International Journal of Production Research, 2014
http://dx.doi.org/10.1080/00207543.2014.930535

An estimation of distribution algorithm for hybrid flow shop scheduling under stochastic processing times

K. Wang[a*], S.H. Choi[b] and H. Qin[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, Hong Kong;* [c]*School of Management, Huazhong University of Science and Technology, Wuhan, China*

(*Received 17 October 2013; accepted 28 May 2014*)

The estimation of distribution algorithm (EDA) has recently emerged as a promising alternative to traditional evolutionary algorithms for solving combinatorial optimisation problems. This paper presents a novel two-phase simulation-based EDA (TPSB-EDA) for minimising the makespan of a hybrid flow shop under stochastic processing times. To address the stochastic scheduling problem efficiently, the proposed TPSB-EDA incorporates a two-phase simulation model to estimate the performance of candidate solutions. In this model, an optimal back propagation network is firstly applied to identify a set of roughly good solutions, and then the selected solutions are further evaluated by a discrete-event simulation algorithm. Moreover, an annealing selection mechanism (ASM) is adopted to preserve the population diversity of EDA. Different from the selection operators of common EDAs, the ASM uses Boltzmann probability in the annealing algorithm to select part of population to establish the probabilistic model. Computation results indicate that the TPSB-EDA provides good solutions in the aspects of solution quality and computational efficiency.

Keywords: annealing selection mechanism; back propagation network; estimation of distribution algorithm; hybrid flow shop; stochastic processing times; two-phase simulation model

1. Introduction

The hybrid flow shop (HFS) scheduling problem is one of the most extensively studied combinatorial optimisation problems. HFS systems are more commonly found in a wide variety of real-world industries, including paper bag factories, automobile industry, steel production, semiconductor manufacturing, etc. In such systems, machines are arranged into stages in series. At each stage, a number of functionally identical machines operate in parallel, and a job has to pass through all stages in the same order.

Research efforts on HFS scheduling generally consider a static environment, rendering the generated schedules infeasible in real-world manufacturing environments, which are subject to a wide range of stochastic uncertainties, such as machine breakdown and stochastic processing times. Therefore, dynamic HFS scheduling has indeed attracted much attention in recent years. The HFS scheduling problem has been proven NP-hard in nature (Garey and Johnson 1979; Gupta 1988). Consideration of uncertainties further aggravates its complexity.

Due to great difficulties in solving dynamic HFS scheduling, no algorithm can generate an optimal solution in polynomial time. Thus, meta-heuristics are commonly used to find approximate solutions. To minimise the effect of uncertainties in an HFS, the simulation-based meta-heuristics have recently attracted increasing attention. These approaches usually incorporate a discrete-event simulation model into a meta-heuristic (Wang and Choi 2014), such as simulated annealing (SA) heuristic (Allaoui and Artiba 2004), genetic algorithm (GA) (Gholami, Zandieh, and Alem-Tabriz 2009; Dugardin, Yalaoui, and Amodeo 2010), immune algorithm (Zandieh and Gholami 2009) and ant colony optimisation (ACO) (Ahmadizar, Ghazanfari, and Ghomi 2010).

The meta-heuristics are iterative processes that iteratively improve candidate solutions. In the simulation-based meta-heuristics, the discrete-event simulation model is adopted to evaluate candidate solutions over iterations until some convergence criterion is met. Such evaluation process is generally time-consuming since it requires a larger number of simulation replications to mimic the real-life manufacturing environment. As the number of simulation replications or candidate solutions grows, the computation time increases greatly (Ahmadizar, Ghazanfari, and Ghomi 2010; Dugardin,

*Corresponding author. Email: kai.wang@whu.edu.cn

Yalaoui, and Amodeo 2010). Therefore, the major drawback of simulation-based meta-heuristics is the large computation cost of evaluating candidate solutions.

