in which most of the information and data cannot be known beforehand [59]. For these stochastic problems, efficient simulation–optimization methods have been developed during the last decades. Among these methods, we are particularly interested in the combination of simulation with metaheuristics (simheuristics), since this hybridization constitutes a promising approach yet to be explored in its full potential. Andradóttir [3] offers an overview of simulation–optimization using metaheuristics, while some recent articles combining simulation with metaheuristics can be found in Angelidis et al. [4] and Laroque et al. [40]. In the context of scheduling problems, several authors have discussed the role of simulation in solving real-life instances and have also proposed simulation-based approaches which are typically combined with optimization algorithms (including metaheuristics), among others: Azzaro-Pantel et al. [8], Deroussi et al. [19], Arakawa et al. [6], Fowler et al. [21], Klemmt et al. [39], Frantzén et al. [23], Hu and Zhang [29], Bassi et al. [13], or Kima and Choi [38].

Fig. 1 illustrates a simple PFSP with three jobs and three machines, where O_{ij} represents the operation of job i in machine j ($1 \leqslant i \leqslant 3$, $1 \leqslant j \leqslant 3$). Notice that, for a given permutation of jobs, even a single change in the processing time of one job in one machine (O_{12} and O_{21} in the figure, respectively) can have a noticeable impact on the value of the final makespan.

As with other combinatorial optimization problems, a number of different approaches and methodologies have been developed to deal with the PFSP. These approaches range from exact optimization methods – such as linear and constraint programming, which can provide solutions to small-size problems, to approximate methods – such as heuristics and metaheuristics, which can provide near-optimal solutions for medium- and large-sized problems. Moreover, some of these methodologies are able to provide a set of alternative solutions among which the decision-maker can choose according to his/her expertise. However, the situation with the PFSPST is different. To the best of our knowledge, there is a lack of methods able to

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

provide high-quality solutions to the stochastic version of the PFSP. As discussed in the literature review section, most of the existing approaches are quite theoretical and require many assumptions on the probability distributions that model job processing times, while other approaches seem to be valid only for small-size instances. In the articles by Dodin [20], Honkomp et al. [28], Gourgand et al. [27], and Baker and Altheimer [10], simulation-based techniques have been used to get results for the PFSPST. In some of these articles, however, simulation was mainly used as a backup method to validate the results generated by other analytical methods (e.g., Markov chains). Consequently, only Normal or Exponential probability distributions were employed to model processing times. Moreover, these papers made strong assumptions on the size of the instances being analyzed.

Accordingly, the main contributions of this paper to the existing PFSPST literature can be summarized as follows:

(a) To introduce a simulation–optimization algorithm able to solve large-size instances in short computing times – just a few minutes or even less – regardless the probability distributions that model the stochastic processing times at each job-machine pair. Our approach does not make any assumption on the size of the instances or on the specific distributions employed to model processing times. In fact, in a real-life scenario, the specific distributions to be used will have to be fitted from historical data (observations). The assumption of processing times following a Normal distribution – quite common in the existing literature – is quite unrealistic and restrictive, since distributions such as the Log-Normal or the Weibull are usually much better candidates to model processing times with positive values.
(b) To discuss how real-life observations associated with a given permutation of jobs – or, alternatively, simulation outputs provided by our approach – can be analyzed using reliability techniques [30] to provide more information on the stochastic makespan. As we discuss later in more detail, it can be very convenient to use survival functions (inverses of cumulative probability functions) to compare alternative PFSPST solutions with similar expected makespans, specially in those scenarios in which percentile information on the makespan variable associated with a given permutation of jobs can be useful. Reliability analysis is preferred here over classical statistical techniques due to the fact that censored or incomplete observations might be present when real-life data are collected.
(c) To introduce a set of well-defined and easily replicable PFSPST instances based on well-tested PFSP benchmarks.

The rest of the paper is organized as follows: Section 2 offers a complete literature review on the PFSPST. Section 3 provides an (upper-level) overview of our simulation-based approach. Section 4 gives the (lower-level) details, which might be necessary for implementing a software version of the proposed method. Section 5 presents some numerical experiments that illustrate our methodology. Section 6 discusses the use of reliability analysis techniques to compare alternative solutions. Finally, Section 7 summarizes the main contributions of the paper.

2. Literature review on the PFSP with stochastic times

The literature on the Permutation Flow shop Problem with Stochastic Times (PFSPST) is not as extensive as for the deterministic case. Banerjee [12] and Makino [43] were the first to study the PFSP with stochastic times. The former proposed a decision rule for the single machine problem with random processing times following a known probability distribution, with the objective of minimizing the maximum probability of lateness. The latter studied the 2-jobs with 2-machines problem as well as the 2-jobs with 3-machines problem considering Exponential and k-Erlang distributions for the processing times. The goal was to minimize the expected makespan. An extension of Makino's work for 3 and 4 jobs is found in Talwar [57], where the processing times are also exponentially distributed, with the objective of minimizing the idle time of the last machine. Bagga [9] considered the model where the PFSPST has an intermediate storage of infinite capacity between all machines and also assumed exponentially distributed processing times. Another optimal rule, in this case for the n-jobs with 2-machines problem, is that proposed by Cunningham and Dutta [18], where the processing times of the jobs are assumed to be exponentially distributed and the objective is to minimize the expected job completion time. In Weber [60], we find two rules for the identical parallel machines problem with random processing times and with the complementary objectives of minimizing the expected makespan or the flow-time. In Pinedo [47], models for the permutation flow shop problem with and without intermediate storage are proposed, where the processing times of a given job on the various machines are independent realizations from the same probability distribution and the objective is to minimize the expected makespan. This author also considered the case where only two jobs are stochastic and the remaining $(n-2)$ jobs are deterministic. He tried to prove that if a sequence of jobs schedules either one of the stochastic jobs first and the other one last, then it minimizes the expected makespan. However, Suresh et al. [53] showed later that Pinedo's conjecture was not always true. Wie and Pinedo [63] considered the PFSPST with the objective of minimizing the makespan and blocking, that is, they assumed that there is no storage space between two machines and, therefore, a job may leave a machine and start its processing on the next one only when the next machine is free.

Dodin [20] considered different probability distributions (Uniform, Normal, and Exponential) for processing times. He assumed these processing times were independent and not identically distributed random variables. His goal was also to minimize the expected makespan. Kamburowski [34] and Kamburowski [35] studied the 2-machines and 3-machines cases, respectively, where the processing times were assumed to be independent variables and the objective was to minimize the expected makespan. In Zhang et al. [66] the authors considered a hypothesis-test based simulated annealing, also with the

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

objective of minimizing the makespan. Gourgand et al. [26] and Gourgand et al. [27] used recursive algorithms, based on Markov chains, to compute the expected makespan in conjunction with a simulation model. However, the simulation model was mainly used for validating the Markovian model and, therefore, simulation was not being exploited in its full potential. Also, the proposed Markovian model has severe limitations, since it only can deal with cases where the problem size is relatively small. As stated by the authors, "The CPU time for the Markovian model increases very quickly with the number of machines and the number of jobs [...] for 20 jobs and 10 machines, the Markovian model obtains no result in less than 4 h." Furthermore, these models seem to be only applicable when processing times are exponentially distributed. A generalization of Johnson's and Talwar's rules for the 2-machine case can be found in Kalczynski and Kamburowsky [33], where the random processing times are independent and Gompertz distributed – again, the objective here is to minimize the expected makespan. Wang et al. [61] and Wang et al. [62] proposed Genetic Algorithms approaches for minimizing the expected makespan when processing times follow a Uniform distribution. More recently, Baker and Trietsch [11] developed heuristics for the 2-machine PFSPST where the processing times are independent random variables following a known probability distribution – also with the objective of minimizing the expected makespan. In Baker and Altheimer [10] heuristics for the m-machine case are proposed for the same problem. Some authors also studied different variations of the permutation flow shop problem. Thus, Allaoui et al. [2], Choi and Wang [17], and Kianfar et al. [37] considered the stochastic hybrid flow shop scheduling problem. In the first two works, the objective was to minimize the expected makespan, while in the case of the later one the goal was to minimize the average tardiness of jobs. In Zhou and Cui [67], an approach for the multi-objective stochastic PFSP is considered. Here, processing times are normally distributed with two objectives to minimize: flow time and delay time of the jobs.

It is possible to notice that most of the aforementioned publications made restrictive assumptions on the probability distributions modeling the processing times of each job-machine pair. Also, some of these authors even made the assumption that the processing times are independent and identically distributed in order to simplify the problem, which is not a quite realistic hypothesis. Different objective functions are considered, even multi-objective approaches. However, most of the papers focus on optimizing expected makespan, while very few of them consider additional probabilistic information on the stochastic makespan associated to a given solution.

Regarding other approaches used to deal with uncertainty, it is possible to consider two fundamental types: proactive scheduling and reactive scheduling. *Proactive scheduling* or *robust scheduling* [49], takes uncertainty into consideration to construct an original predictive schedule called "the reference". This approach is interesting if this uncertainty can easily be quantified. When there are no actual data to formulate a distribution, it is useful to consider several scenarios. More specifically, r*obust scheduling* tries to obtain schedules that are able to confront disruptions without requiring new schedules, that is, they can be adapted to external events with little change. Recently, several authors have studied this problem. In Artigues et al. [7] a family of schedules are generated where the operations within a family are totally permutable. In Bonfill et al. [14] a genetic algorithm is proposed to obtain robust schedules. In Pierreval and Durieux-Paris [46] an approach to compare solutions taking into account their robustness is proposed. Regarding the permutation flow shop problem, Al Kattan and Maragoud [1], Ghezail et al. [25], and more recently Liu et al. [41] proposed methods to deal with uncertainties from a robustness point of view. Several papers can also be found for other scheduling problems: Chaari et al. [16] addressed the hybrid flow shop scheduling problem, and regarding the parallel machines problem some results are found in Anglani et al. [5] and Xu et al. [64]. *Reactive scheduling*, on the other hand, revises and re-optimizes the schedule when an unexpected event occurs. It may be based on a predictive scheduling. In this case, one tries to repair this existing schedule by taking into account the actual state of the system. Otherwise, one can make decisions dynamically as an unexpected event occurs. Thus, in Katragjini et al. [36] an exhaustive review about rescheduling under disruptions can be found.

