A Hybrid Estimation of Distribution Algorithm for Simulation-Based Scheduling in a Stochastic Permutation Flowshop

K. Wang[a], S.H. Choi[b], H. Lu[c]

a Department of Management Science and Engineering, Economics and Management School, Wuhan University, Wuhan, China
b Department of Industrial and Manufacturing Systems Engineering, The University of Hong Kong, Pokfulam Road, Hong Kong
c Department of Logistics Management, School of Logistics Engineering, Wuhan University of Technology, Wuhan, China

Abstract

The permutation flowshop scheduling problem (PFSP) is NP-complete and tends to be more complicated when considering stochastic uncertainties in the real-world manufacturing environments. In this paper, a two-stage simulation-based hybrid estimation of distribution algorithm (TSSB-HEDA) is presented to schedule the permutation flowshop under stochastic processing times. To deal with processing time uncertainty, TSSB-HEDA evaluates candidate solutions using a novel two-stage simulation model (TSSM). This model first adopts the regression-based meta-modelling technique to determine a number of promising candidate solutions with less computation cost, and then uses a more accurate but time-consuming simulator to evaluate the performance of these selected ones. In addition, to avoid getting trapped into premature convergence, TSSB-HEDA employs both the probabilistic model of EDA and genetic operators of genetic algorithm (GA) to generate the offspring individuals. Enlightened by the weight training process of neural networks, a self-adaptive learning mechanism (SALM) is employed to dynamically adjust the ratio of offspring individuals generated by the probabilistic model. Computational experiments on Taillard's benchmarks show that TSSB-HEDA is competitive in terms of both solution quality and computational performance.

Keywords: permutation flowshop scheduling; stochastic processing times; estimation of distribution algorithm; genetic algorithm; meta-model.

1. Introduction

The permutation flowshop scheduling problem (PFSP) is a well-known and well-studied combinatorial optimisation problem (Gupta and Stafford Jr, 2006; Vallada and Ruiz, 2009). In the classical PFSP, jobs arrive at the shop floor simultaneously and then follow the same processing order on each of the machines. The PFSP has been proven strongly NP-complete for

more than two machines (Garey, 1976). Due to its great significance in both academic and real-world applications, the PFSP has attracted considerable attention after the pioneering work of Johnson (1954).

Although a tremendous amount of effort has been devoted to addressing the PFSP, most of the research works consider a static environment, in which no unexpected events would occur to disturb job processing. Real-world manufacturing environments, however, tend to suffer a variety of uncertainties, including change of processing time, machine breakdown, rush orders, and job cancellations, etc. (Gholami et al., 2009; Ouelhadj and Petrovic, 2009). Therefore, permutation flowshop scheduling under uncertainties has recently received an increasing attention.

Three types of approaches, namely exact algorithms, heuristics, and meta-heuristics, are commonly adopted to solve the PFSPs in the literature (Ruiz and Maroto, 2005; Xu et al., 2014). Exact algorithms aim to achieve the optimal solution, and hence are computationally expensive for large-sized PFSPs. Examples of such methods are branch and bound approaches (Chung et al., 2002). In addition to exact algorithms, heuristics and meta-heuristics have also been introduced to find approximate solutions within reasonable computational cost. Since most existing heuristic methods, such as constructive heuristics and improvement heuristics, tend to perform poorly on large-sized PFSPs (Ceberio et al., 2014), a wide range of meta-heuristics have been applied to address the PFSPs (Zobolas et al., 2009).

To deal with uncertainties in a flowshop, the simulation-based meta-heuristics (SBM) have been successfully developed to construct and evaluate candidate solutions. In these approaches, a discrete-event simulator is usually incorporated into a meta-heuristic (Wang et al., 2013), such as genetic algorithm (GA) (Dugardin et al., 2010), immune algorithm (Zandieh and Gholami, 2009), ant colony optimisation (ACO) algorithm (Ahmadizar et al., 2010), and hybrid meta-heuristics (Safari and Sadjadi, 2011), etc. As an iterative procedure, the meta-heuristic guides its subordinate heuristics to iteratively produce high-quality candidate solutions until a termination criterion is met. In the SBM, the performance of candidate solutions is estimated over iterations using the discrete-event simulator. Accordingly, the computation time of such an evaluation process inevitably greatly increases with the growth of the number of candidate solutions or simulation replications. The main disadvantage of SBM technique therefore lies in the large computation time required for performance evaluation under uncertainties (Dugardin et al., 2010).

To overcome such drawback, some effective approaches have recently been proposed for scheduling under uncertainties. Instead of estimating the performance of all candidate solutions,

these approaches only evaluate a number of promising candidate solutions by a time-consuming simulator. Zhang et al. (2012) developed a hybrid particle swarm optimisation (PSO) algorithm for stochastic job shop scheduling problems. They first adopted the lower bound of the objective value to give a quick performance evaluation on candidate solutions, and then only the ones in the satisfactory regions were further estimated using a discrete-event simulator. Moreover, to determine the promising candidate solutions for further optimisation under uncertainties, both Horng et al. (2012) and Juan et al. (2014) applied the stochastic simulation model with less simulation replications for performance evaluation.

Despite an increasing amount of research interest in developing new SBMs, few research works have been reported on improving their computational performance for scheduling problems under uncertainties.

This paper therefore presents an effective SBM to address the PFSPs under stochastic processing times. Enlightened by the works of Zhang's (2012), Horng's (2012), and Juan et al. (2014), we incorporate an efficient two-stage simulation-based model into a hybrid estimation of distribution algorithm (EDA) to generate good-quality schedules with less computational effort.

The EDA was first introduced by Mühlenbein and Paass (1996) as an alternative to conventional evolutionary algorithms (EA). Different from conventional EAs, EDA adopts a probabilistic model to generate the offspring. This model is established by learning from an elite set of individuals in the previous population. As an effective method to inherit good genes over generations, EDA has recently been successfully used to a wide range of combinatorial optimisation problems (Hauschild and Pelikan, 2011), such as the PFSP and its variants. Jarboui et al. (2009) presented an efficient EDA to solve the PFSP with total flow time minimisation. For the same scheduling problem, Zhang and Li (2011) improved the EDA efficiency by incorporating the longest common subsequence into the probabilistic model. Wang et al. (2013) developed an effective EDA to minimise the makespan of the distributed permutation flowshop. More recently, Ceberio et al. (2014) introduced a probabilistic distance-based ranking exponential model, named the Mallows model, to construct EDA solutions. To further investigate the performance of EDA for the PFSP, hybridisation of EDA with other meta-heuristics has also been studied. Liu et al. (2011) hybridised EDA with PSO to allow social information sharing among candidate solutions. Moreover, Tzeng et al. (2012) incorporated the idea of ant colony system (ACS) into EDA to schedule a permutation flowshop.

The proposed two-stage simulation-based hybrid EDA (TSSB-HEDA) differentiates itself

from the conventional EDA by two mechanisms, namely a two-stage simulation model (TSSM) and a self-adaptive learning mechanism (SALM). To reduce the computation cost of TSSB-HEDA, TSSM first employs a regression-based meta-model to provide a rough estimation of candidate solutions, and only a number of promising ones are identified and further evaluated using a discrete-event simulator. Moreover, to prevent EDA from early search stagnation, TSSB-HEDA employs both the probabilistic model of EDA and genetic operators of GA to produce offspring individuals. Motivated by the idea of neural network training, SALM dynamically adjusts the ratio of offspring generated by the probabilistic model to avoid being trapped into premature convergence. An extensive search of literature on PFSP suggests that not much research effort has been devoted to applying EDA to schedule the permutation flowshop under uncertainties.

The rest of the paper is organised as follows. Section 2 presents the mathematical formulation of PFSP. Section 3 describes the proposed TSSB-HEDA in details. To validate the performance of TSSB-HEDA under stochastic processing times, simulations are conducted and the computation results are analysed in Section 4. Finally, in Section 5, we conclude the paper and discuss some topics for future research.

2. Problem description

The PFSP is a well-known combinatorial optimisation problem. In the classical PFSP, a finite set J = {1, 2, … , n} of n jobs are firstly released simultaneously to the shop floor, and then are processed on a finite set M = {m_1, m_2,… , m_m} of m machines with no pre-emption allowed. Each job j, j∈J, consists of m operations that have to be processed on the machines in the order of m_1, m_2,… , m_m. All the jobs have deterministic processing times and follow the same processing order on each machine.