To reduce the computation burden incurred from carrying out simulation replications, a potentially effective way is to conduct the time-consuming simulation only on a set of selected candidate solutions. Ahmadizar, Ghazanfari, and Ghomi (2010) presented a simulation-based ACO algorithm to minimise the makespan in a stochastic group shop. In the proposed approach, the performance of the constructed solution was first evaluated by the lower bound on the expected makespan without simulating, and only the solutions that might improve the current best one were further evaluated by a simulation model. More recently, Horng, Lin, and Yang (2012) incorporated an evolutionary strategy into ordinal optimisation to address a stochastic job-shop scheduling problem. To identify good schedules for further optimisation, they conducted the stochastic simulation with short simulation length to roughly estimate the fitness of candidate solutions. Although the simulation-based meta-heuristics have recently received increasing attention to address dynamic scheduling problems, few studies have been found to improve its computation efficiency.

This study aims to establish an effective simulation-based scheduling approach that provides high-quality solutions with relatively low computation cost in a dynamic manufacturing environment. Enlightened by the works of Ahmadizar, Ghazanfari, and Ghomi (2010) and Horng, Lin, and Yang (2012), we propose a novel two-phase simulation-based estimation of distribution algorithm (TPSB-EDA) to schedule a dynamic HFS. The estimation of distribution algorithm (EDA) is a relatively novel population-based evolutionary algorithm. Unlike traditional evolutionary algorithms, it samples new solutions from a probabilistic model which represents the distribution of promising solutions from the previous search. Due to its effectiveness and search ability, EDA has been successfully applied to a variety of academic and real-world problems during recent years, including flow shop scheduling problems (Hauschild and Pelikan 2011; Chen and Chen 2013). Jarboui, Eddaly, and Siarry (2009) proposed an EDA to minimise the total flow time in a permutation flow shop. Pan and Ruiz (2012) developed an efficient EDA for solving lot-streaming flow shop scheduling problems with set-up times. More recently, Wang et al. (2013a, 2013b) adopted EDAs to schedule the distributed permutation flow shop and the HFS with identical parallel machines, respectively. However, to the best of our knowledge, there is no research work about the EDA for solving dynamic HFS scheduling problems.

In this paper, we integrate the EDA with a discrete-event simulation model to minimise the makespan of HFS scheduling problems under stochastic processing times. The uniqueness of the proposed TPSB-EDA is characterised by the two-phase simulation model (TPSM) and the annealing selection mechanism (ASM). To avoid large computation cost when evaluating the candidate solutions, TPSM first employs a back propagation network (BPN) to identify a set of roughly good solutions and then evaluates them by a discrete-event simulation algorithm. To preserve the population diversity of EDA, ASM uses the Boltzmann probability in the annealing algorithm to select offspring individuals for constructing the probabilistic model.

The rest of this paper is organised as follows. The next section briefly reviews the relevant literature, focusing mainly on dynamic HFS scheduling. Section 3 is devoted to problem description. Section 4 provides a detailed explanation of the proposed TPSB-EDA for solving the considered problem. To evaluate the effectiveness of the TPSB-EDA, simulation is conducted and computation results are analysed in Section 5. Finally, conclusions are summarised and some directions of future work are discussed in Section 6.

2. Literature review

Ever since the HFS scheduling problem was identified in 1970s (Arthanari and Ramamurthy 1971), it has attracted considerable attention during the past decades (Wang 2005; Luo et al. 2012). However, only a few studies on dynamic HFS scheduling have been reported. Hence, we have to broaden the literature scope to include a variety of manufacturing systems, mainly focused on flow shops, job shops and flexible manufacturing systems.

To address dynamic scheduling problems, some promising and effective scheduling techniques, including heuristics, meta-heuristics, agent-based approaches and other artificial intelligence techniques, may be used.