It is important to notice also that most authors are using exact methods or rules to solve the PFSPST, while only a few are using simulation-based methods. Most exact methods are limited by the probability distribution they can use and the size of the problem they can solve. Also, exact methods cannot easily handle dependencies among processing times. However, simulation techniques are able to take these situations into account in a quite natural way. Recently, Baker and Altheimer [10] combined simulation and heuristics to solve the PSFPST with the objective of minimizing the expected makespan. They created three sets of stochastic flow shop problems where the processing times were distributed following an Exponential, Uniform, and Log-Normal, respectively. The authors proposed three heuristic methods, two of them based on the CDS heuristic by Campbell et al. [15]. The third one is based on the well known NEH heuristic by Nawaz et al. [45]. Moreover, the authors used the genetic algorithm available in the Excel Solver. They considered the result of that genetic algorithm as a "near-optimal solution" to calculate the sub-optimality of the sequences obtained by the three proposed heuristics. However, the size of the benchmarks tested by these authors was limited to small problems with up to 10 jobs and 6 machines. As shown in the experimental section, our approach is scalable to much larger instances. Moreover, according to some preliminary results, our approach seems to perform at least as well as the ones proposed by Baker and Altheimer [10], especially as the size of the problem increases. Unfortunately, we could not perform a fair and detailed comparison with their approaches due to the reduced reproducibility level of the results published in their paper: (a) no computational times were provided; (b) no absolute values were provided either, just percentage gaps with respect an unspecified best value – even so, we could estimate this unknown value using the NEH value as a common reference base; and (c) the parameters of the probability distributions used in their experiments were not given as constant values, but instead they were random variables too – thus introducing an unnecessary random effect in the definition of the problem inputs that can have a noticeable impact on

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

the final results. To facilitate reproducibility of results and avoid such comparison problems in future research on the PFSPST, our paper provides a well-defined set of benchmarks as well as complete information about the obtained results.

3. Overview of the SIM-ESP simheuristic approach

Our approach is inspired by the following facts: (a) the PFSPST can be seen as a generalization of the PFSP or, to be more specific, the PFSP is just a PFSPST with constant demands – random demands with zero variance; and (b) while the PFSPST is yet an emerging research area, extremely efficient metaheuristics do already exist for solving the PFSP – in fact, state-of-the-art metaheuristics based on the use of Genetic Algorithms, Iterated Local Search, Tabu Search, Simulated Annealing, or Ant Colony Optimization, are able to provide near-optimal solutions for most known PFSP benchmarks. Therefore, given an initial PFSPST instance, our approach proposes: (i) to transform the stochastic instance into a deterministic instance by replacing each random variable time by its mean or expected time; and then (ii) to obtain a set of high-quality solutions for the deterministic problem by using any efficient PFSP algorithm. At this point, it should be noticed that a solution to the PFSP instance is simply a given permutation of jobs, and therefore it is also a feasible solution of the original PFSPST instance. Then, simulation is used to determine which solution, among the best-found deterministic ones, produces the lowest expected makespan when considering stochastic times. To be more specific: (i) given a solution (permutation of jobs) to the PFSP, the probability distribution of each job-machine processing time is used to generate a random variate; and then, (ii) these random variates are used to compute the stochastic makespan associated with the given permutation of jobs. Iterating this process, it is possible to generate a random sample of makespan observations associated with the given solution. From this sample, different statistics for the stochastic makespan can be obtained, including point and interval estimates for the expected stochastic makespan.

From the previous discussion it is clear that our strategy assumes that there exists a strong correlation between solutions for the PFSP and solutions for the PFSPST. Put in other words, the set of 'good' solutions for the PFSP is likely to intersect the set of good solutions for the PFSPST. We expect this assumption to be particularly valid when stochastic times show a relatively low or medium variability. Notice, however, that the best-found PFSP solution – the one with the lowest deterministic makespan – will not necessarily be the best-found PFSPST solution – the one with the lowest expected stochastic makespan. This is due to the fact that the stochastic makespan of a solution might be sensitive to variations in the processing times – i.e., the aforementioned correlation will usually be smaller than one. Fig. 2 illustrates these ideas with data obtained from one of the classical instances described later in the experimental section: instance 59 from the Taillard's set where processing times, P_{ij}, follow a Log-Normal distribution with $\mathrm{Var}[P_{ij}] = k\ E[P_{ij}]$ (with $k \in \{0.1, 0.5, 2\}$). The points in Fig. 2 were obtained after an extensive exploration of the solution space. Each point represents a good solution for both the PFSP and the PFSPST. For each of these solutions, the associated makespan is plotted in the horizontal axis, while the associated expected makespan is plotted in the vertical axis. Notice how as the variance level increases (higher values of k) the expected values of the stochastic makespans are also shifted upwards – i.e., more uncertainty levels implies higher expected makespans. Also, notice that as the variance level increases, the slope of the associated regression line diminishes. Thus, the higher the variance level the more likely it is that ranking differences can be observed between the list of top solutions sorted by deterministic value and the list of top solutions sorted by stochastic value. For the three analyzed scenarios, however, the Pearson correlation coefficient, r, is significant: $r = 0.973$ (p-value = 0.000) for $k = 0.1$; $r = 0.942$ (p-value = 0.000) for $k = 0.5$; and $r = 717$ (p-value = 0.03) for $k = 2$.

In order to efficiently implement the aforementioned approach, we have designed the SIM-ESP simheuristic algorithm, which combines the Iterated Local Search (ILS) metaheuristic [42] with simulation. The ILS framework has been successfully used in the past to solve the PFSP. In particular, it is worthy to mention the IG algorithm developed by Ruiz and Stützle [51] for solving the deterministic version of the problem using an ILS-based framework. According to different authors [51,68,48],

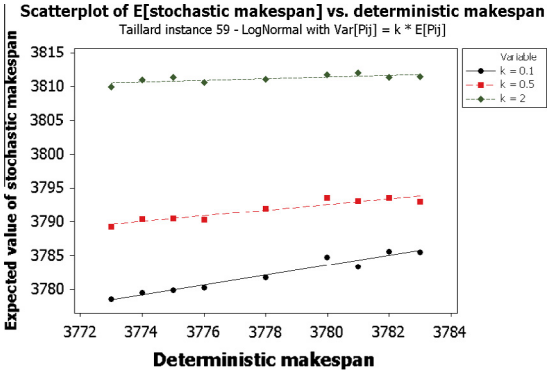

Fig. 2. Correlation assumption.

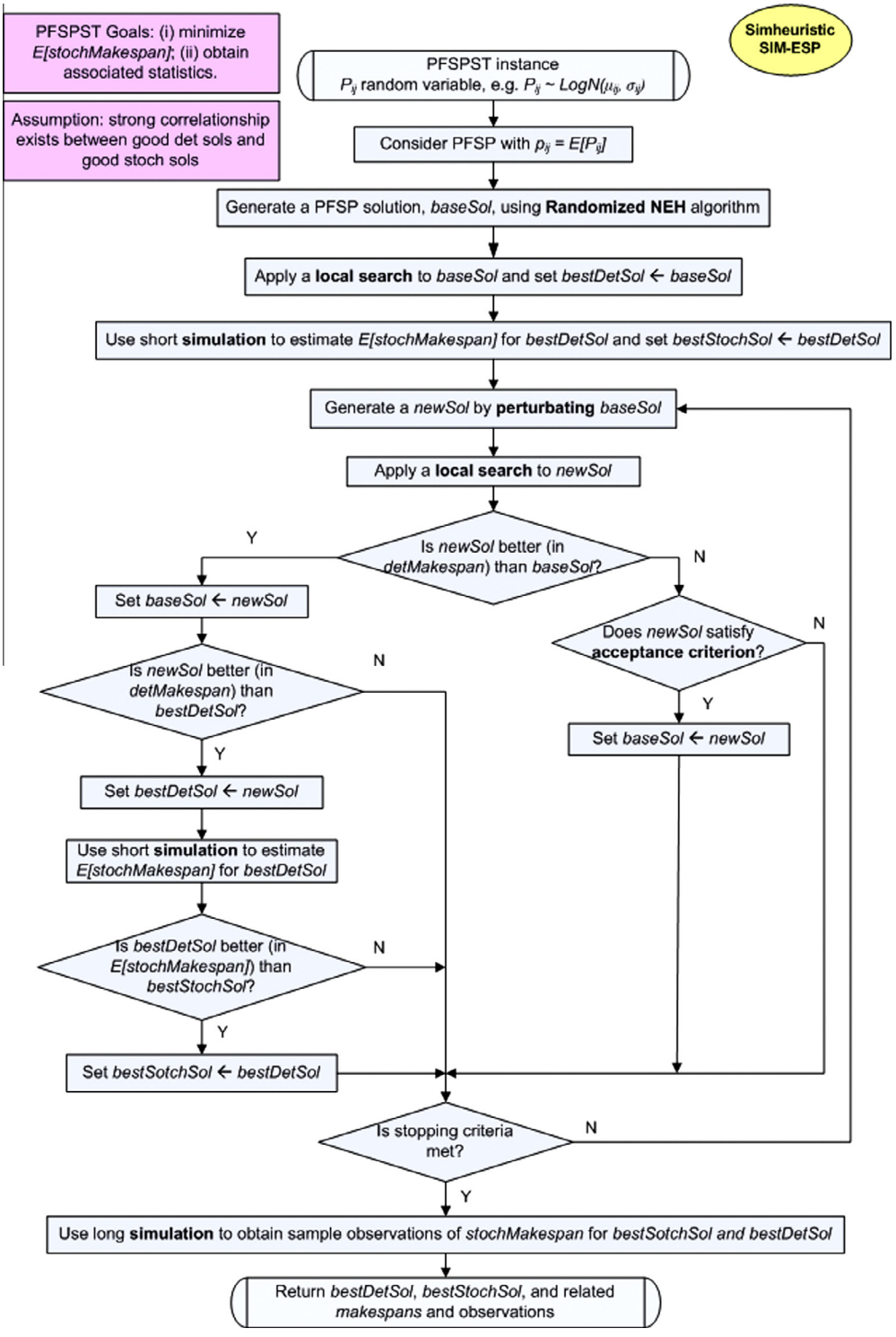

Fig. 3. Flowchart diagram of the SIM-ESP simheuristic algorithm.