In the real-world manufacturing environments, however, a variety of unexpected events, such as tool wear, equipment failure, operator unavailability, and quality issues, may lead to uncertain processing times (Lawrence and Sewell, 1997). This paper describes the processing time uncertainty using the level of processing time variation (LPTV), which is described as follows:

$$LPTV = \sigma / E(P) \tag{1}$$

where E(P) and σ indicate the expected value and the standard deviation of processing time, respectively. According to formula (1), a larger LPTV may result in a large deviation between the expected and the actual processing times. For example, suppose E[P] of a job equals 15

time units, LPTV values of 0.2 and 0.4 lead the standard deviation of actual processing time from E[P] to be 3 and 6 times units respectively.

The objective of PFSP in this study is to determine a feasible permutation π to minimise the makespan, i.e. the maximum completion time of all operations. With consideration of processing time uncertainty, we formulate the PFSP as follows:

$$min\left\{E\left[C(\pi_n, m)\right]\right\} \tag{2}$$

Subject to the following constraints:

$$C(\pi_1,1) = SP(\pi_1,1) \tag{3}$$

$$C(\pi_j,1) = C(\pi_{j-1},1) + SP(\pi_j,1),\ \ j = 2,...,n \tag{4}$$

$$C(\pi_1,i) = C(\pi_1,i-1) + SP(\pi_1,i),\ \ i = 2,...,m \tag{5}$$

$$C(\pi_j,i) = max\left\{C(\pi_{j-1},i), C(\pi_j,i-1)\right\} + SP(\pi_j,i)\ ,\ \ i = 2,...,m;\ j = 2,...,n \tag{6}$$

Where

i: machine index, $1 \le$ i $\le$ m

j: job index, $1 \le$ j $\le$ n

π_j: j^{th} job in permutation $\pi = \{\pi_1,...,\pi_n\}$, $1 \le$ j $\le$ n

$C(\pi_j,i)$: completion time of Job π_j on machine i

$SP(\pi_j,i)$: stochastic processing time of Job π_j on machine i

3. Details of the proposed two-stage simulation-based hybrid EDA (TSSB-HEDA)

3.1 The framework of TSSB-HEDA

As a relatively new paradigm of EAs, EDA reproduces the offspring from a probabilistic model, rather than crossover and mutation operators used in traditional GAs. The probabilistic model describes statistical information of elite individuals from previous generations, and therefore is capable of predicting the most promising search area. As an iterative procedure, EDA includes the following steps (Pan and Ruiz, 2012): (1) Randomly generate an initial population; (2) Choose some good individuals to construct the elite set; (3) Establish the probabilistic model from the elite set; (4) Generate new individuals from the estimated probabilistic model; (5) Steps 2-4 are repeated until a stopping criterion is satisfied.

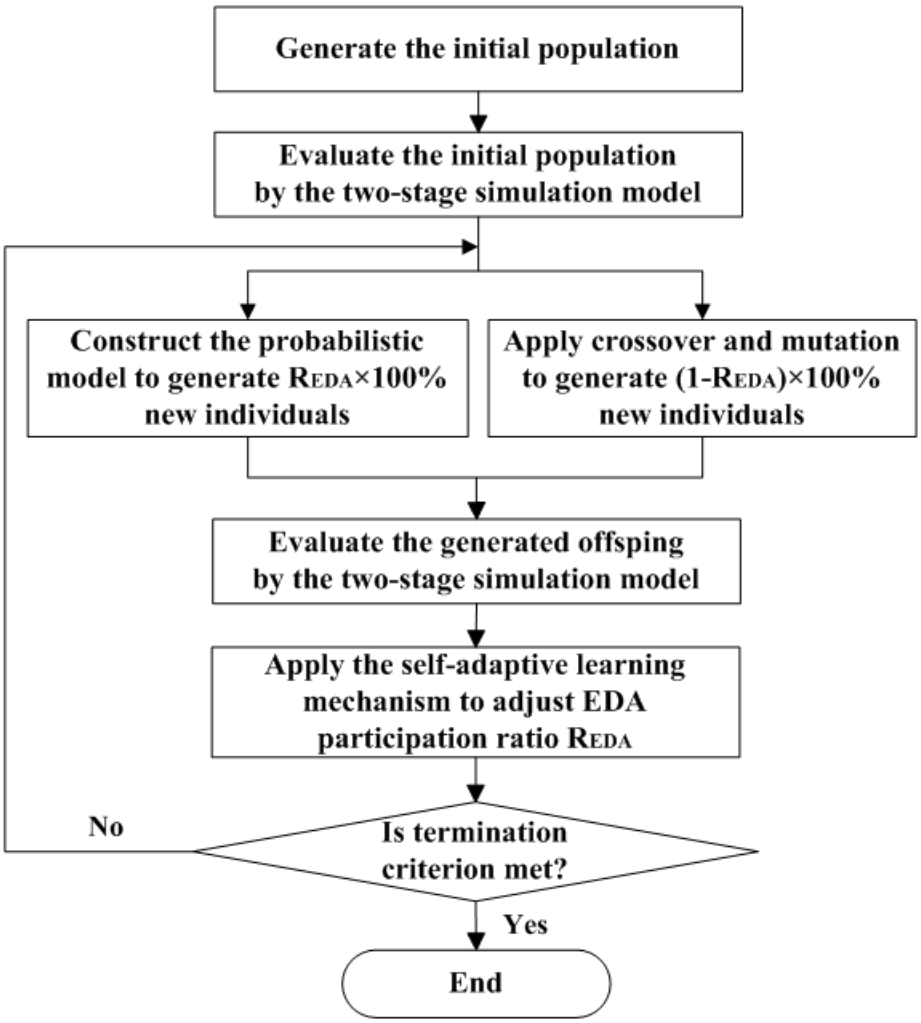


Fig. 1. The framework of TSSB-HEDA

Since new populations are generated entirely from the probabilistic model, EDA may not generate diversified individuals. It tends to trap into premature convergence after some generations (Chen et al., 2010). Incorporation of meta-heuristics, particularly GA, might be an effective approach to avoiding premature convergence. Although the genetic operators of GA, such as crossover and mutation, can provide the population diversity by perturbing the good solution structure, few studies on the integration of GA with EDA have been reported. Some existing hybrid frameworks mainly include ACGA (Chang et al., 2009) and eACGA (Chen et al., 2012). In these hybrid approaches, the probabilistic model of EDA and genetic operators of GA are alternated to produce the offspring over a predefined number of generations. No adaptive strategy is adopted to guide the process of population generation.

Fig. 1 illustrates the framework of the proposed TSSB-HEDA, which is differentiated from the conventional EDA by the two-stage simulation model (TSSM) and hybridisation of EDA and GA. To provide an efficient method for performance evaluation, TSSB-HEDA incorporates

TSSM into EDA to estimate the performance of offspring individuals under stochastic processing times. Furthermore, to preserve the population diversity, both the probabilistic model of EDA and genetic operators of GA are applied to create new individuals in this study. Enlightened by the weight training process of neural networks, the EDA participation ratio R_{EDA}, indicating the ratio of individuals generated by the probabilistic model, is dynamically adjusted by a self-adaptive learning mechanism (SALM). The rest of Section 3 first provides a detailed explanation of TSSM and the hybrid EDA with genetic operators, and subsequently follows the complete procedure of TSSB-HEDA.

3.2 The Two-stage simulation model (TSSM) under stochastic processing times

TSSB-HEDA applies the TSSM to reduce the computation cost of performance evaluation considering processing time uncertainty. In the first stage, a regressing-based meta-model is applied to estimate the performance of candidate solutions, so that a number of promising ones can be quickly determined. From these selected solutions, a discrete-event simulator is then used in the second stage to obtain a more accurate evaluation.

3.2.1 Stage I: using the regression-based meta-model for performance evaluation

In the real-world manufacturing systems, processing time uncertainty may advance or postpone job processing, and in turn lead to deviations between the planned and the actual schedules. The effect of schedule deviations, if not be properly compensated by the slack or idle time on the machines, could eventually degrade schedule performance (Saad, 2003; Wang and Choi, 2014). Accordingly, less slack or idle time on the machines may result in significant degradation of schedule performance. Moreover, the configuration of permutation flowshop systems, including the number of jobs to be processed, the number of machines, and LPTVs of machines may influence the performance degradation of the planned schedule. To estimate the schedule performance in a realistic permutation flowshop, there is therefore a need to develop an effective method to predict the performance degradation of planned schedules.

Simulation modelling and analysis on manufacturing systems are often complicated, time-consuming, and challenging. Due to the robust and fast decision support in the decision-making process, meta-modelling techniques, including regression and neural network models, have been widely applied to predict the simulation results in real-world manufacturing environments. Instead of an actual simulation model, a meta-model approximates a functional relationship between simulation input parameters and system responses (Yu and Popplewell, 1994; Vinod and Sridharan, 2011). Since regression analysis is one of the commonly used

methods for finding such functional relationship, it has been applied to establish the meta-model to evaluate the schedule performance in this study.