Heuristics are strategies that help produce solutions. The most well-known scheduling heuristics are dispatching rules and schedule repair methods (Ouelhadj and Petrovic 2009). The dispatching rule is a typical completely reactive approach, in which jobs are selected by sorting them according to predefined criteria. As the dispatching rule can find a reasonably good solution in a relatively short time, it has already been widely used to solve dynamic HFS scheduling problems (Kianfar, Ghomi, and Karimi 2009; Kia, Davoudpour, and Zandieh 2010). Unfortunately, no single one is globally better than all the others. Therefore, a natural extension of the dispatching approach allows dynamic selection of a dispatching rule for schedule generation. The artificial intelligence techniques, such as case-based reasoning, neural networks and inductive learning, are commonly used to acquire the scheduling knowledge for the selection of dispatching rules (Tang, Liu, and Liu 2005; Priore et al. 2006; Mouelhi-Chibani and Pierreval 2010; Choi, Kim, and Lee 2011). Schedule repair methods are also heuristics that have played a significant role in dealing with uncertainties. The most common schedule repair methods include right-shift schedule repair, match-up schedule repair and partial schedule repair (Akturk and Gorgulu 1999; Vieira, Herrmann, and Lin 2003). The right-shift schedule repair postpones the remaining operations by the amount of time needed to make the schedule feasible. The match-up schedule repair re-schedules to match up with the pre-schedule at some point in the future. The partial schedule repair only reschedules the operations that are affected by the disruption.

Meta-heuristics are high-level heuristics that iteratively make use of heuristics or some rules to obtain better solutions. They are usually applied to generate predictive schedules, which can be rescheduled during execution in response to unexpected uncertainties (Vieira, Herrmann, and Lin 2003; Aytug et al. 2005). Besides, to generate robust schedules, some research works on dynamic HFS scheduling incorporated a discrete-event simulation model into a meta-heuristic. Allaoui and Artiba (2004) integrated a flexible simulation model with SA heuristic to address HFS scheduling with maintenance constraints. The simulation model was employed to evaluate solutions generated by SA. Gholami, Zandieh, and Alem-Tabriz (2009) presented a heuristic to solve HFS scheduling problems with sequence-dependent set-ups and stochastic machine breakdown. They incorporated a simulation algorithm into GA to evaluate the expected makespans. For the same scheduling problem, Zandieh and Gholami (2009) embedded a simulator into an immune algorithm to generate schedules. More recently, Ahmadizar, Ghazanfari, and Ghomi (2010) proposed a simulation optimisation approach to a stochastic group shop scheduling problem. This approach was a hybrid of an ACO algorithm, a heuristic algorithm to generate good solutions and a discrete-event simulation model to evaluate the solution performance. Dugardin, Yalaoui, and Amodeo (2010) developed three GA-based methods to address re-entrant HFS scheduling problems with stochastic processing times. In the proposed algorithms, solutions generated by GA were evaluated by multiple independent simulations.

Agent-based scheduling is another promising approach to constructing flexible and dynamic schedules. Since an agent-based scheduling system is made up of autonomous agents that coordinate dynamically to optimise both local and global objectives, its overall performance greatly depends on the coordination mechanism among agents.

One of the most commonly used coordination mechanisms is the Contract Net Protocol (CNP) (Smith 1980). In the original CNP, a manager broadcasts an available task to all capable contractors. Then each contractor bids for the task in the bidding phase. After the manager has collected all the bids from the contractors and compared them using some predefined criteria, the task is awarded to the best contactor. To make the coordination more efficient, a variety of modified CNPs (Lau et al. 2006; Wong et al. 2006; Leitão and Restivo 2008), such as the multi-round and multi-task coordination, have been developed to solve scheduling problems in dynamic manufacturing environments. Since the agents have to exchange bids or other information for negotiation, the CNP-based coordination cannot avoid the limitation of communication latency (Shen 2002; Xiang and Lee 2008). Hence, the insect-inspired coordination, enlightened by the behaviour of social insects, such as ants and wasps, has aroused much research interest. The insect-inspired coordination mechanism allows agents to use indirect communication, such as the stigmergy of the ant colony, to achieve their goals. Some insect-inspired multi-agent prototypes can be found in Xiang and Lee (2008) and Rajabinasab and Mansour (2011).