the IG algorithm has outperformed any other algorithm so far, including Genetic Algorithms and Tabu Search algorithms, with many more parameters. That is why our approach is based in an ILS framework. Fig. 3 shows the overview flow-chart of the SIM-ESP simheuristic, which is explained next:

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

1. Consider a PFSPST instance defined by a set of jobs J and a set of machines M with stochastic processing times, P_{ij}, for each job $i \in J$ in each machine $j \in M$, where each P_{ij} follows a known probability distribution – either theoretical or empirical – with known mean $E[P_{ij}]$.
2. Consider the PFSP defined by a set of jobs J and a set of machines M, where the processing times, $p_{ij,}$ are deterministic values given by $p_{ij} = E[P_{ij}]$.
3. Generate a base solution (permutation of jobs) to the PFSP using the well-known NEH heuristic [45] or, alternatively, a biased-randomized version. The randomized approach – whose details are given in the next section – aims at increasing exploration of the solutions space by diversifying the initial solution employed by the algorithm when multiple runs are executed.
4. Apply a classical local search process to the base solution – see details in the next section – and consider the improved solution as the new base solution as well as the best 'deterministic' solution so far – where deterministic refers to the PFSP.
5. Use a short simulation (for instance, about 200 runs) to quickly get an estimate of the expected makespan associated with the permutation of jobs given by the best deterministic solution. Then, initialize the best 'stochastic' solution as the best deterministic one – where stochastic refers to the PFSPST.
6. Start an Iterated Local Search process in which both best deterministic and stochastic solutions will be iteratively improved. This ILS process will continue until a stopping condition, usually time-based, is reached. At each iteration of this process the following steps are completed: (i) a perturbation operator and a subsequent local search are applied to the base solution to generate a new solution – see details in next section; (ii) if the makespan of the new solution is lower than the one of the current base solution, then the base solution is updated and the new solution is also compared against the best deterministic solution so far to decide if the latter needs to be updated as well; each time the best deterministic solution is updated, a new short simulation is used to estimate the new stochastic makespan and, if appropriate, update the best stochastic solution; and (iii) in case (ii) does not apply, then an acceptance criterion – see details in next section – is used to decide whether or not the base solution must be updated (deteriorated) to the new solution – this deterioration of the base solution is sometimes allowed in order to reduce the risk of getting trapped in a local minimum.
7. Use a long simulation (for instance, 1E6 runs) to generate accurate estimations for the expected values of two different stochastic makespans, the one associated with the best stochastic solution and the one associated with the best deterministic solution. Notice that, when considering the parameters (not the estimates), the deterministic makespan associated with the best deterministic solution and the (stochastic) expected makespan associated with the best deterministic solution constitute, respectively, a lower and an upper bound for the (stochastic) expected makespan associated with the best stochastic solution.

Finally, the algorithm returns the following information: (i) the best deterministic and stochastic solutions – including associated makespans, and (ii) the sample observations obtained for stochastic makespans associated with the best deterministic and the best stochastic solutions – as showed in the experimental section, these observations can be used in order to obtain additional statistics which allow to understand better their individual nature as well as to compare different solutions using criteria other than makespan.

As discussed in the literature review, several authors have proposed approaches that transform the PFSPST into an equivalent PFSP and then apply an existing PFSP heuristic to the resulting problem, among others Dodin [20] or Gourgand et al. [26]. However, our approach differs from those others in the following aspects: (i) while previous approaches are founded on theoretical chance-constrained models, our approach uses a more practical perspective based on the use of Monte Carlo simulation; and (ii) while previous approaches need to assume a particular behavior for the random variables that model the processing times, our simulation-based approach offers more flexibility and does not require these assumptions that, in some cases, may be rather artificial or restrictive. Thus, as far as we know, the methodology presented here offers unique advantages over other existing approaches for solving the PFSPST. Some of the potential benefits of our approach are discussed next:

- Since it uses simulation to deal with the stochastic behavior of processing times, the methodology is valid for any probability distribution with a known mean, either theoretical – e.g., Normal, Log-Normal, Weibull, or Gamma – or experimental – in which case bootstrapping techniques can be used to generate the random values. Being mostly associated with non-negative and continuous values, real-life processing times are not likely to follow a Normal or even an Exponential distribution – as it is often assumed in the existing PFSPST literature. On the contrary, as it is usually done in reliability analysis, random times should be modeled by using any theoretical or experimental distribution offering non-negative values and asymmetries generated by long right-hand tails – e.g., Log-Normal, Weibull, or Gamma.
- In some sense, the methodology is reducing a complex PFSPST – where no efficient metaheuristics have been developed yet – to a more tractable PFSP where excellent, fast and extensively tested metaheuristics exist. In fact, one of these efficient metaheuristics is employed as the basis for our approach. This adds credibility to the quality of the final solution or solutions provided to the decision-maker.

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

- By using simulation, the methodology can be naturally extended to consider a different distribution for each processing time and even possible dependencies among these processing times – e.g., by updating the individual distributions during the simulation as a way to model these dependencies.
- Finally, notice that the methodology exposed here can be applied to PFSPST instances of large sizes. In particular, as far as the deterministic version of the problem can be solved – and current metaheuristics can obtain good solutions to PFSP instances with hundreds of jobs and machines in just a few minutes or even less, the stochastic version of the problem can be also solved using our approach. The simulation component does not add too much additional time (typically just some seconds), since during the local search process we run short simulations and we only run them in a reduced number of promising solutions.

4. Some additional details of our approach

While the previous section provided an overview of the SIM-ESP algorithm, this section offers some additional details that are relevant to completely understand our approach as well as to implement it as a computer application. One procedure used in different parts of the SIM-ESP algorithm is the local search (Fig. 4), which in our case is the same method used by many other authors, e.g., Ruiz and Stützle [51]. Roughly speaking, it consists in an iterative process with the following steps at each iteration: (i) a position k is randomly selected from the permutation of jobs which define the current solution; (ii) a *'shift-to-left'* movement is applied on the job at position k, i.e., the job is inserted in each possible position on the left of k (Fig. 5) and, for each of this insertions, the resulting makespan is computed using Taillard's accelerations [54]; and *(iii)* finally, the job is set at the position providing the best makespan –including the original one if no improvement is reached. These steps are iterated until all positions have been visited or an improvement is reached – in the latter case, the set of already visited positions becomes empty and the entire process is restarted.

In our approach, we make use of a biased-randomized version of the classical NEH heuristic [45]. In fact, most PFSP algorithms use the solution provided by the NEH heuristic as a relatively good initial solution. Using a heuristic solution instead of generating a uniformly random solution is typically considered a good practice in order to accelerate the convergence of metaheuristic algorithms [56]. However, it seems reasonable to think that when multiple runs of the algorithm are executed – either in sequential or in parallel mode, each run employing a different randomization seed, using always the same initial solution can be a severe drawback for fast convergence, especially for those instances in which the heuristic solution provides a relatively poor solution [32]. In this context, the term poor does not necessarily refer to the makespan value of the solution, but rather to the number of movements or transformations that must be applied to the initial solution in order to arrive at a near-optimal solution. Since we are especially interested in running multiple iterations of any given instance, we designed a strategy to generate different randomized initial solutions – all of them of similar quality in terms of

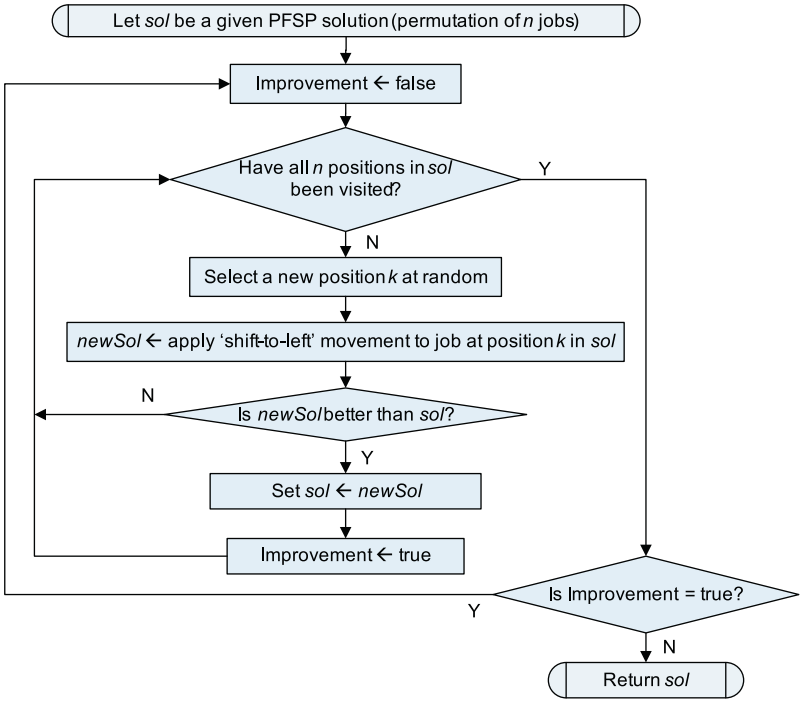

Fig. 4. Flowchart diagram of the local search process.

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

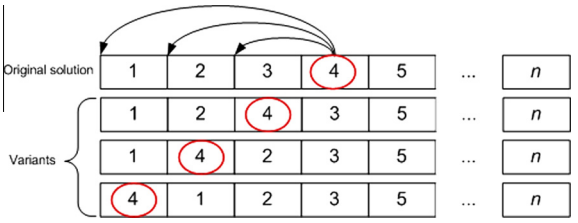

Fig. 5. Shift-to-left operator.

makespan – by introducing a biased (skewed) randomization process into the NEH heuristic. Using a skewed distribution instead of a uniform one allows to keep most of the logic behind the NEH heuristic. In effect, the NEH heuristic is an iterative algorithm which uses a list of jobs sorted by their total completion time on all the machines to construct a solution for the PFSP. At each step of this iterative process, the NEH removes the job at the top of that list (the one with maximum completion time) and adds this job into a permutation at the position that results in the best partial solution with respect to makespan. As a result, the NEH provides a 'common sense' deterministic solution, by trying to schedule the most demanding jobs first. For each job in the list, our method assigns to it a different probability of being selected (Fig. 6). According to our design, this probability should be coherent with the total processing time of each job, i.e.: jobs with higher total times will be more likely to be selected from the list before those with lower total times (skewed distribution of probabilities). To satisfy these requirements, we employ a discretized version of the decreasing triangular distribution during the solution-construction process: each time a new job has to be selected from the list, a triangular distribution that assigns linearly diminishing probabilities to each eligible job according to its corresponding total-processing-time value is employed. Other skewed probability distributions – like the geometric one – have been successfully employed in routing problems by Juan et al. [31] to generate multiple alternative solutions by inducing a similar biased-randomizaton process into a classical heuristic. In the case of the PFSP, the decreasing triangular probability distribution was chosen since it contains no parameters to be set and provides satisfactory results.