To obtain a number of promising candidate solutions, the multiple linear regression meta-model is firstly applied to estimate the degradation of schedule performance (DSP) with processing time uncertainty. The makespan of the actual schedule, i.e. M_{AS}, is then computed as follows:

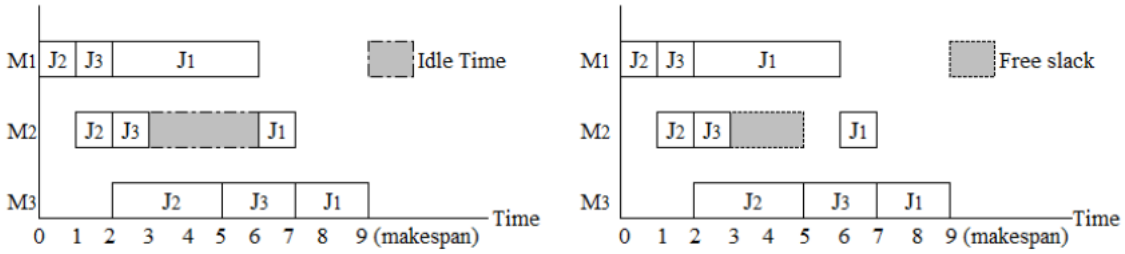
$$M_{AS} = M_{PS} \times DSP \tag{7}$$

where M_{PS} represents the makespan of the planned schedule. Based on M_{AS}, the promising candidate solutions can be easily identified.

The DSP of a planned schedule may be affected by machine size, job size, LPTV, and the slack ratio (SR). The SR, describing the slack per processing time of a planned schedule, is measured by

$$SR = \sum_{i=1}^{m}\sum_{j=1}^{n} S_{ij} / P_{ij} \tag{8}$$

where m denotes the number of machines, n represents the number of jobs to be processed, S_{ij} and P_{ij} indicate the free slack and the processing time of job j on machine i respectively.



(a) Idle time in a schedule (b) Free slack in a schedule

Fig. 2. Comparison of idle time and free slack in a schedule

In the literature of scheduling under uncertainties, free slack is commonly applied to estimate the robustness of a given schedule (Xiong et al., 2013). It is therefore used in this study to define the SR. Different from idle time, free slack in a permutation flowshop is measured by the amount of time that an operation could be right-shifted without delaying its start on the next machine. The difference between free slack and idle time of a given schedule is depicted in Fig. 2.

To predict the DSP of a planned schedule, we adopt job size, machine size, LPTV, and SR as the independent variables, and the multiple linear regression meta-model is accordingly established as follows:

$$DSP = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4 + \beta_5 x_1 x_2 + \beta_6 x_1 x_3 + \beta_7 x_1 x_4 + \beta_8 x_2 x_3 + \beta_9 x_2 x_4 + \beta_{10} x_3 x_4 + e \quad (9)$$

where x_1, x_2, x_3, and x_4 represent job size, machine size, LPTV, and SR; β_0 is the constant; β_1, β_2, β_3, and β_4 denote the coefficients corresponding to job size, machine size, LPTV, and SR; $\beta_5, \beta_6, \ldots, \beta_{10}$ are the coefficients corresponding to the interaction effects between job size, machine size, LPTV, and SR; e is the error.

The meta-model is often constructed in a three-step process (Hurrion and Birgil, 1999; Vinod and Sridharan, 2009). First, a factorial experimental design is conducted to obtain a set of simulation results. The meta-model is then established using either regression or neural network techniques. Finally, a validation test on the developed meta-model is conducted by comparing predicted results with simulation results. In this study, the simulation results under stochastic processing times (to be presented in Section 4.2) are used to build the meta-model shown in formula (9). After obtaining the multiple linear regression meta-model, the performance of candidate solutions can be roughly estimated according to formula (7), and accordingly the best $\alpha \times 100\% \left(\alpha \in [0,1]\right)$ ones can be chosen to establish a set of promising solutions Π_p for further performance evaluation.

3.2.2 Stage II: using the discrete-event simulator for performance evaluation

In this stage, only the promising solutions Π_p are further estimated using a discrete-event simulator. To model the manufacturing process in a permutation flowshop, processing time uncertainty is considered in this simulator, which is described below.

Algorithm I: The discrete-event simulator

Step 1: Set the number of simulation replications N_{sim} and counter $n = 1$.

Step 2: Initialise the level of processing time variation ($LPTV_k$) for each machine M_k.

Step 3: Allocate the jobs to machines according to the job sequence of a candidate solution.

Step 4: Following the processing route in a permutation flowshop (i.e. from the first machine to the last machine), the actual completion times of jobs on each machine M_k are obtained as follows:

Step 4.1: Identify the first unprocessed job j on machine M_k and record its start time.

Step 4.2: Apply a truncated normal distribution (to be detailed in Section 4.1) to generate actual processing time of job j on machine M_k.

Step 4.3: Obtain the completion time of job j on machine M_k by summing its start time with its actual processing time.

Step 4.4: If all the jobs assigned to machine M_k have been completed, record their completion times; otherwise, return to step 4.1.

Step 5: Record the simulated makespan under stochastic processing times as M_{sim}^{n}, i.e. the completion time of the last job on the last machine.

Step 6: If $n > N_{sim}$, return the average simulated makespan over N_{sim} simulation replications as $M_{avg} = \sum_{n=1}^{N_{sim}} M_{sim}^{n} \Big/ N_{sim}$; otherwise, set $n = n+1$ and return to step 3.

Based on the above simulator, M_{avg} is used to evaluate the candidate solutions of Π_p. To model the manufacturing process in a permutation flowshop, this simulator employs a large $N_{sim} = 100$ for performance evaluation under stochastic processing times, inevitably resulting in a costly and time-consuming evaluation process.

3.3 The hybrid EDA with genetic operators

TSSB-HEDA employs both TSSM and genetic operators to address the PFSPs with processing time uncertainty. The details of TSSB-HEDA are described as follows.

3.3.1 Solution encoding and initial population

TSSB-HEDA applies the permutation-based encoding scheme to represent individual solutions. Such a method has been widely adopted for permutation flowshop scheduling during the past decades (Wang et al., 2010; Gao et al., 2011; Li and Pan, 2014). For example, as a solution to a PFSP with 4 jobs, job sequence {1, 4, 2, 3} shows that job 1 is first scheduled, then successively followed by job 4, job 2, and job 3. To better cover the promising regions of the entire search space, TSSB-HEDA generates the initial population randomly.

3.3.2 Selection

The probabilistic model of EDA is established based on an elite set of individuals, which are selected from previous population using a selection operator. To reduce the computation time of performance evaluation under stochastic processing times, TSSM adopts a regression-based meta-model to determine some promising individuals, which are further estimated using a simulator, i.e. Algorithm I. Since different methods are used to estimate the individual fitness in TSSB-HEDA, it is not suitable to select good individuals from the population only depending on their fitness values. Therefore, a modified linear rank selection is employed to choose part

of individuals with good performance under stochastic processing times.

The modified linear rank selection involves two steps: (1) the individuals are ranked in ascending order of the estimated fitness under stochastic processing times, which is the reciprocal of the objective function. Based on the regression-based meta-model, the worst $(1-\alpha)\times 100\%\left(\alpha \in [0,1]\right)$ individuals are first identified and sorted. Then, using Algorithm I, the rest $\alpha \times 100\%\left(\alpha \in [0,1]\right)$ individuals are evaluated and sorted; (2) the individuals are selected with the probability:

$$P(x_k^i) = r_k^i \Big/ \sum_{x_k^i \in P_k} r_k^i \tag{10}$$

where P_k denotes the k^{th} population of TSSB-HEDA, x_k^i and r_k^i represent the i^{th} individual and its rank in population P_k respectively. To produce the elite set for the probabilistic model construction, such selection process continues until $\beta \times 100\%\,(\beta \in [0,1])$ individuals of current population have been chosen.