To establish better dynamic scheduling systems, some artificial intelligence techniques, such as neural networks, fuzzy logic and data mining, have also been applied to acquire scheduling knowledge. Li, Chen, and Lin (2003) used neural networks and IF-THEN scheduling rules to learn robust scheduling knowledge in dynamic manufacturing systems. To address a dynamic job shop scheduling problem, Zandieh and Adibi (2010) proposed a variable neighbourhood search (VNS) in which an artificial neural network (ANN) was adopted to update VNS parameters at any rescheduling point. Duenas, Petrovic, and Petrovic (2005) applied a fuzzy set to model the uncertainty of raw material shortage in a real-life scheduling problem of a pottery company. Lipi, Ahsan Akhtar Hasin, and Noor-E-Alam (2009) established two fuzzy inference systems to obtain the job priority and machine priority for HFS scheduling, respectively. Sha and Liu (2005) developed a dynamic job shop scheduling model that incorporated a data mining tool for mining the knowledge about due date assignment. Wang and Choi (2012) presented a novel decomposition-based approach (DBA) to minimising the makespan of a dynamic HFS. In the proposed approach, a modified K-means clustering algorithm was applied to decompose an HFS scheduling problem into sub-problems.

As a new emerging research area, dynamic scheduling in manufacturing systems has recently attracted increasing attention. However, an extensive literature review indicates that no research works have adopted EDAs to address dynamic HFS scheduling problems. This paper therefore attempts to integrate the EDA with a discrete-event simulation model to solve HFS scheduling problems under stochastic processing times.

3. Problem description

A typical HFS system is defined by a set $S = \{1, 2, \ldots, t\}$ of t ($t \geq 2$) processing stages. At each stage k, $k \in S$, there is a machine set S_k of m_k parallel machines, where $m_k \geq 2$ in at least one stage. The set $J = \{1, 2, \ldots, n\}$ of n independent jobs has to be sequentially processed on machines of the sets $S_1, \ldots, S_t$. Each job j, $j \in J$, has its release time and its processing time at each stage k is deterministic. In a real-world HFS, however, the actual processing time of a job on a machine might be uncertain due to some unpredictable events, such as quality issues, machine breakdown, tool wear and operator unavailability (Lawrence and Sewell 1997). This paper therefore considers the HFS scheduling problem under stochastic processing times.

To model the stochastic processing times in an HFS, the coefficient of processing time variation (CPTV) (Wang and Choi 2012, 2014) is adopted in this study. The CPTV is defined as CPTV = $\sigma/E(P)$, where $E(P)$ denotes the expected processing time and σ represents the standard deviation of the actual processing times. As an indicator to processing time uncertainty, a large CPTV tends to lead to a large deviation between the realised and the planned schedules.

3.1 *Assumptions*

To address the HFS scheduling problems under stochastic processing times, we make the assumptions:

Assumption 1	All machines are immediately available for continuous processing when jobs are arrived simultaneously in an HFS;
Assumption 2	No machine can perform more than one operation at a time;
Assumption 3	An operation is not allowed to be processed on more than one machine;
Assumption 4	For the same job, any parallel machine at a stage takes equal processing time;
Assumption 5	Both the set-up time for job processing and the travel time between stages are negligible;
Assumption 6	To avoid the blocking of machines, there are infinite buffers between stages;
Assumption 7	Job processing may take longer or shorter processing time than expected;
Assumption 8	Since a stage consists of a number of functionally identical parallel machines, these machines have the same CPTV. The CPTV, however, may be different for the machines at other stages.