One of the most distinctive characteristic in our approach is the integration of a simulation procedure inside the described metaheuristic. Simulation is used here to estimate the expected makespan associated with a given solution – additional applications of the simulation component in our approach will be discussed later in this article. Thus, for a given solution and set of stochastic times, the simulation procedure performs as follows (Fig. 7): (i) using the corresponding probability distribution, a random variate is generated for each job-machine processing time; (ii) according to the permutation of jobs given by the solution, these random variates are used to generate a random observation of the (stochastic) makespan; and (iii) the previous steps are iterated in order to obtain a random sample of makespan observations, which can then be used to estimate the expected makespan – including interval estimates, as well as many other statistics of interests about the distribution of the stochastic makespan, e.g., quartiles, variance, or extreme values.

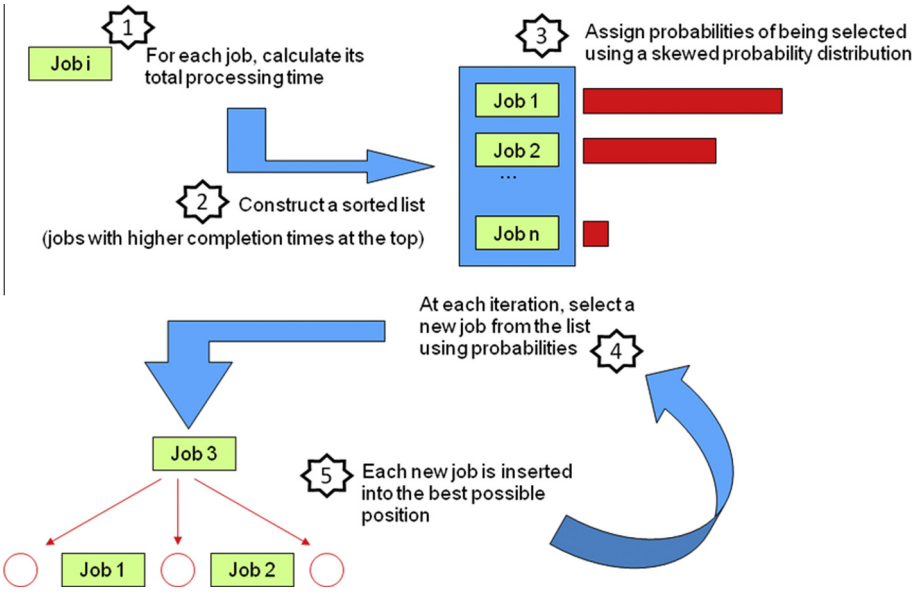

Fig. 6. Biased-randomized version of the classical NEH heuristic.

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

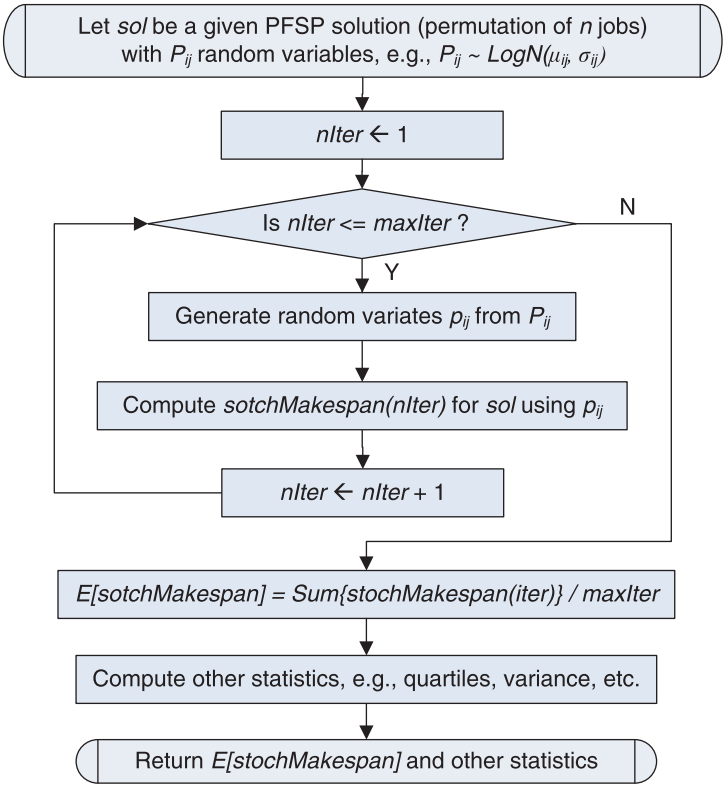

Fig. 7. Flowchart diagram of the simulation procedure.

Finally, the perturbation and the acceptance criterion processes employed in our approach for the PFSPST are the ones proposed in Juan et al. [32] for the deterministic PFSP. On the one hand, regarding the perturbation process, the so-called 'enhanced-swap' operator is used. This is a very simple, fast, and efficient operator that basically does the following: (i) randomly selects two different jobs from the current solution; (ii) interchanges both jobs, that is, interchanges their positions in the permutation; and (iii) applies a classical shift-to-left movement (described above) to each of these jobs following a left-to-right order. On the other hand, regarding the acceptance criterion, the algorithm uses a Demon-like process [56, pp. 138]. This Demon-like acceptance criterion, together with the perturbation process, is designed to help avoiding local minima during the algorithm execution. In order to do so, the criterion simply states that even if a newly generated solution is worse than the base solution, the base solution will be updated (deteriorated) to this new solution as long as: (i) no consecutive deteriorations take place; and (ii) the degradation does not exceed the value of the last improvement. A more detailed description of these two processes can be found in the aforementioned article. Notice, however, that both the perturbation operator and the acceptance criterion could be substituted by other similar operators and criteria – e.g., the ones proposed in Ruiz and Stützle [51] – without affecting too much the performance of the algorithm or its structure.

5. A numerical experiment varying the level of uncertainty

The SIM-ESP algorithm described in this paper was implemented as a Java application. An Intel Xeon at 2.0 GHz and 4 GB RAM was used to perform all tests, which were run directly on the Netbeans IDE platform for Java over Windows 7.

In the PFSP literature, there exists a classical set of very well-known benchmarks commonly used to test new algorithmic approaches. However, with the arguably exception of the somewhat limited benchmark proposed by Baker and Altheimer [10], there are no standard benchmarks in the PFSFST literature. In our opinion, this lack reveals the immaturity of the PFSFST knowledge area when compared with the PFSP area. Thus, we have decided to employ a natural generalization of several classical PFSP instances by using random processing times instead of constant ones. This approach has at least three advantages: (i) all data details –including processing times for each job-machine pair – are clearly given, so that other authors can use the same data sets for verifying and benchmarking purposes; (ii) we are using a well-known set of benchmarks – with high quality best-known solutions (BKS) – which includes instances of different sizes; and (iii) as discussed bellow, our PFSPST results for each instance can be compared with the corresponding BKS for the associated PFSP – ideally, our results should converge to these BKS as variances in processing times tend to zero. Thus, in order to test our methodology, we generalized the set of classical PFSP instances from Taillard [55]. These instances, which are available from http://mistic.heigvd.ch/taillard/default.htm, are grouped in 12 sets of 10 instances each according to the number of jobs and the

number of machines, i.e., set 20 × 5, set 20 × 10, set 20 × 20, set 50 × 5, set 50 × 10, set 50 × 20, set 100 × 5, set 100 × 10, set 100 × 20, set 200 × 10, set 200 × 20, and set 500 × 20.

For each instance in the Taillard's set, we changed p_{ij} – the deterministic processing times of job i in machine j – to stochastic processing times P_{ij} with $E[P_{ij}] = p_{ij}$. In other words, we considered the processing time of each job as a random variable following a well-known probability distribution with a given mean and a given variance. As stated before, one of the original contributions of our methodology is that, since it uses simulation, it can handle any probability distribution modeling the processing times, i.e., it does not need to assume that processing times follow a Normal or an Exponential distribution. To illustrate this, we selected a Log-Normal distribution for modeling processing times, although any other distribution with a known mean could have been used instead – notice that in a real-world case, historical data would be used to model each processing time by a different probability distribution. The Log-Normal distribution – which is a more natural choice than the Normal distribution when modeling non-negative processing times – has two parameters: the location parameter, μ_{ij}, and the scale parameter, σ_{ij}. According to the properties of the Log-Normal distribution, these parameters will be given by the following expressions:

$$\mu_{ij} = \ln(E[P_{ij}]) - \frac{1}{2} \cdot \ln\left(1 + \frac{Var[P_{ij}]}{E[P_{ij}]^2}\right)$$

$$\sigma_{ij} = \left|\sqrt{\ln\left(1 + \frac{Var[P_{ij}]}{E[P_{ij}]^2}\right)}\right|$$

Finally, we considered four different uncertainty scenarios by modifying the variance levels of each instance, i.e., we will consider $Var[P_{ij}] = k\,E[P_{ij}]$ for different values of the parameter k. In particular, we consider a first scenario with relatively low variances, specifically $k = 0.1$; a second scenario with relatively low-medium variances, specifically $k = 0.5$; a third scenario with relatively medium variances, specifically $k = 2$; and finally a fourth scenario with relatively medium–high variances, specifically $k = 5$. Fig. 8 illustrates these four scenarios for the case of a job i with an expected processing time in machine j of $E[P_{ij}] = 60$. While in the first (low-variance) scenario 95% of the actual processing times fall between 55.3 and 64.9, in the last (medium–high variance) scenario 95% of the actual processing times fall between 33.1 and 100.4. As discussed above