3.3.3 Probabilistic model

Instead of using genetic operators, the probabilistic model is applied in the conventional EDA to guide the exploration of the search space. It therefore affects the EDA performance substantially. To address the PFSP with total flow time minimisation, Jarboui et al. (2009) developed an effective probabilistic model with consideration of both job order in the sequence and similar job blocks of selected elite parents. In their proposed model, the probability for selecting job j at position k, i.e. π_{jk}, is computed as follows:

$$\pi_{jk} = \frac{\eta_{jk} \times \mu_{j[k-1]}}{\sum_{l \in \Omega_k}\left(\eta_{lk} \times \mu_{l[k-1]}\right)} \tag{11}$$

where η_{jk} represents the number of times that job j appears before or at position k in the selected individuals augmented by a given constant δ_1; $\mu_{j[k-1]}$ denotes the number of times that job j appears immediately after the job at position k−1 in the selected individuals augmented by a given constant δ_2; Ω_k indicates the set of jobs that are not scheduled until position k.

Such a probabilistic modelling method has its own disadvantage when applied to solve the PFSP. According to the definition of $\mu_{j[k-1]}$, it equals zero when $k=1$, since no job can be positioned before job j if it is selected as the first job to be processed. This may lead the probability of positioning any job at position 1 to be zero. Therefore, instead of being determined by the genetic information of the elite set, the first job in the sequence is randomly

chosen. To overcome such drawback, we apply a new probabilistic model to determine π_{jk} as

$$\pi_{jk} = \begin{cases} \dfrac{\eta_{jk}}{\sum_{l\in\Omega_k}\eta_{jk}}, & k=1 \\ \dfrac{\eta_{jk}\times\mu_{j[k-1]}}{\sum_{l\in\Omega_k}\left(\eta_{lk}\times\mu_{l[k-1]}\right)}, & k=2,3...,n \end{cases} \tag{12}$$

Based on formula (12), the job assigned to a specific position k in the sequence can be determined by the probability of π_{jk}. For each generation, $R_{EDA}\times 100\%$ ($R_{EDA}\in[0,1]$) offspring are generated by the probabilistic model.

3.3.4 Crossover and mutation

To prevent EDA from being trapped into premature convergence, genetic operators, namely crossover and mutation, are adopted to generate part of population. As the primary genetic operator, crossover guides the exploration of new promising regions in the search space. It usually produces the offspring by interchanging parts of their parents (genes) (Huang et al., 2015). TSSB-HEDA applies the order crossover to generate the offspring, which is explained as follow: two parents are first divided into three parts by selecting two random cut points. Then the middle parts of two children are then directly inherited from their parents, and the remaining genes of child 1 are filled following the job order appeared in parent 2. Similarly, the remaining genes of child 2 are determined based on parent 1. An example of order crossover is illustrated in Fig. 3.

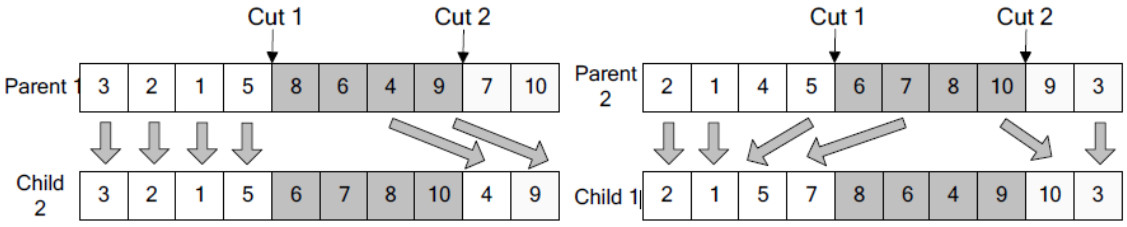


Fig. 3. An example of order crossover

To preserve the population diversity, mutation changes one or more genes in a chromosome from its initial state (Choi and Wang, 2012). TSSB-HEDA adopts the pairwise interchange mutation to swap two randomly selected jobs of the individual, as illustrated in Fig. 4.

For each generation of TSSB-HEDA, the order crossover is performed to generate $(1-R_{EDA})\times 100\%$ ($R_{EDA}\in[0,1]$) of new individuals, and each of them is mutated with rate p_m.

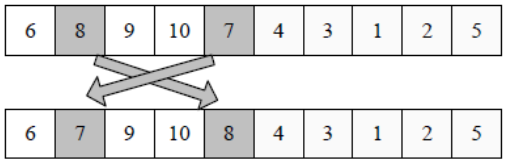


Fig. 4. An example of pairwise interchange mutation

3.3.5 Self-adaptive learning mechanism

To improve the population diversity of EDA, both the probabilistic model of EDA and genetic operators of GA are adopted to produce the offspring. The proposed TSSB-HEDA applies a self-adaptive learning mechanism (SALM) to determine the EDA participation ratio, which indicates how many individuals of a population are produced by the probabilistic model. SALM is based on the weight training process of neural network, which has been adopted by Agarwal et al. (2006) to address the traditional flowshop scheduling problems. In their research work, a set of weights were used to construct solutions by perturbing the data of original scheduling problem. To iteratively improve the solutions, they employed a learning approach to dynamically change the weights by reinforcement and backtracking. The set of weights were reinforced if an iteration resulted in improvement, and were backtracked to the best set of weights so far if there was no improvement over a certain number of iterations.

Inspired by the work of Agarwal et al. (2006), SALM uses both reinforcement and backtracking to adjust the EDA participation ratio R_{EDA}. The idea of reinforcement originates from the weight training process of neural network. If the best individual found so far improves, we record R_{EDA} at current population as the best one and then reinforce its change from previous generation using a reinforcement factor (RF). Thus, if R_{EDA} increases from its previous value, more offspring are sampled from the probabilistic model at the next generation. Otherwise, fewer offspring are produced using the probabilistic model. If no improvement happens at an iteration, a perturbation strategy is used to update R_{EDA}, which allow either slightly increase or decrease the number of offspring generated by the probabilistic model. In addition to reinforcement, SALM allows backtracking to a previous R_{EDA} according to the predefined tolerate iterations with no improvement (TINI). If the best individual does not improve for consecutive TINI generations, EDA may produce offspring with similar structure and tend to get trapped into premature convergence. Therefore, backtracking is performed by changing the current R_{EDA} to the best one so far. The procedure of SALM is detailed below.

Algorithm II: SALM

Step 1: Find the best individual in the current generation *i*. If it provides the best makespan so far, record the makespan as the best makespan M_b and the current R_{EDA}^i as the best EDA participation ratio BR_{EDA}.

Step 2: If the TINI counter $k \le TINI$ and improvement of M_b occurs at current generation *i*, set $k = 1$ and use one of the following strategies to reinforce the EDA participation ratio at generation *i+1*, i.e. R_{EDA}^{i+1}. In these two strategies, *RF* indicates the reinforcement factor.

(a) $R_{EDA}^{i+1} = min\left\{R_{EDA}^i + RF \times \left(R_{EDA}^i - R_{EDA}^{i-1}\right), 1\right\},\ if\ R_{EDA}^i - R_{EDA}^{i-1} \ge 0$

(b) $R_{EDA}^{i+1} = max\left\{R_{EDA}^i + RF \times \left(R_{EDA}^i - R_{EDA}^{i-1}\right), 0\right\},\ if\ R_{EDA}^i - R_{EDA}^{i-1} < 0$

Step 3: If $k \le TINI$ and no improvement of M_b occurs at current generation *i*, set $k = k + 1$ and select one of the following strategies to update the EDA participation ratio at generation *i+1*, i.e. R_{EDA}^{i+1}. In these two strategies, r represents a uniformly random number from [0, 1] and γ is the learning rate.

(a) $R_{EDA}^{i+1} = min\left\{R_{EDA}^i + r \times \gamma, 1\right\},\ if\ r \ge 0.5$

(b) $R_{EDA}^{i+1} = max\left\{R_{EDA}^i - r \times \gamma, 0\right\},\ if\ r < 0.5$

Step 4: If $k > TINI$ and no improvement of M_b occurs during the past TINI generations, set $k = 1$ and $R_{EDA}^{i+1} = BR_{EDA}$.

After the offspring are produced by the probabilistic model and genetic operators, SALM is applied to adjust the EDA participation ratio. We initialise the TINI counter $k = 1$ and the EDA participation ratio $R_{EDA} = 50\%$ when SALM is implemented for the first time.

3.3.6 Stopping criterion

A variety of stopping criteria have been considered in the literature of EDA, such as the number of generations (Wang et al., 2013), the number of consecutive generations with no improvement (Zhang and Li, 2011), the number of examined solutions (Chen and Chen, 2013), and bound of computation time (Jarboui et al., 2009), etc. Similar to that of Wang et al. (2013), the maximum number of generations is adopted as the stopping criterion of TSSB-HEDA.