3.2 *Notations*

The notations for the problem formulation are defined as follows:

k	stage index, $1 \leq k \leq t$
i, i_1, i_2	machine index, $1 \leq i, i_1, i_2 \leq m_k$
j, j_1, j_2	job index, $1 \leq j, j_1, j_2 \leq n$
m_k	number of parallel machines at stage k
S_k	set of parallel machines at stage k
C_{kj}	completion time of job j at stage k
$A_{kij_1j_2}$	a Boolean variable, 1 if job j_2 starts to be processed immediately after job j_1 on machine i at stage k, and 0 otherwise
F_{kij}	a Boolean variable, 1 if job j is first processed on machine i at stage k, and 0 otherwise
SP_{kj}	stochastic processing time of job j at stage k
SP_{kij}	stochastic processing time of job j on machine i at stage k
$E(P_{kj})$	expected processing time of job j at stage k
$E(P_{kij})$	expected processing time of job j on machine i at stage k
ST_{kij}	start time of job j on machine i at stage k

3.3 *Mathematical formulation*

Using the above notations, the HFS scheduling problem under stochastic processing times is formulated as follows:

$$\min\{E(\max[C_{tj}]), \quad j = 1, 2, \ldots, n\} \tag{1}$$

Subject to the following constraints:

$$C_{1j} = \mathrm{SP}_{1j}, \quad \text{if } \sum_{i=1}^{m_1} F_{1ij} = 1 \tag{2}$$

$$C_{1j_2} = \sum_{i=1}^{m_1}\sum_{j_1=1}^{n}(A_{1ij_1j_2} \times C_{1j_1}) + SP_{1j_2}, \quad \text{if } \sum_{i=1}^{m_1} F_{1ij_2} = 0 \tag{3}$$

$$C_{kj} = C_{(k-1)j} + \mathrm{SP}_{kj}, \quad \text{if } k > 1 \ \& \ \sum_{i=1}^{m_k} F_{kij} = 1 \tag{4}$$

$$C_{kj_2} = \max\left\{\sum_{i=1}^{m_k}\sum_{j_1=1}^{n}(A_{kij_1j_2} \times C_{kj_1}),\ C_{(k-1)j_2}\right\} + \mathrm{SP}_{kj_2}, \quad \text{if } \ k > 1 \ \& \ \sum_{i=1}^{m_k} F_{kij_2} = 0 \tag{5}$$

$$\mathrm{ST}_{(k+1)i_1j} - \mathrm{ST}_{ki_2j} \geq \mathrm{SP}_{ki_2j} \tag{6}$$

$$[(\mathrm{ST}_{kij_1} - \mathrm{ST}_{kij_2}) \geq \mathrm{SP}_{kij_2}] \quad \text{or} \quad [(\mathrm{ST}_{kij_2} - \mathrm{ST}_{kij_1}) \geq \mathrm{SP}_{kij_1}] \tag{7}$$

$$E(P_{ki_1j}) = E(P_{ki_2j}) = E(P_{kj}),\ (i_1, i_2) \in S_k \tag{8}$$

The objective function (1) is to minimise the expected makespan, which is equivalent to the maximum job completion time at the last stage under stochastic processing times. For any machine at the first stage, the completion time of the first job and that of each successive job is determined by Constraints (2) and (3), respectively. Similarly for any machine at all other stages, the completion time of the first job and that of each successive job is given using Constraints (4) and (5), respectively. In addition, Constraint (6) indicates the relations of job start times at two successive stages, while Constraint (7) shows the relation of job start times on a machine. Lastly, Constraint (8) ensures that the same job processed on any parallel machines at a stage requires equal processing time. To model the uncertainty of processing time in an HFS, SP_{kj} and SP_{kij}, Constraints (2)–(7) are sampled from a normal distribution, which are detailed in Section 5.1.