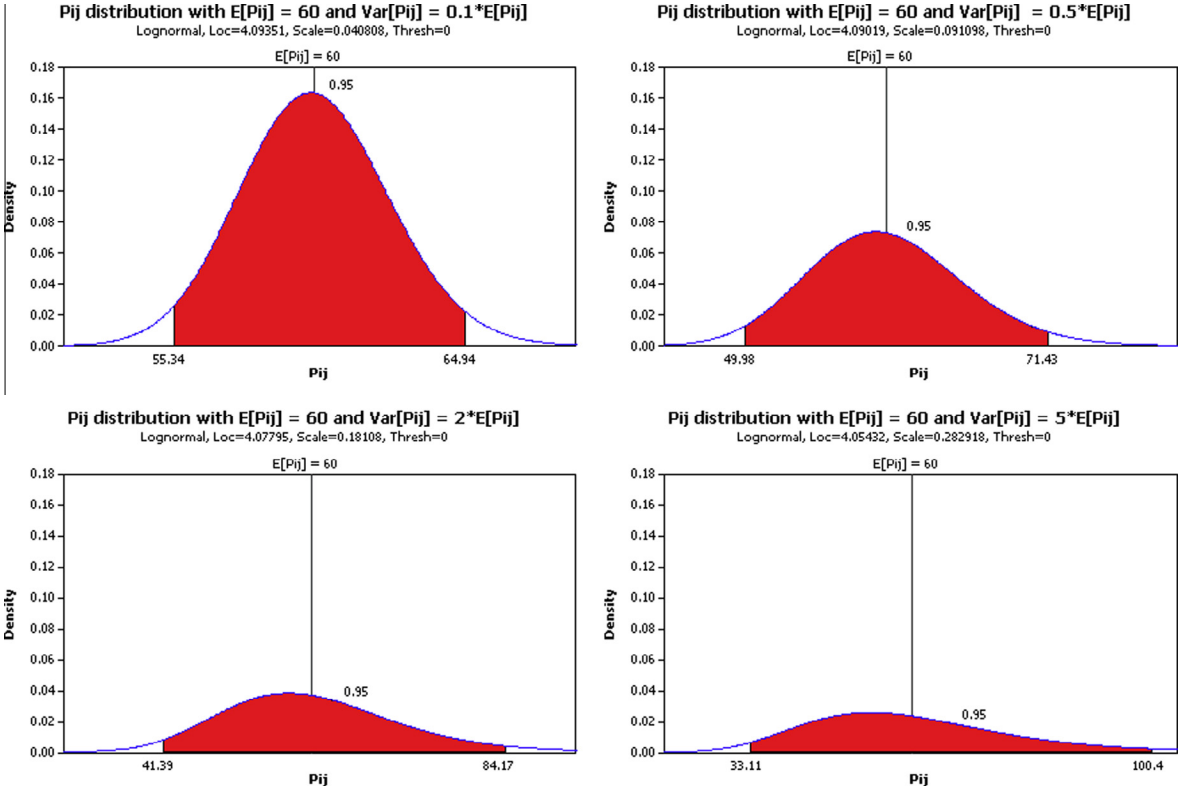

Fig. 8. Different uncertainty scenarios (variability levels).

Table 1
Best and average results of the SIM-ESP after 10 runs for 30 randomly selected Taillard's instances – low variability scenarios.

No.	Instance	PFSP (deterministic makespan)				PFSPST – LOW VARIANCE (expected stochastic makespan)											
						$k = 0.1$						$k = 0.5$					
		DM-BKS (1)	DM-BDS10 (2)	DM-ADS10	GAP (1)–(2) (%)	SM-BSS10 (3)	SM-ASS10	GAP (2)–(3) (%)	SM-BDS10 (4)	SM-ADS10	GAP (2)–(4) (%)	SM-BSS10 (5)	SM-ASS10	GAP (2)–(5) (%)	SM-BDS10 (6)	SM-ADS10	GAP (2)–(6) (%)
1	tai007_20_5	1234	1234	1237.5	0.00	1236.7	1240.7	0.22	1236.7	1240.7	0.22	1240.1	1243.8	0.49	1240.1	1243.9	0.49
2	tai009_20_5	1230	1230	1230.0	0.00	1231.8	1232.8	0.15	1231.8	1232.8	0.15	1237.3	1238.7	0.59	1237.3	1238.7	0.59
3	tai010_20_5	1108	1108	1108.0	0.00	1108.7	1109.9	0.06	1108.7	1109.9	0.06	1113.9	1115.8	0.53	1113.9	1116.1	0.53
4	tai011_20_10	1582	1582	1582.0	0.00	1583.8	1584.7	0.11	1583.8	1584.7	0.11	1587.9	1589.4	0.37	1588.8	1589.6	0.43
5	tai013_20_10	1496	1496	1496.8	0.00	1499.3	1500.0	0.22	1499.3	1500.0	0.22	1503.0	1504.2	0.47	1503.0	1504.7	0.47
6	tai027_20_20	2273	2273	2273.0	0.00	2273.2	2273.9	0.01	2273.2	2273.9	0.01	2277.1	2278.6	0.18	2277.1	2278.6	0.18
7	tai036_50_5	2829	2829	2829.0	0.00	2829.5	2830.7	0.02	2829.5	2831.2	0.02	2829.8	2831.8	0.03	2829.8	2832.6	0.03
8	tai040_50_5	2782	2782	2782.0	0.00	2782.1	2783.1	0.00	2782.1	2783.2	0.00	2783.1	2784.5	0.04	2783.1	2785.6	0.04
9	tai044_50_10	3063	3063	3063.2	0.00	3064.1	3066.0	0.03	3064.1	3066.3	0.03	3067.8	3071.7	0.16	3067.8	3074.0	0.16
10	tai045_50_10	2976	2984	2997.5	0.27	2989.2	3002.9	0.18	2989.2	3003.1	0.18	2999.7	3012.8	0.53	3000.0	3013.6	0.54
11	tai046_50_10	3006	3006	3007.0	0.00	3011.8	3016.5	0.19	3011.8	3016.6	0.19	3018.3	3029.3	0.41	3018.3	3029.6	0.41
12	tai047_50_10	3093	3101	3114.8	0.26	3109.3	3121.7	0.27	3109.3	3122.0	0.27	3131.6	3134.1	0.99	3131.9	3134.4	1.00
13	tai052_50_20	3704	3714	3724.8	0.27	3717.1	3728.2	0.08	3717.1	3728.5	0.08	3726.0	3738.5	0.32	3727.2	3738.6	0.35
14	tai055_50_20	3610	3627	3640.4	0.47	3632.0	3648.1	0.14	3632.0	3648.4	0.14	3631.5	3646.6	0.13	3631.5	3646.6	0.13
15	tai062_100_5	5268	5268	5268.6	0.00	5268.3	5272.9	0.01	5268.3	5273.4	0.01	5271.1	5277.4	0.06	5271.1	5277.4	0.06
16	tai067_100_5	5246	5246	5246.0	0.00	5248.3	5251.9	0.04	5248.3	5252.2	0.04	5256.4	5263.8	0.20	5256.4	5265.2	0.20
17	tai078_100_10	5617	5624	5639.2	0.12	5635.2	5649.7	0.20	5635.2	5649.7	0.20	5646.3	5663.1	0.40	5646.3	5663.8	0.40
18	tai082_100_20	6183	6244	6256.3	0.99	6246.2	6262.7	0.03	6246.2	6263.0	0.03	6249.7	6269.6	0.09	6249.7	6269.9	0.09
19	tai087_100_20	6268	6316	6344.4	0.77	6333.0	6351.1	0.27	6333.0	6351.5	0.27	6351.8	6370.0	0.57	6351.8	6371.9	0.57
20	tai094_200_10	10,889	10,889	10892.6	0.00	10892.7	10896.6	0.03	10892.7	10896.7	0.03	10895.2	10906.0	0.06	10895.2	10908.0	0.06
21	tai097_200_10	10,854	10,860	10876.3	0.06	10867.9	10882.0	0.07	10867.9	10882.3	0.07	10878.2	10894.4	0.17	10878.2	10898.7	0.17
22	tai102_200_20	11,203	11,356	11366.5	1.37	11377.8	11401.6	0.19	11377.8	11401.8	0.19	11390.2	11417.5	0.30	11395.4	11418.8	0.35
23	tai103_200_20	11,281	11,411	11469.9	1.15	11420.6	11478.3	0.08	11420.6	11478.5	0.08	11462.0	11505.1	0.45	11462.0	11508.7	0.45
24	tai104_200_20	11,275	11,379	11416.0	0.92	11424.2	11458.7	0.40	11424.2	11459.7	0.40	11386.6	11427.4	0.07	11386.6	11429.1	0.07
25	tai105_200_20	11,259	11,340	11376.0	0.72	11361.1	11388.2	0.19	11361.1	11389.0	0.19	11375.3	11410.1	0.31	11375.3	11411.5	0.31
26	tai107_200_20	11,360	11,463	11477.9	0.91	11479.9	11506.5	0.15	11479.9	11506.7	0.15	11511.9	11532.5	0.43	11511.9	11533.1	0.43
27	tai108_200_20	11,334	11,455	11480.9	1.07	11470.0	11500.7	0.13	11470.0	11500.8	0.13	11497.0	11528.5	0.37	11497.0	11529.6	0.37
28	tai112_500_20	26,520	26,702	26778.4	0.69	26769.1	26803.6	0.25	26776.5	26804.5	0.28	26757.9	26837.3	0.21	26757.9	26838.9	0.21
29	tai113_500_20	26,371	26,500	26563.5	0.49	26514.6	26577.3	0.06	26514.6	26578.7	0.06	26559.4	26621.2	0.22	26559.4	26627.0	0.22
30	tai118_500_20	26,560	26,654	26747.7	0.35	26711.4	26788.7	0.22	26711.4	26788.8	0.22	26713.0	26810.9	0.22	26716.3	26813.4	0.23
	Averages				0.36			0.13			0.13			0.31			0.32

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

Table 2
Best and average results of the SIM-ESP after 10 runs for 30 randomly selected Taillard's instances – high variability scenarios.