3.4 Complete procedure of TSSB-HEDA

According to the detailed description above, the complete procedure of TSSB-HEDA is

described as follows:

Notation:

N_{sim}: the number of simulation replications

P_s: the size of population

α: the ratio of population related to identifying promising individuals for further performance evaluation

β: the ratio of population related to establishing the probabilistic model of EDA

TINI: tolerate iterations with no improvement of SALM

RF: reinforcement factor of SALM

γ: the learning rate of SALM

k: the TINI counter of SALM

R_{EDA}: the EDA participation ratio

S_c: the best individual of current population

S_b: the best individual found so far

CurGen: the current generation index

MaxGen: the maximum number of generations

Algorithm III: TSSB-HEDA

Step 1: Set the algorithm parameters N_{sim}, P_s, P_m, α, β, γ, *TINI*, *RF*, *MaxGen*. Let $CurGen = 1, k = 1$, and $R_{EDA} = 50\%$;

Step 2: Randomly initialise a population of P_s individuals;

Step 3: Estimate the initial population under stochastic processing times using the TSSM and choose the best one as S_b;

Step 4: Identify $\beta \times P_s$ individuals to establish the elite set Π_e;

Step 5: Construct the probabilistic model of EDA using Π_e according to formula (12);

Step 6: Apply the constructed probabilistic model to sample and generate $R_{EDA} \times P_s$ offspring;

Step 7: Apply the order crossover and the pairwise interchange mutation to generate $(1 - R_{EDA}) \times P_s$ offspring;

Step 8: Employ the TSSM to determine the best individual of the current population under stochastic processing times, i.e. S_c, and then update $S_b = S_c$ if the expected makespan of S_c is less than that of S_b;

Step 9: Perform SALM to adjust R_{EDA}.

Step 10: If $CurGen \geq MaxGen$, return S_b; otherwise, let $CurGen = CurGen + 1$ and return to step 4.

4. Computational experiments

The proposed TSSB-HEDA is coded in the Netbeans™ development environment 7.4 and performed on a PC with Intel® Core™ i5 2.7GHz processor and 6GB memory. To validate the effectiveness of TSSB-HEDA, we designed and conducted three experiments, which are detailed in the following sections.

4.1 Design of experiments

The first experiment constructs the regression-based meta-model to predict the DSP of planned schedules under stochastic processing times. Instead of conducting a full factorial experiment, the second experiment applies the Taguchi experimental design to identify the near optimum values of key parameters in TSSB-HEDA. After these two experiments, TSSB-HEDA is well prepared to address the PFSP under stochastic processing times, and its performance with makespan criterion is subsequently analysed in the third experiment.

To model the processing time uncertainty in a permutation flowshop, the actual job processing time P is generated from a normal distribution with its expected value of $E(P)$ and the standard deviation of σ. To guarantee job processing time to be non-negative in these three experiments, we adopt a left-truncated normal distribution at zero, i.e. $P \sim N\left(E(P), \sigma^2\right)$ where $P \in (0, +\infty)$. For the second and the third experiments, TSSB-HEDA and the compared algorithms are terminated when 10×n generations are reached, where n is the number of jobs.

4.2 Development of regression-based meta-model for DSP Estimation

To obtain the regression-based meta-model for DSP estimation, it is necessary to first generate the simulation results that describe the effect of simulation inputs, including job size, machine size, LPTV, and SR, on the performance degradation of planned schedules under stochastic processing times. The levels of these four simulation inputs used in the experiment are presented in Table 1.

Table 1 Simulation inputs and their levels

Factors	Levels
Job size	20, 70, 120, 170, 220
Machine size	5, 10, 15, 20, 25
LPTV	0.10, 0.20, 0.30, 0.40, 0.50
SR	0.25, 0.30, 0.35, 0.40, 0.45

For each possible combination of simulation inputs, an experimental PFSP with processing time uncertainty is generated. To solve this problem, a job sequence is first randomly generated and converted into a feasible schedule with a specific SR. The makespans of such schedules under stochastic processing times and deterministic processing times, i.e. M_{AS} and M_{PS}, are subsequently obtained. Lastly, the degradation of schedule performance of a planned schedule, i.e. DSP, is computed as

$$DSP = M_{AS}/M_{PS} - 1 \tag{13}$$

Such process continues until each combination of the simulation inputs in Table 1 has been examined. Therefore, a total of 625 (5×5×5×5 = 625) simulation experiments are conducted to generate the simulation results.

Table 2 Results of analysis of variance for the meta-model

Model	Sum of squares	Mean squares	F-ratio	P-value	Correlation of coefficient R^2
Degradation of Schedule Performance (DSP)	39.422	3.942	1271.157	<0.001	0.953

Based on the simulation results, multiple linear regression analysis has been applied to establish the meta-model using formula (9). Accordingly, the proposed regression-based meta-model is obtained as follows:

$$\begin{aligned} DSP = &-0.04497 + 0.00019x_1 + 0.00593x_2 + 2.10028x_3 - 0.16319x_4 - 0.00002x_1x_2 \\ &- 0.00266x_1x_3 - 0.00033x_1x_4 + 0.02441x_2x_3 - 0.00358x_2x_4 - 1.71680x_3x_4 \end{aligned} \tag{14}$$

To evaluate the effectiveness of the developed meta-model, Table 2 gives the results of analysis of variance with 5% significance level for the meta-model. According to Table 2, the following observations are made: (1) The regression-based meta-model is statistically significant because of the small p-value (less than 0.05); (2) The correlation of coefficient R^2 (larger than 0.95) implies that the meta-model can explain over 95% of the variance in the estimated performance degradation. These observations indicate that the

regression-based meta-model is statistically adequate to fit the simulation results under stochastic processing times.

To further test the validity of the developed meta-model, we consider two typical PFSPs, i.e. 50×10 and 100×20, in which the expected processing times are generated from the uniform distribution with range [1, 20]. For these two PFSPs, different values of LPTV and SR are adopted as the inputs of the simulation model. LPTV is chosen to be 0.15, 0.25, 0.35, and 0.45. SR can be 0.26, 0.34, and 0.42. Table 3 presents the DSPs obtained using the discrete-event simulator and the predicted DSPs using the meta-model for the chosen LPTVs and SRs. It is clear that the percentage deviation between the meta-model results and the simulation results is less than 5%, indicating that the developed meta-model provides a good prediction of DSP under stochastic processing times.

Table 3 Validation results of the meta-model

LPTV	SR	Test problem (No. of jobs x no. of machines)					
		50×10			100×20		
		Simulation results	Meta-model results	Error deviation	Simulation results	Meta-model results	Error deviation
0.15	0.26	0.225	0.223	0.009	0.268	0.264	0.015
	0.34	0.182	0.185	0.016	0.224	0.222	0.009
	0.42	0.145	0.147	0.014	0.177	0.180	0.017
0.25	0.26	0.397	0.395	0.005	0.457	0.452	0.011
	0.34	0.342	0.348	0.018	0.387	0.396	0.023
	0.42	0.291	0.296	0.017	0.346	0.341	0.014
0.35	0.26	0.568	0.576	0.014	0.626	0.640	0.022
	0.34	0.521	0.510	0.021	0.560	0.570	0.018
	0.42	0.438	0.445	0.016	0.489	0.501	0.025
0.45	0.26	0.741	0.752	0.015	0.837	0.827	0.012
	0.34	0.657	0.673	0.024	0.730	0.744	0.019
	0.42	0.606	0.594	0.020	0.641	0.661	0.031

4.3 Parameter tuning of TSSB-HEDA

The performance of TSSB-HEDA depends on the values of some important parameters, such as P_s (the size of population), α (the ratio of population related to identifying promising

individuals for further performance evaluation), β (the ratio of population related to establishing the probabilistic model of EDA), p_m (mutation rate), γ (the learning rate of SALM), *TINI* (tolerate iterations with no improvement of SALM), and *RF* (reinforcement factor of SALM). To determine the near optimum values of these parameters, Taguchi experiments (Taguchi, 1986) are conducted on a moderate-size PFSP with 50 jobs and 10 machines. In this PFSP, LPTVs of machines and expected job processing times are uniformly generated in the ranges [0.1, 0.5] and [1, 20], respectively. Compared with a full factorial experiment, the Taguchi method is capable of reducing the number of experiments substantially (Naderi et al., 2010).

Seven key parameters of TSSB-HEDA and their different factor levels considered in Taguchi experiments are presented in Table 4. Accordingly, an orthogonal array L_{18} ($6^1 \times 3^6$) shown in Table 5 is established by MINITAB 16 for parameter tuning. Rather than performing $6^1 \times 3^6$= 4,374 experiments in a full factorial design, we only conduct a total of 18 Taguchi experiments to identify the near optimum values of these seven parameters. For each of the Taguchi experiment, TSSB-HEDA with a specific level combination of factors is first run separately 20 times for the same experimental PFSP under stochastic processing times. The average of simulated makespans, i.e. the value of response variable (RV) in a Taguchi experiment, is then determined. Based on RV values, the average response at each factor level is obtained and presented in Fig. 5.