4. The proposed TPSB-EDA

EDA, proposed by Mühlenbein and Paass (1996), has emerged as a prominent alternative to traditional evolutionary algorithms. Instead of using the conventional crossover and mutation operations of GA, EDA reproduces new populations based on a probabilistic model, which is established by explicitly extracting global statistical information from a population of parents. The general procedure of EDA consists of the following iterative steps (Zhang and Li 2011):

(1) *Initialisation*: randomly generate an initial population.
(2) *Probabilistic model construction*: select part of the parent individuals to construct the elite set, from which a probabilistic model representing the individual distribution of the elite set is established.
(3) *Offspring production:* generate new offspring according to the probabilistic model.
(4) *Local search (optional):* conduct the local search on offspring individuals to enhance the local exploitation.
(5) *Replacement*: replace part/all of the individuals in the current population with the new offspring.
(6) *Termination check*: repeat from step 2 until some termination criterion is met.

To address the large computation complexity of simulation-based meta-heuristics, the proposed TPSB-EDA incorporates a TPSM into the EDA to evaluate the populations in a dynamic HFS manufacturing environment. Moreover, to avoid early search stagnation, an ASM is applied to establish probabilistic models. The framework of TPSB-EDA is illustrated in Figure 1. In this section, we first present the details of TPSM and EDA, and then summarise the procedure of TPSB-EDA.

4.1 *The TPSM*

To reduce the computational burden incurred from carrying out simulation replications, the proposed TPSB-EDA adopts a TPSM to estimate the performance of candidate solutions under stochastic processing times. In the first phase, a set of roughly good solutions is identified by a BPN-based rough model (BBRM) without consuming much computation time, and in the second phase, only these selected solutions are further evaluated by a discrete-event simulation algorithm.

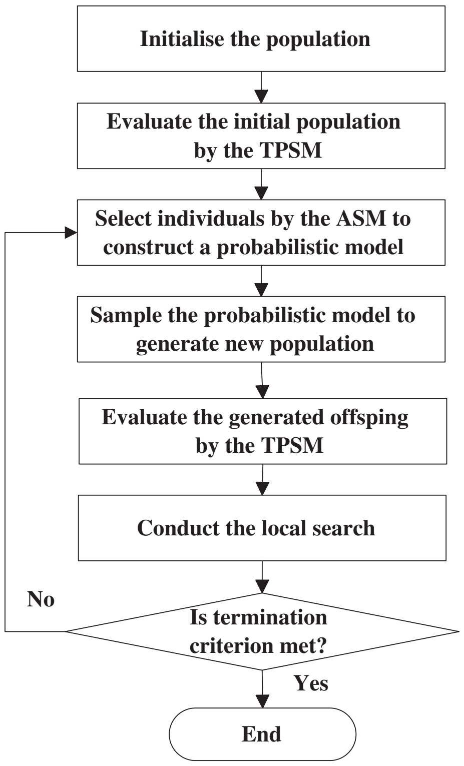
Initialise the population
Evaluate the initial population by the TPSM
Select individuals by the ASM to construct a probabilistic model
Sample the probabilistic model to generate new population
Evaluate the generated offsping by the TPSM
Conduct the local search
Is termination criterion met?
No
Yes
End

Figure 1. The framework of TPSB-EDA.

4.1.1 *Phase I: obtaining a set of roughly good solutions*

The occurrence of uncertainties affects the manufacturing system status and causes deviations from the planned schedules. Such deviations, if not effectively absorbed by the idle time or slack in the planned schedules, could in turn result in performance deterioration (PD) (Szelke and Márkus 1997). Less idle time or slack in the planned schedules and increasing system uncertainty, such as large CPTV, may lead to significant PD.