No.	Instance	PFSP (deterministic makespan)				PFSPST – HIGH VARIANCE (expected stochastic makespan)											
						$k = 2$						$k = 5$					
		DM-BKS (1)	DM-BDS10 (2)	DM-ADS10	GAP (1)–(2) (%)	SM-BSS10 (7)	SM-ABSS10	GAP (2)–(7) (%)	SM-BDS10 (8)	SM-ADS10	GAP (2)–(8) (%)	SM-BSS10 (9)	SM-ASS10	GAP (2)–(9) (%)	SM-BDS10 (10)	SM-ADS10	GAP (2)–(10) (%)
1	tai007_20_5	1234	1234	1237.5	0.00	1248.1	1251.8	1.14	1248.1	1252.2	1.14	1252.7	1257.1	1.52	1252.9	1259.7	1.53
2	tai009_20_5	1230	1230	1230.0	0.00	1246.8	1250.2	1.37	1246.8	1250.2	1.37	1259.7	1264.8	2.41	1259.7	1265.1	2.41
3	tai010_20_5	1108	1108	1108.0	0.00	1125.9	1128.0	1.62	1126.6	1128.4	1.68	1141.5	1145.2	3.02	1141.5	1146.4	3.02
4	tai011_20_10	1582	1582	1582.0	0.00	1599.2	1600.9	1.09	1599.6	1601.6	1.11	1613.3	1616.4	1.98	1615.8	1618.2	2.13
5	tai013_20_10	1496	1496	1496.8	0.00	1511.9	1516.3	1.06	1511.9	1517.1	1.06	1519.6	1528.9	1.58	1525.3	1532.5	1.96
6	tai027_20_20	2273	2273	2273.0	0.00	2288.4	2290.0	0.68	2288.4	2290.0	0.68	2299.3	2303.1	1.16	2300.2	2304.9	1.20
7	tai036_50_5	2829	2829	2829.0	0.00	2834.4	2843.1	0.19	2837.0	2847.2	0.28	2838.2	2852.5	0.32	2852.4	2863.2	0.83
8	tai040_50_5	2782	2782	2782.0	0.00	2783.7	2790.6	0.06	2785.1	2794.2	0.11	2792.5	2802.1	0.38	2792.5	2807.5	0.38
9	tai044_50_10	3063	3063	3063.2	0.00	3079.0	3088.3	0.52	3079.5	3092.5	0.54	3093.6	3104.2	1.00	3093.6	3108.1	1.00
10	tai045_50_10	2976	2984	2997.5	0.27	3017.6	3035.4	1.13	3017.6	3037.2	1.13	3041.8	3061.0	1.94	3041.8	3065.5	1.94
11	tai046_50_10	3006	3006	3007.0	0.00	3036.4	3042.2	1.01	3036.4	3043.5	1.01	3059.6	3069.3	1.78	3059.6	3071.8	1.78
12	tai047_50_10	3093	3101	3114.8	0.26	3145.1	3157.8	1.42	3145.1	3159.3	1.42	3175.7	3185.3	2.41	3176.1	3187.9	2.42
13	tai052_50_20	3704	3714	3724.8	0.27	3733.3	3749.9	0.52	3733.6	3753.4	0.53	3759.1	3772.0	1.22	3763.5	3777.0	1.33
14	tai055_50_20	3610	3627	3640.4	0.47	3645.5	3662.4	0.51	3651.5	3665.0	0.68	3668.8	3683.6	1.15	3668.8	3686.4	1.15
15	tai062_100_5	5268	5268	5268.6	0.00	5285.5	5293.2	0.33	5285.9	5301.2	0.34	5295.5	5312.6	0.52	5295.5	5317.8	0.52
16	tai067_100_5	5246	5246	5246.0	0.00	5283.3	5290.8	0.71	5283.3	5294.1	0.71	5310.3	5319.9	1.23	5310.3	5323.6	1.23
17	tai078_100_10	5617	5624	5639.2	0.12	5691.0	5698.0	1.19	5691.0	5700.9	1.19	5713.9	5727.5	1.60	5718.7	5731.1	1.68
18	tai082_100_20	6183	6244	6256.3	0.99	6262.6	6279.1	0.30	6265.8	6281.7	0.35	6280.9	6304.3	0.59	6298.5	6314.8	0.87
19	tai087_100_20	6268	6316	6344.4	0.77	6364.5	6392.1	0.77	6366.8	6394.2	0.80	6398.3	6431.0	1.30	6413.6	6441.2	1.55
20	tai094_200_10	10,889	10,889	10892.6	0.00	10907.8	10924.8	0.17	10907.8	10931.6	0.17	10932.3	10955.4	0.40	10932.3	10965.4	0.40
21	tai097_200_10	10,854	10,860	10876.3	0.06	10894.9	10927.4	0.32	10894.9	10932.6	0.32	10934.1	10960.4	0.68	10934.1	10967.0	0.68
22	tai102_200_20	11,203	11,356	11366.5	1.37	11423.7	11454.0	0.60	11439.5	11457.5	0.74	11458.0	11487.7	0.90	11458.0	11493.7	0.90
23	tai103_200_20	11,281	11,411	11469.9	1.15	11490.1	11526.6	0.69	11496.2	11530.4	0.75	11531.4	11569.3	1.05	11541.7	11582.1	1.15
24	tai104_200_20	11,275	11,379	11416.0	0.92	11424.0	11477.2	0.40	11424.0	11482.5	0.40	11448.7	11498.9	0.61	11474.7	11512.6	0.84
25	tai105_200_20	11,259	11,340	11376.0	0.72	11432.4	11447.0	0.81	11435.4	11450.0	0.84	11468.2	11507.3	1.13	11477.9	11516.9	1.22
26	tai107_200_20	11,360	11,463	11477.9	0.91	11552.7	11572.3	0.78	11552.7	11575.5	0.78	11582.4	11615.4	1.04	11585.3	11620.2	1.07
27	tai108_200_20	11,334	11,455	11480.9	1.07	11541.4	11568.2	0.75	11544.0	11569.6	0.78	11586.3	11611.8	1.15	11601.2	11622.1	1.28
28	tai112_500_20	26,520	26,702	26778.4	0.69	26862.2	26903.7	0.60	26865.6	26912.8	0.61	26933.0	26993.3	0.87	26949.4	27012.1	0.93
29	tai113_500_20	26,371	26,500	26563.5	0.49	26610.2	26668.8	0.42	26613.3	26677.3	0.43	26683.2	26734.5	0.69	26684.9	26745.1	0.70
30	tai118_500_20	26,560	26,654	26747.7	0.35	26816.3	26871.4	0.61	26823.3	26883.2	0.64	26890.2	26972.4	0.89	26915.3	26980.6	0.98
	Averages				0.36			0.73			0.75			1.22			1.30

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

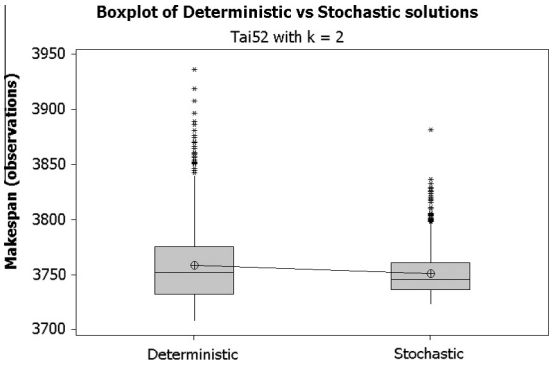

Fig. 9. Using simulation outputs to compare different solutions.

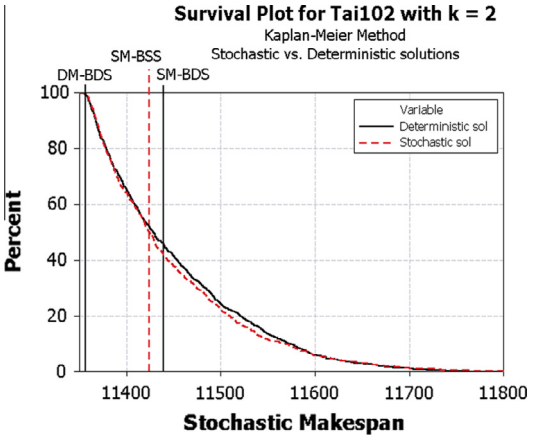

Fig. 10. Survival plot without intersecting solutions.

(Fig. 2), as uncertainty (variance) in the processing times increases, the expected makespan will also tend to increase. Also, notice that the original PFSP instances are a particular case of these new defined instances when $Var[P_{ij}] = 0$ for all $i \in J$ and $j \in M$.

A total of 30 instances were randomly selected from the Taillard's set described above. For each uncertainty scenario and for each selected instance, 10 independent iterations (replicas) were run. Each replica was run for a maximum time $t_{max} = n \times m \times 0.03\ s$, where n is the *number of jobs*, and m is the *number of machines* in the tested instance. Then, for each set of replicas, the best and average experimental solutions (permutations of jobs) found were registered. Also, the Best-Known Solution (BKS) for the PFSP associated with each instance was obtained either from the aforementioned website or from Zobolas et al. [68]. Notice that the t_{max} we are employing is really a small value in terms of computational times. Thus, for the smallest instances $t_{max} = 5$ s, while for the largest ones $t_{max} = 500$ s. For the different uncertainty scenarios considered, Tables 1 and 2 show the results obtained in this experiment, both for the PFSP and the PFSPST. Keeping in mind that a solution is just a permutation of jobs – both in the deterministic and the stochastic case, the following abbreviations are used:

- *DM-BKS:* Deterministic Makespan associated with the Best-Know Solution.
- *DM-BDS10/DM-ADS10:* Deterministic Makespan associated with our Best/Average Deterministic Solution after 10 runs of our algorithm.
- *SM-BDS10/SM-ADS10:* Stochastic Makespan associated with our Best/Average Deterministic Solution after 10 runs of our algorithm.
- *SM-BSS10/SM-ASS10:* Stochastic Makespan associated with our Best/Average Stochastic Solution after 10 runs of our algorithm.

From Tables 1 and 2, the following conclusions can be derived:

- Despite the limited time for each run, the algorithm is able to find high-quality solutions for the PFSP (average gap of 0.36% with respect to the BKS). This result allows to validate the algorithm employed to obtain deterministic solutions, which was a necessary step since our approach uses them to find candidate solutions for the PFSPST.