Table 4 Factors and their levels of TSSB-HEDA

Factor	Level 1	Level 2	Level 3	Level 4	Level 5	Level 6
α	0.20	0.25	0.30	0.35	0.40	0.45
P_s	100	300	500	-	-	-
β	0.10	0.15	0.20	-	-	-
p_m	0.10	0.20	0.30	-	-	-
γ	0.05	0.10	0.15	-	-	-
TINI	10	30	50	-	-	-
RF	1.00	1.10	1.20	-	-	-

Table 5 Orthogonal array L_{18} ($6^1 \times 3^6$) of TSSB-HEDA

Trial	Factor						
	α	P_s	β	p_m	γ	*TINI*	*RF*
1	1	1	1	1	1	1	1
2	1	2	2	2	2	2	2
3	1	3	3	3	3	3	3
4	2	1	1	2	2	3	3
5	2	2	2	3	3	1	1
6	2	3	3	1	1	2	2
7	3	1	2	1	3	2	3
8	3	2	3	2	1	3	1
9	3	3	1	3	2	1	2
10	4	1	3	3	2	2	1
11	4	2	1	1	3	3	2
12	4	3	2	2	1	1	3
13	5	1	2	3	1	3	2
14	5	2	3	1	2	1	3
15	5	3	1	2	3	2	1
16	6	1	3	2	3	1	2
17	6	2	1	3	1	2	3
18	6	3	2	1	2	3	1

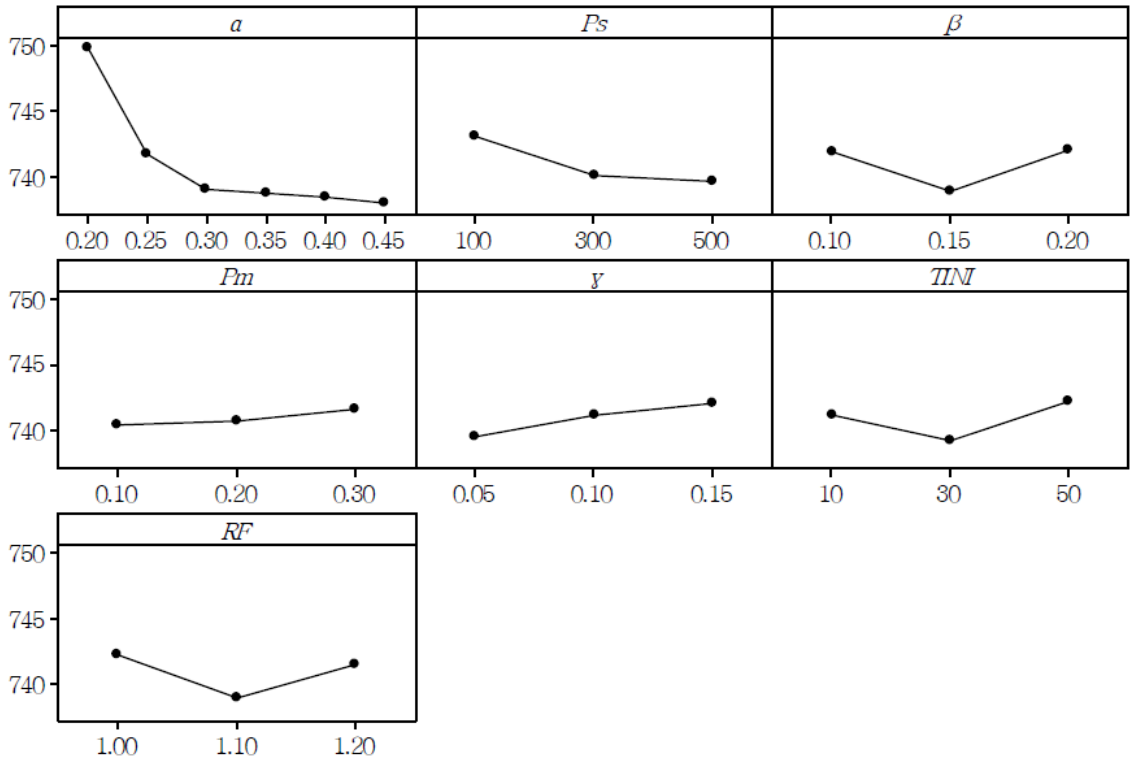


Fig. 5. Factor level trend of TSSB-HEDA

To further analyse the significance of individual factor in TSSB-HEDA, Table 6 ranks the factors according to their Delta statistics, which is the difference between the highest and lowest average of RV. The rank of a factor shows its relative importance to the performance of TSSB-HEDA.

Table 6 Average response value and rank of each factor

Level	Factor						
	α	P_s	β	p_m	γ	*TINI*	*RF*
1	749.8	743.1	741.9	740.5	739.5	741.3	742.3
2	741.8	740.1	738.9	740.7	741.2	739.3	739.0
3	739.0	739.6	742.1	741.6	742.1	742.3	741.5
4	738.7	-	-	-	-	-	-
5	738.4	-	-	-	-	-	-
6	737.9	-	-	-	-	-	-
Delta (max-min)	11.9	3.5	3.2	1.1	2.6	3.0	3.3
Rank	1	2	4	7	6	5	3

From the ranks in Table 6, it is clear that α has the most significant effect on the performance of TSSB-HEDA. Although a large α may result in a more accurate performance estimation of candidate solutions under stochastic processing time, the large computation cost of modelling processing time uncertainty has to be considered. As shown in Fig. 5, the performance of TSSB-HEDA improves slightly once α is larger than 0.3. The parameter α of 0.3 is therefore adopted in this study to prevent large computation time. The value of P_s is likewise set to 300 when considering the amount of computation cost. For the parameters associated with SALM, the moderate values of *TINI* and *RF* can help avoid large fluctuation of the EDA partition ratio, and hence may provide a better balance between population diversity and convergence performance. Furthermore, the values of β, p_m, and γ can be easily determined by Fig. 5. From the above analysis, all the key parameters of TSSB-HEDA are accordingly set as: $\alpha = 0.30$, $P_s = 300$, $\beta = 0.15$, $p_m = 0.10$, $\gamma = 0.05$, $TINI = 30$, and $RF = 1.10$.

4.4 Performance evaluation of TSSB-HEDA

Different from traditional EDA for permutation flowshop scheduling in a static environment, the proposed TSSB-HEDA is characterised by using TSSM and genetic operators to construct solutions considering processing time uncertainty. Therefore, to evaluate the effectiveness of TSSB-HEDA, it is compared with three scheduling algorithms, namely simulation-based

hybrid EDA (SB-HEDA), TSSB-EDA with no genetic operators (TSSB-EDA), and TSSB-GA. The brief explanation of these compared scheduling algorithms is shown below:

- SB-HEDA: This algorithm is the same as TSSB-HEDA, except for applying a time-consuming simulator to estimate all candidate solutions rather than some selected promising ones. Compared with TSSB-HEDA, SB-HEDA provides a more accurate performance estimation of candidate solutions, so that it tends to outperform TSSB-HEDA in terms of solution quality. However, with respect to computational efficiency, SB-HEDA is more costly and time-consuming since it takes $O(P_s)$ time for performance evaluation, while TSSB-HEDA takes about $O(\alpha P_s)$ $(\alpha \in [0,1])$ time for such task.
- TSSB-EDA: Rather than hybridising the probabilistic model of EDA with genetic operators of GA for population generation in TSSB-HEDA, this algorithm only applies the probabilistic model to produce the offspring.
- TSSB-GA: Similar to TSSB-HEDA, the initial population is randomly generated and two genetic operators, namely order crossover and swap mutation, are adopted to produce the offspring. The crossover and mutation rates are empirically fixed at 0.8 and 0.1, respectively.