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

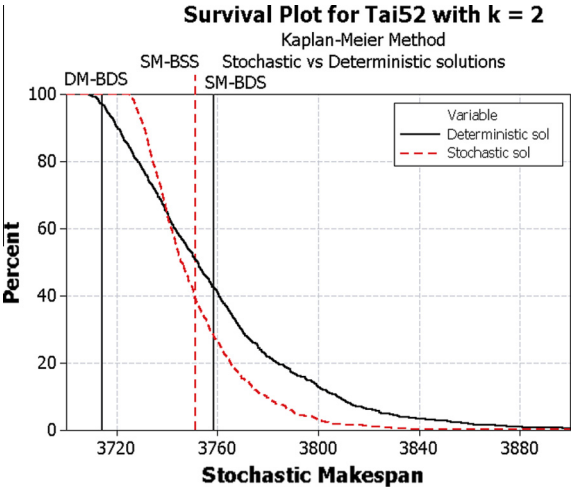

Fig. 11. Survival plot with intersecting solutions.

- As it should be expected for accurate estimates provided by long run simulations, the expected makespan associated to our best solution (for any uncertainty level) is always bounded by the deterministic makespan (lower bound) and the expected makespan (upper bound) values associated with the best deterministic solution.
- Likewise, as the uncertainty level increases the gap between the stochastic solutions for the PFSPST and the deterministic solution for the PFSP increases as well. Notice, however, that this gap is always relatively low, ranging from 0.13% for $k = 0.1$ to 1.22% for $k = 5$. Since the deterministic solution of the PFSP is also a lower bound for the optimum value of the PFSPST, this implies that our solution is relatively close to the optimal solution in all tested instances.
- For each instance-uncertainty level combination, the average value of the 10 runs is always relatively close to the best value obtained in those 10 runs. This implies that our algorithm is expected to perform quite well every time it is run.

6. Using reliability-based methods to compare different solutions

So far, we have been comparing solutions by using their respective expected makespan values. While this has been the most utilized criterion in the existing literature so far, decision-makers might be interested in considering additional information about the makespan associated with a given solution (permutation of jobs). Since this makespan is a random variable, a manager can ask him or herself questions such as: which is the variance and distribution of this makespan? and which is the probability of this makespan being lower than a given threshold?. Precisely, this is where historical data or, alternatively, simulation can make a difference by providing random observations of the makespan associated with a given solution. Fig. 9 shows a boxplot comparing makespans associated with the best deterministic solution and the best stochastic solution, respectively, for the *Tai52* instance with $k = 2$. Each of these two permutations is used to solve the stochastic version of the problem. Therefore, each time one of these permutations is applied in a scenario with stochastic processing times, a random makespan is obtained as an output. Thus, the boxplot illustrates differences between both solutions, not only in terms of expected makespan but also in distribution and variance of each set of outputs. Notice how the stochastic solution provided by our approach offers not only a smaller expected makespan but also smaller quartiles and variance.

Even more interesting, since a makespan represents the necessary time to complete all tasks – i.e., work total duration, it is possible to draw a parallelism here with reliability techniques, which refer to the analysis of duration times [44]. Reliability techniques might be preferred over classical statistical techniques, e.g., classical percentile or distribution-function plots. This might be especially true when dealing with historical data on random work duration, since some observations might be censored (incomplete data). Under the existence of censored data, reliability techniques can become a useful tool in order to provide valuable information about the survival (durability) function associated with each solution. Fig. 10 shows, for the *Tail102* instance with $k = 2$, the survival functions associated, respectively, with the best deterministic solution and the best stochastic solution. These functions have been obtained using the Kaplan–Meier estimator [65], which is a well-known non-parametric estimator in reliability analysis – when working with non-censored data, the Kaplan–Meier estimator basically returns the inverse of the empirical distribution function. First of all, the vertical lines show that $DM\text{-}BDS \leqslant SM\text{-}BSS \leqslant SM\text{-}BDS$, i.e., our stochastic solution is between the lower and upper bounds delimited by the deterministic solution. Also, notice that the survival function associated with the stochastic solution –generated from the observed stochastic makespans when using the stochastic solution – is always lower than the one associated with the deterministic solution – which has been generated from the observed stochastic makespans when using the deterministic solution. This means that the probability that the tasks are still ongoing (not finished yet) at any given deadline will be always lower when using the stochastic solution.

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

However, survival functions associated to different solutions can sometimes intersect in one or more points. This is illustrated in Fig. 11, which refers to the *Tai52* instance with $k = 2$. When two survival functions cross each other, then no solution is absolutely better than the other. As can be noticed in Fig. 11, while the deterministic solution shows lower probabilities of ongoing tasks for low threshold values (below 3740, approximately), for high threshold values is the stochastic solution the one offering lower probabilities of work not being finished yet. Of course, in some other cases intersection could happen the other way around, meaning that the solution with the lowest expected makespan will not offer the higher probabilities of work being completed before a given deadline located beyond the intersection point. This justifies the necessity of considering tools such as boxplots and survival plots in order to complement the limited information provided by the expected makespan value.

7. Conclusions and future work

In this paper we have presented a simheuristic algorithm, combining an Iterated Local Search metaheuristic with Monte Carlo simulation, to solve the Permutation Flow shop Problem with Stochastic Times (PFSPST). The basic idea behind our approach consists in: (a) to transform the initial stochastic problem into a deterministic one by considering constant processing times given by the expected values of the stochastic times; then (b) generate, using an efficient metaheuristic algorithm, a reduced set of high-quality solutions for the deterministic permutation Flow Shop problem – i.e., solutions with a low deterministic makespan; and (c) use Monte Carlo simulation to estimate the expected value of the stochastic makespan for each of the aforementioned solutions. By integrating simulation inside a metaheuristic, the methodology allows to solve the problem for any probability distribution – either theoretical or empirical – used to model job-machine processing times, even in the case these distributions differ from one job-machine pair to another. The efficiency of the proposed approach is discussed throughout a set of experiments based on adapted instances from well-known benchmarks for the deterministic version of the problem. Another contribution of the paper is to point out the analogy between work duration times and failure times, which makes it possible to employ techniques from survival analysis in order to better compare alternative solutions. This is especially interesting when real-life censored observations associated with a given permutation of jobs are used to compare different solutions. This might be the case, for instance, when the random makespan values associated with a given solution show a high variability. As a result, some realizations of the proposed solution might result in makespan values exceeding the duration of the experiment, thus providing censored or incomplete observations. All in all, the paper illustrates some of the benefits that can be attained when combining simulation with metaheuristics in solving combinatorial optimization problems with stochastic inputs. Regarding future work, we plan to analyze several variations of the algorithm presented in this paper. For instance, the algorithm could employ a stochastic-driven approach for updating the base solution. In other words, the base solution would be updated based on the value of the expected makespan instead of on the value of the deterministic makespan, as proposed in this paper. Another interesting variation to explore could be the use of criteria other than the makespan to guide the exploration of the solution space. This can be attained, for example, by integrating the reliability-based methods in the solution-evaluation process inside the metaheuristic.

Acknowledgements

This work has been partially supported by the Spanish Ministry of Science and Innovation (TRA2010-21644-C03). It has been developed in the context of the IN3-ICSO program and the CYTED-HAROSA network (http://dpcs.uoc.edu).

References

[1] I. Al Kattan, R. Maragoud, Performance analysis of flowshop scheduling using genetic algorithm enhanced with simulation, Int. J. Ind. Eng. – Theory, Applic. Pract. 15 (2008) 62–72.
[2] H. Allaoui, S. Lamouri, M. Lebbar, A robustness framework for a stochastic hybrid flow shop to minimize the makespan, Proc. Int. Conf. Service Syst. Service Manage. (2006) 1097–1102.
[3] S. Andradóttir, An overview of simulation optimization via random search, in: S.G. Henderson, B.L. Nelson (Eds.), Handbooks in Operations Research and Management Science, vol. 13, Elsevier, 2006, pp. 617–631.
[4] E. Angelidis, D. Bohn, O. Rose, A simulation-based optimization heuristic using self-organization for complex assembly lines, in: Proceedings of the 2012 Winter Simulation Conference, 2012, pp. 1231–1240.
[5] A. Anglani, A. Grieco, E. Guerriero, R. Musmanno, Robust scheduling of parallel machines with sequence-dependent set-up costs, Eur. J. Oper. Res. 161 (2005) 704–720.
[6] M. Arakawa, M. Fuyuki, I. Inoue, An optimization-oriented method for simulation-based job shop scheduling incorporating capacity adjustment function, Int. J. Prod. Econ. 85 (3) (2003) 359–369.
[7] C. Artigues, J.C. Billaur, C. Esswein, Maximization of solution flexibility for robust shop scheduling, Eur. J. Oper. Res. 165 (2005) 314–328.
[8] C. Azzaro-Pantel, L. Bernal-Haro, P. Baudet, S. Domenech, L. Pibouleau, A two-stage methodology for short-term batch plant scheduling: discrete-event simulation and genetic algorithm, Comput. Chem. Eng. 22 (10) (1998) 1461–1481.
[9] P.C. Bagga, N-Job, 2-machine sequencing problem with stochastic service, Oper. Res. 7 (1970) 184–197.
[10] K.R. Baker, D. Altheimer, Heuristic solution methods for the stochastic flow shop problem, Eur. J. Oper. Res. 216 (2012) 172–177.
[11] K.R. Baker, D. Trietsch, Three heuristic procedures for the stochastic, two-machine flow shop problem, J. Sched. 14 (2011) 445–454.
[12] B.P. Banerjee, Single facility sequencing with random execution times, Oper. Res. 13 (1965) 358–364.
[13] H.V. Bassi, E. Filho, L. Bahiense, Planning and scheduling a fleet of rigs using simulation–optimization, Comput. Ind. Eng. 63 (4) (2012) 1074–1088.
[14] A. Bonfill, A. Espuña, L. Puigjaner, Proactive approach to address the uncertainty in short-term scheduling, Comput. Chem. Eng. 32 (2008) 1689–1706.
[15] H.G. Campbell, R.A. Dudek, M.L. Smith, A heuristic algorithm for the n job, m machine sequencing problem, Manage. Sci. 16 (1970) 630–637.
[16] T. Chaari, S. Chaabane, T. Loukil, D. Trentesaux, A genetic algorithm for robust hybrid flow shop scheduling, Int. J. Comput. Integr. Manuf. 24 (2011) 821–833.

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005

[17] S.H. Choi, K. Wang, Flexible flow shop scheduling with stochastic processing times: a decomposition-based approach, Comput. Ind. Eng. 63 (2012) 362–373.
[18] A.A. Cunningham, S.K. Dutta, Scheduling jobs, with exponentially distributed processing times, on two machines of a flow shop, Naval Res. Logis. 20 (1973) 69–81.
[19] L. Deroussi, M. Gourgand, N. Tchernev, Coupling local search methods and Simulated annealing to the job shop scheduling problem with transportation, in: Proceedings of the 8th IEEE International Conference on Emerging Technologies and Factory Automation, 2001, pp. 659–667.
[20] B. Dodin, Determining the optimal sequences and the distributional properties of their completion times in stochastic flow shops, Comput. Oper. Res. 23 (1996) 829–843.
[21] J.W. Fowler, L. Mönch, O. Rose, Scheduling and simulation, Handbook of Production Scheduling. International Series in Operations Research & Management Science, vol. 89, Springer, US, 2006, pp. 109–133.
[22] J.M. Framinan, J.N.D. Gupta, R. Leisten, A review and classification of heuristics for permutation flow shop scheduling with makespan objective, J. Oper. Res. Soc. 55 (2004) 1243–1255.
[23] M. Frantzén, A. Ng, P. Moore, A simulation-based scheduling system for real-time optimization and decision making support, Robot. Comput.-Integrated Manuf. 27 (4) (2011) 696–705.
[24] M.R. Garey, D.S. Johnson, R. Sethi, The complexity of flow shop and job shop scheduling, Math. Oper. Res. 1 (2) (1976) 117–129.
[25] F. Ghezail, H. Pierreval, S. Hajri-Gabouj, Analysis of robustness in proactive scheduling: a graphical approach, Comput. Ind. Eng. 58 (2010) 193–198.
[26] M. Gourgand, N. Grangeon, S. Norre, A contribution to the stochastic flow shop scheduling problem, Eur. J. Oper. Res. 151 (2003) 415–433.
[27] M. Gourgand, N. Grangeon, S. Norre, Markovian analysis for performance evaluation and scheduling in m machine stochastic flow shop with buffers of any capacity, Eur. J. Oper. Res. 161 (2005) 126–147.
[28] S.J. Honkomp, L. Mockus, G.V. Reklaitis, Robust scheduling with processing time uncertainty, Comput. Chem. Eng. 21 (1997) 1055–1060.
[29] H. Hu, H. Zhang, A simulation-based two-stage scheduling methodology for controlling semiconductor wafer fabs, Expert Syst. Applic. 39 (14) (2012) 11677–11684.
[30] N.P. Jewell, A.C. Kimber, M.T. Lee, G.A. Whitmore, Lifetime Data: Models in Reliability and Survival Analysis, Springer, 1996.
[31] A. Juan, J. Faulin, J. Jorba, D. Riera, D. Masip, B. Barrios, On the use of Monte Carlo simulation, cache and splitting techniques to improve the Clarke and Wright savings heuristics, J. Oper. Res. Soc. 62 (2011) 1085–1097.
[32] A. Juan, H. Lourenço, M. Mateo, R. Luo, Q. Castella, Using iterated local search for solving the flow-shop problem: parametrization, randomization and parallelization issues, Int. Trans. Oper. Res. 21 (1) (2014) 103–126.
[33] P.J. Kalczynski, J. Kamburowsky, Generalization of Johnson's and Talwar's scheduling rules in two-machine stochastic flow shops, J. Oper. Res. Soc. 55 (2004) 1358–1362.
[34] J. Kamburowski, Stochastically minimizing the makespan in two machine flow shops without blocking, Eur. J. Oper. Res. 112 (1999) 304–309.
[35] J. Kamburowski, On three-machine flow shops with random job processing times, Eur. J. Oper. Res. 125 (2000) 440–449.
[36] K. Katragjini, E. Vallada, R. Ruiz, Flowshop rescheduling under different type of disruptions, Int. J. Prod. Res. 51 (3) (2013) 780–797.
[37] K. Kianfar, S.M.T. Fatemi-Ghomi, A.O. Jadid, Study of stochastic sequence-dependent flexible flow shop via developing a dispatching rule and a hybrid GA, Eng. Applic. Artif. Intell. 25 (2012) 494–506.
[38] T. Kima, B.K. Choi, Production system-based simulation for backward on-line job change scheduling, Simul. Model. Pract. Theory 40 (2014) 12–27.
[39] A. Klemmt, S. Horn, G. Weigert, K. Wolter, Simulation-based optimization vs. mathematical programming: a hybrid approach for optimizing scheduling problems, Robot. Comput.-Integrated Manuf. 25 (6) (2009) 917–925.
[40] C. Laroque, A. Klaas, J.H. Fischer, M. Kuntze, Fast converging, automated experiment runs for material flow simulations using distributed computing and combined metaheuristics, in: Proceedings of the 2012 Winter Simulation Conference, 2012, pp. 102–111.
[41] Q. Liu, S. Ullah, C. Zhang, An inproved genetic algorithm for robust permutation flowshop scheduling problem, Int. J. Adv. Manuf. Technol. 56 (2011) 345–354.
[42] H.R. Lourenço, O. Martin, T. Stützle, Iterated local search: framework and applications, in: Handbook of Metaheuristics, International Series in Operations Research and Management Science, vol. 146, Kluwer Academic Publishers, 2010, pp. 363–397.
[43] T. Makino, On a scheduling problem, J. Oper. Res. Soc. Jpn. 8 (1965) 32–44.
[44] M. Modarres, M. Kaminskiy, V. Krivtsov, Reliability Engineering and Risk Analysis: A Practical Guide, Marcel Dekker, Inc., New York, 1999.
[45] M. Nawaz, J.E. Enscore, I. Ham, A heuristic algorithm for the m-machine, n-job flowshop sequencing problem, Omega 11 (1983) 91–95.
[46] H. Pierreval, S. Durieux-Paris, Robust simulation with a base environmental scenario, Eur. J. Oper. Res. 182 (2007) 783–793.
[47] M. Pinedo, Minimizing the expected makespan in stochastic flow shops, Oper. Res. 30 (1982) 148–162.
[48] I. Ribas, R. Companys, X. Tort-Martorell, Comparing three-step heuristics for the permutation flow shop problem, Comput. Oper. Res. 37–12 (2010) 2062–2070.
[49] B. Roy, Robustness in operational research and decision aiding: a multi-faceted issue, Eur. J. Oper. Res. 200 (2010) 629–638.
[50] R. Ruiz, C. Maroto, A comprehensive review and evaluation of permutation flow shop heuristics, Eur. J. Oper. Res. 165 (2005) 479–494.
[51] R. Ruiz, T. Stützle, A simple and effective iterated greedy algorithm for the permutation flow shop scheduling problem, Eur. J. Oper. Res. 177 (2007) 2033–2049.
[52] D. Sturrock, New solutions for production dilemmas, Ind. Eng. (December) (2012) 47–52.
[53] S. Suresh, R.D. Foley, S.D. Dickey, On Pinedo's conjecture for scheduling in a stochastic flowshop, Oper. Res. 33 (1985) 1146–1153.
[54] E. Taillard, Some efficient heuristic methods for the flow shop sequencing problem, Eur. J. Oper. Res. 47 (1990) 65–74.
[55] E. Taillard, Benchmarks for basic scheduling problems, Eur. J. Oper. Res. 64 (1993) 278–285.
[56] E. Talbi, Metaheuristics: From Design to Implementation, Wiley, 2009.
[57] P.P. Talwar, A note on sequencing problems with uncertain job times, J. Oper. Res. Soc. Jpn. 9 (1967) 93–97.
[58] E. Vallada, R. Ruiz, G. Minella, Minimising total tardiness in the m-machine flowshop problem: a review and evaluation of heuristics and metaheuristics, Comput. Oper. Res. 35 (2008) 1350–1373.
[59] N.M. van Dijk, E. van der Sluis, Practical optimization by OR and simulation, Simul. Model. Pract. Theory 16 (2008) 1113–1122.
[60] R.R. Weber, Scheduling jobs with stochastic processing requirements on parallel machines to minimize makespan or flowtime, J. Appl. Probab. 19 (1982) 167–182.
[61] L. Wang, L. Zhang, D.Z. Zheng, A class of hypothesis-test-based genetic algorithms for flow shop scheduling with stochastic processing times, Int. J. Adv. Manuf. Technol. 25 (2005) 1157–1163.
[62] L. Wang, L. Zhang, D.Z. Zheng, Genetic ordinal optimization for stochastic flow shop scheduling, Int. J. Adv. Manuf. Technol. 27 (2005) 166–173.
[63] S.H. Wie, M. Pinedo, On minimizing the expected makespan and flowtime in stochastic flow shop with blocking, Math. Oper. Res. 11 (1986) 336–342.
[64] X. Xu, W. Cui, J. Lin, Y. Qian, Robust makespan minimisation in identical parallel machine scheduling problema with interval data, Int. J. Prod. Res. 51 (2013) 3532–3548.
[65] L.C. Wolstenholme, Reliability Modelling: A Statistical Approach, Chapman & Hall/CRC, Boca Raton, FL, 1999.
[66] L. Zhang, L. Wang, F. Tang, Hypothesis test based simulated annealing for stochastic flow shop scheduling, Proc. Second Int. Conf. Mach. Learn. Cyber. (2003) 1607–1612.
[67] Q. Zhou, X. Cui, Research on multiobjective flow shop scheduling with stochastic processing times and machine breakdowns, Proc. Int. Conf. Service Oper. Logis. Informatics (2008) 1718–1724.
[68] C. Zobolas, C. Tarantilis, G. Ioannou, Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm, Comput. Oper. Res. 36 (2009) 1249–1267.

Please cite this article in press as: A.A. Juan et al., A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times, Simulat. Modell. Pract. Theory (2014), http://dx.doi.org/10.1016/j.simpat.2014.02.005