To have a fair comparison, the key parameters of SB-HEDA, TSSB-EDA, and TSSB-GA, such as P_s, $MaxGen$, and N_{sim}, have the same values as those used in TSSB-HEDA. Moreover, all these four algorithms are stopped after 10×n generations. This experiment is conducted on the well-known Taillard's benchmark problems with m = 5, 10, and 20 and n = 20, 50, 100, and 200 (Taillard, 1993). Ten instances are considered for each PFSP, in which the LPTVs of machines are generated from the uniform distribution with range [0.1, 0.5]. To compare schedule performance of TSSB-HEDA, SB-HEDA, TSSB-EDA, and TSSB-GA, we perform the four algorithms 10 times for each instance and measure the solution quality of each instance group by the minimum relative percentage deviation (denoted as Δ_{min}), the average relative percentage deviation (denoted as Δ_{avg}), and the maximum relative percentage deviation (denoted as Δ_{max}). These three performance measures are computed as follows:

$$\Delta_{min} = \sum_{r=1}^{10} \left(\frac{\left(BS_{alg}^{r} - S_{best}^{r} \right) \times 100}{S_{best}^{r}} \right) \Big/ 10 \tag{15}$$

$$\Delta_{avg} = \sum_{r=1}^{10} \left(\frac{\left(AS_{alg}^{r} - S_{best}^{r} \right) \times 100}{S_{best}^{r}} \right) \Big/ 10 \tag{16}$$

$$\Delta_{max} = \sum_{r=1}^{10} \left(\frac{\left(WS_{alg}^{r} - S_{best}^{r} \right) \times 100}{S_{best}^{r}} \right) \Big/ 10 \tag{17}$$

where BS_{alg}^{r}, AS_{alg}^{r}, and WS_{alg}^{r} respectively denote makespans of the best, average, and worst solutions obtained using a specific algorithm for instance r; S_{best}^{r} represents the minimum makespan among the solutions generated using all compared algorithms for instance r.

Table 7 shows the experiment results of SB-HEDA and TSSB-HEDA under stochastic processing times. According to the experiment results, the following are observed: (1) SB-HEDA performs better than TSSB-HEDA in terms of Δ_{min}, Δ_{avg}, and Δ_{max}. The good performance of SB-HEDA lies in evaluating all the offspring individuals by the time-consuming simulator; (2) since the difference in the performance of SB-HEDA and TSSB-HEDA is not significant, TSSM is found to be effective to estimate the schedule performance considering processing time uncertainty.

Table 7 Comparison results of SB-HEDA and TSSB-HEDA under normal processing times

No. of jobs x no. of machines	SB-HEDA			TSSB-HEDA		
	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}
20×5	0	0	0	0.29	0.51	0.70
20×10	0	0.05	0.09	0.36	0.58	0.81
20×20	0	0.02	0.06	0.25	0.50	0.82
50×5	0.08	0.15	0.21	0.41	0.61	0.85
50×10	0.09	0.19	0.28	0.54	0.67	0.91
50×20	0.19	0.24	0.37	0.57	0.82	1.03
100×5	0.08	0.16	0.25	0.39	0.61	0.84
100×10	0.13	0.27	0.51	0.55	0.73	0.97
100×20	0.16	0.38	0.44	0.58	0.85	1.09
200×10	0.18	0.32	0.56	0.48	0.71	0.97
200×20	0.16	0.36	0.63	0.59	0.86	1.15
Average	0.10	0.19	0.31	0.46	0.68	0.92

Different from the conventional EDA, TSSB-HEDA hybridises EDA with GA to address the PFSP under stochastic processing times. Therefore, to further validate the effectiveness of TSSB-HEDA, it is compared with TSSB-EDA and TSSB-GA. The experiment results in Table 8 indicate that TSSB-HEDA performs significantly better than either TSSB-EDA or TSSB-GA. Furthermore, Δ_{max} of TSSB-HEDA is less than Δ_{min} of TSSB-EDA and TSSB-GA for all the test problems, which shows the superiority of the proposed TSSB-HEDA.

Table 8 Computation results of TSSB-HEDA, TSSB-EDA, and TSSB-GA under normal processing times

No. of jobs x no. of machines	TSSB-HEDA			TSSB-EDA			TSSB-GA		
	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}	Δ_{min}	Δ_{avg}	Δ_{max}
20×5	0	0.05	0.23	2.54	2.81	3.27	1.65	1.86	2.17
20×10	0	0.11	0.17	2.66	2.89	3.13	1.75	1.91	2.28
20×20	0.12	0.22	0.33	2.71	3.04	3.39	1.83	2.05	2.41
50×5	0.08	0.14	0.26	2.59	2.88	3.23	1.69	1.97	2.36
50×10	0.10	0.31	0.38	2.64	3.08	3.37	1.77	2.08	2.39
50×20	0.21	0.36	0.44	2.84	3.19	3.44	1.89	2.15	2.51
100×5	0.09	0.25	0.38	2.63	2.97	3.35	1.77	1.99	2.34
100×10	0.16	0.40	0.63	2.68	3.13	3.59	1.81	2.19	2.58
100×20	0.12	0.35	0.70	2.60	3.09	3.67	1.74	2.14	2.47
200×10	0.15	0.38	0.51	2.65	3.08	3.44	1.80	2.21	2.59
200×20	0.20	0.48	0.77	2.72	3.28	3.75	1.87	2.34	2.71
Average	0.11	0.28	0.44	2.66	3.04	3.42	1.78	2.08	2.44

To further investigate the performance of the proposed TSSB-HEDA, the computation time of TSSB-HEDA, SB-HEDA, TSSB-EDA, and TSSB-GA are compared. Table 9 summarises the average CPU times in seconds of these four algorithms. From this table, the following conclusions can be made:

(1) SB-HEDA is computationally more expensive than TSSB-HEDA, although it provides slightly better results, as shown in Table 7. TSSB-HEDA is computationally more efficient than SB-HEDA because it only applies the time-consuming simulator to evaluate a small portion of a population, i.e. $\alpha P_s (\alpha \in [0,1])$, rather than the whole population.

(2) As a hybridisation of EDA and GA, the average CPU time of SB-HEDA is between those of TSSB-EDA and TSSB-GA.

Table 9 Average CPU times of SB-HEDA, TSSB-HEDA, TSSB-EDA, and TSSB-GA under normal processing times

No. of jobs x no. of machines	SB-HEDA	TSSB-HEDA	TSSB-EDA	TSSB-GA
20×5	52.8	16.7	29.4	6.3
20×10	68.2	23.3	36.2	8.2
20×20	89.5	28.7	42.2	10.6
50×5	581.6	158.1	280.8	16.8
50×10	621.7	177.5	302.4	35.9
50×20	674.4	214.2	324.7	46.5
100×5	2487.2	781.9	1255.5	52.9
100×10	2689.5	882.4	1385.5	121.2
100×20	2910.9	925.6	1426.3	209.2
200×10	10213.2	3364.1	5410.3	741.0
200×20	11463.5	3881.8	6334.2	1137.1
Average	2895.7	950.4	1529.8	216.9

5. Conclusion

In this paper, an effective two-stage simulation-based hybrid EDA (TSSB-HEDA) is presented to address the permutation flowshop scheduling problems with processing time uncertainty. To reduce the computation cost of evaluating the offspring, TSSB-HEDA employs a novel two-stage simulation model (TSSM) for performance estimation. In the first stage of TSSM, a regression-based meta-model is adopted to provide a quick performance evaluation on offspring individuals, and only a number of promising ones are subsequently estimated in the second stage using a relatively time-consuming simulator. Furthermore, to enhance the population diversity of EDA, TSSB-HEDA applies both the probabilistic model of EDA and genetic operators of GA to produce the offspring. Inspired by the idea of neural network leaning, a self-adaptive learning mechanism (SALM) is established to determine the ratio of offspring generated by the probabilistic model.

To validate the performance of TSSB-HEDA, it has been compared with three scheduling algorithms, namely simulation-based hybrid EDA (SB-HEDA), TSSB-HEDA with no genetic operators (TSSB-EDA), and TSSB-GA. The experiment results on the well-known Taillard's benchmark problems show that TSSB-HEDA can maintain a good balance between schedule performance and computation time. Such a quality-time balance is resulted from the efficiency of TSSM in estimating the schedule performance with processing time uncertainty, and SALM in dynamically adjusting the ratio of individuals generated by EDA to avoid early search

stagnation. Since real-world manufacturing suffers a variety of uncertainties, future research may extend the proposed TSSB-HEDA to deal with other unexpected events in a permutation flowshop, such as rush order and machine breakdown. Furthermore, another research direction can focus on exploring possible hybridisation of EDA with other effective meta-heuristics to enhance the population diversity.

References

Agarwal, A., Colak, S., Eryarsoy, E., 2006. Improvement heuristic for the flow-shop scheduling problem: an adaptive-learning approach. European Journal of Operational Research 169(3), 801-815.

Ahmadizar, F., Ghazanfari, M., Ghomi, S. M. T. F., 2010. Group shops scheduling with makespan criterion subject to random release dates and processing times. Computers & Operations Research 37(1), 152-162.

Ceberio, J., Irurozki, E., Mendiburu, A., and Lozano, J. A., 2014. A Distance-based Ranking Model Estimation of Distribution Algorithm for the Flowshop Scheduling Problem. IEEE Transactions on Evolutionary Computation 18(2), 286-300.

Chang, P. C., Hsieh, J. C., Chen, S. H., Lin, J. L., and Huang, W. H., 2009. Artificial chromosomes embedded in genetic algorithm for a chip resistor scheduling problem in minimizing the makespan. Expert Systems with Applications 36(3), 7135-7141.

Chen, S. H., Chen, M. C., Chang, P. C., Zhang, Q., Chen, Y. M., 2010. Guidelines for developing effective estimation of distribution algorithms in solving single machine scheduling problems. Expert Systems with Applications 37(9), 6441-6451.

Chen, Y. M., Chen, M. C., Chang, P. C., Chen, S. H., 2012. Extended artificial chromosomes genetic algorithm for permutation flowshop scheduling problems. Computers & Industrial Engineering 62(2), 536-545.

Chen, S. H., Chen, M. C., 2013. Addressing the advantages of using ensemble probabilistic models in estimation of distribution algorithms for scheduling problems. International Journal of Production Economics 141(1), 24-33.

Choi, S. H., Wang, K., 2012. Flexible flow shop scheduling with stochastic processing times: A decomposition-based approach. Computers & Industrial Engineering 63(2), 362-373.

Chung, C. S., Flynn, J., Kirca, O., 2002. A branch and bound algorithm to minimize the total flow time for m-machine permutation flowshop problems. International Journal of Production Economics 79(3), 185-196.

Dugardin, F., Yalaoui, F., Amodeo, L., 2010. New multi-objective method to solve reentrant hybrid flowshop scheduling problem. European Journal of Operational Research 203(1), 22-31.

Gao, K. Z., Pan, Q. K., Li, J. Q., 2011. Discrete harmony search algorithm for the no-wait flow shop scheduling problem with total flow time criterion. The International Journal of Advanced Manufacturing Technology 56(5-8), 683-692.

Garey, M. R., Johnson, D. S., Sethi, R., 1976. The complexity of flowshop and jobshop scheduling. Mathematics of operations research 1(2), 117-129.

Gholami, M., Zandieh, M., Alem-Tabriz, A., 2009. Scheduling hybrid flowshop with sequence-dependent setup times and machines with random breakdowns. The International Journal of Advanced Manufacturing Technology 42(1), 189-201.

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

Hauschild, M., Pelikan, M., 2011. An introduction and survey of estimation of distribution algorithms. Swarm and Evolutionary Computation 1(3), 111-128.

Huang, Y., Wang, K., Zhang, T., Pang, C., 2015. Green supply chain coordination with greenhouse gases emissions management: a game-theoretic approach. Journal of Cleaner Production, doi:10.1016/j.jclepro.2015.05.137.

Hurrion, R. D., Birgil, S., 1999. A comparison of factorial and random experimental design methdos for the development of regression and neural network simulation metamodels. Journal of the operational research society, 1018-1033.

Horng, S. C., Lin, S. S., Yang, F. Y., 2012. Evolutionary algorithm for stochastic job shop scheduling with random processing time. Expert Systems with Applications 39(3), 3603-3610.

Jarboui, B., Eddaly, M., Siarry, P., 2009. An estimation of distribution algorithm for minimizing the total flowtime in permutation flowshop scheduling problems. Computers & Operations Research 36(9), 2638-2646.

Johnson, S. M., 1954. Optimal two- and three- stage production schedules with setup times included. Naval research logistics quarterly 1(1), 61-68.

Juan, A. A., Barrios, B. B., Vallada, E., Riera, D., Jorba, J., 2014. A simheuristic algorithm for solving the permutation flow shop problem with stochastic processing times. Simulation Modelling Practice and Theory 46, 101-117.

Lawrence, S. R., Sewell, E. C., 1997. Heuristic, optimal, static, and dynamic schedules when processing times are uncertain. Journal of Operations Management 15 (1), 71-82.

Liu, H., Gao, L., Pan, Q., 2011. A hybrid particle swarm optimization with estimation of distribution algorithm for solving permutation flowshop scheduling problem. Expert Systems

with Applications 38(4), 4348-4360.

Li, J. Q., Pan, Q. K., 2014. Solving the large-scale hybrid flow shop scheduling problem with limited buffers by a hybrid artificial bee colony algorithm. Information Sciences, doi:10.1016/j.ins.2014.10.009

Mühlenbein, H., Paass, G., 1996. From recombination of genes to the estimation of distributions I: Binary parameters. Lecture Notes in Computer Science 1141, 178-187.

Naderi, B., Ghomi, S. M. T., Aminnayeri, M., 2010. A high performing metaheuristic for job shop scheduling with sequence-dependent setup times. Applied Soft Computing 10(3), 703-710.

Ouelhadj, D., Petrovic, S., 2009. A survey of dynamic scheduling in manufacturing systems. Journal of Scheduling 12(4), 417-431.

Pan, Q. K., Ruiz, R., 2012. An estimation of distribution algorithm for lot-streaming flowshop problems with setup times. Omega 40(2), 166-180.

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

Saad, S. M., 2003. The reconfiguration issues in manufacturing systems. Journal of Materials Processing Technology 138(1), 277-283.

Safari, E., Sadjadi, S. J., 2011. A hybrid method for flowshops scheduling with condition-based maintenance constraint and machines breakdown. Expert Systems with Applications 38(3), 2020-2029.

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

Taguchi, G., 1986. Introduction to quality engineering. Asian Productivity Organization.

Tzeng, Y. R., Chen, C. L., Chen, C. L., 2012. A hybrid EDA with ACS for solving permutation flowshop scheduling. The International Journal of Advanced Manufacturing Technology 60(9-12), 1139-1147.

Vallada, E., Ruiz, R., 2009. Cooperative metaheuristics for the permutation flowshop scheduling problem. European Journal of Operational Research 193 (2), 365–376.

Vinod, V., Sridharan, R., 2009. Simulation-based metamodels for scheduling a dynamic job shop with sequence-dependent setup times.International Journal of Production Research 47(6), 1425-1447.

Vinod, V., Sridharan, R., 2011. Simulation modeling and analysis of due-date assignment methods and scheduling decision rules in a dynamic job shop production system. International Journal of Production Economics 129(1), 127-146.

Wang, K., Choi, S. H., Qin, H., Huang, Y., 2013. A cluster-based scheduling model using SPT

and SA for dynamic hybrid flow shop problems. The International Journal of Advanced Manufacturing Technology 67(9-12), 2243-2258.

Wang, K., Choi, S. H., 2014. A holonic approach to flexible flow shop scheduling under stochastic processing times. Computers & Operations Research 43, 157-168.

Wang, L., Pan, Q. K., Suganthan, P. N., Wang, W. H., Wang, Y. M., 2010. A novel hybrid discrete differential evolution algorithm for blocking flow shop scheduling problems. Computers & Operations Research 37(3), 509-520.

Wang, S. Y., Wang, L., Liu, M., Xu, Y., 2013. An effective estimation of distribution algorithm for solving the distributed permutation flowshop scheduling problem. International Journal of Production Economics 145(1), 387-396.

Xiong, J., Xing, L. N., Chen, Y. W., 2013. Robust scheduling for multi-objective flexible job-shop problems with random machine breakdowns. International Journal of Production Economics 141(1), 112-126.

Xu, J., Yin, Y., Cheng, T. C. E., Wu, C. C., Gu, S., 2014. An improved memetic algorithm based on a dynamic neighbourhood for the permutation flowshop scheduling problem. International Journal of Production Research 52(4), 1188-1199.

Yu, B., Popplewell, K., 1994. Metamodels in manufacturing: a review. International Journal of Production Research 32(4), 787-796.

Zandieh, M., Gholami, M., 2009. An immune algorithm for scheduling a hybrid flowshop with sequence-dependent setup times and machines with random breakdowns. International Journal of Production Research 47(24), 6999 - 7027.

Zhang, Y., Li, X., 2011. Estimation of distribution algorithm for permutation flowshops with total flowtime minimization. Computers & Industrial Engineering 60(4), 706-718.

Zhang, R., Song, S., Wu, C., 2012. A two-stage hybrid particle swarm optimization algorithm for the stochastic job shop scheduling problem. Knowledge-Based Systems 27, 393-406.

Zobolas, G. I., Tarantilis, C. D., Ioannou, G., 2009. Minimizing makespan in permutation flow shop scheduling problems using a hybrid metaheuristic algorithm. Computers & Operations Research, 36(4), 1249-1267.