To evaluate the schedule performance in a dynamic manufacturing environment, it is essential to establish an effective model to estimate the PD of a given schedule. ANNs are powerful computational models in solving a variety of problems with difficult or unknown analytical solutions. The ANNs have recently attracted much attention due to the capability of identifying complex nonlinear relationships and predicting underlying trend (Schmitz and Aldrich 1999). As a commonly used ANN, the BPN has been successfully used for modelling, prediction and optimisation of complex systems (Lin and Hwang 1999). Therefore, to obtain a set of roughly good solutions, the BBRM firstly employs a BPN to estimate the PD of a given schedule under stochastic processing times. The makespan of a solution is subsequently described as

$$M_{\mathrm{AS}} = M_{\mathrm{PS}} \times (1 + \mathrm{PD}) \tag{9}$$

where PD is the performance deterioration, and M_{AS} and M_{PS} are the makespans of the actual and planned schedules, respectively. Thus, the roughly good solutions can be easily obtained according to M_{AS}. As illustrated in Figure 2, the details of BPN establishment are as follows:

- Inputs: The inputs consist of five parameters, namely the number of stages, the number of jobs, the number of parallel machines per stage, CPTV, and slack ratio (SR). The SR, indicating the slack per processing time, is measured by

$$SR = \sum_{k=1}^{t} \sum_{j=1}^{n} S_{kj}/P_{kj} \tag{10}$$

where S_{kj} represents the free slack of job j at stage k, P_{kj} denotes the processing time of job j at stage k, t and n are the number of stages and the number of jobs, respectively. Since the free slack is widely used to measure

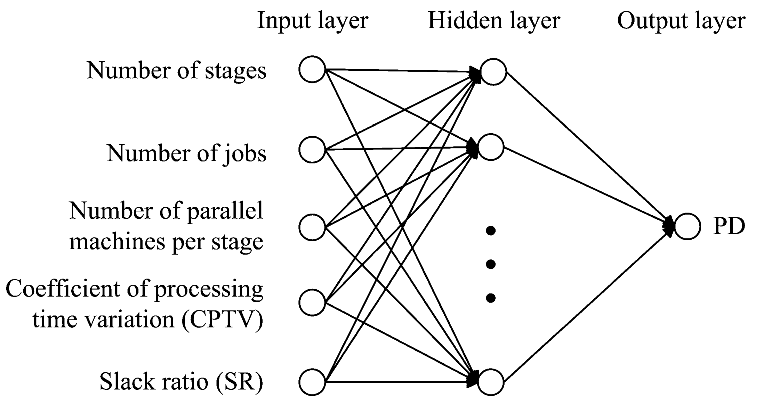
Input layer
Hidden layer
Output layer
Number of stages
Number of jobs
Number of parallel machines per stage
Coefficient of processing time variation (CPTV)
Slack ratio (SR)
PD

Figure 2. The BPN to estimate the PD.

schedule robustness in the scheduling literature (Hazır, Haouari, and Erel 2010), it is adopted to compute the SR in this study. Different from idle time, the free slack in an HFS is the amount of time that a job could be delayed without affecting its earliest start time at the next stage. Figure 3 provides an example to illustrate the difference between idle time and free slack, which are both represented by grey blocks.

- Number of hidden layers: Since one hidden layer is capable of establishing an arbitrary mapping between inputs and outputs for most applications (Wang and Huang 2007), the BPN to predict the PD of a given schedule only has one hidden layer.
- Number of hidden neurons: 2–20. The optimal numbers of hidden neurons required for successful ANN implementations vary with the problems to be solved, and are therefore usually determined experimentally by trial-and-error. In this study, we first train a number of BPNs with different numbers of hidden neurons ranging from 2 to 20, and then determine the optimal one using the minimum mean square error (MSE). As a good performance measure of the model during the training process, the MSE is computed as

$$\mathrm{MSE} = \sqrt{\sum_{i=1}^{n} (x_i - E(i))^2 / n} \tag{11}$$

where n is the number of samples to be evaluated in the training phase, x_i represents the target output related to the sample i ($i = 1, \ldots n$), i.e. the actual PD, $E(i)$ is the model output, i.e. the estimated PD. Among these BPNs, the one that gives the least minimum MSE is selected as the optimal BPN.

- Output: PD. For the same solution to an HFS scheduling problem, M_{AS} and M_{PS} are obtained under stochastic processing times and deterministic processing times, respectively. The PD is subsequently measured as

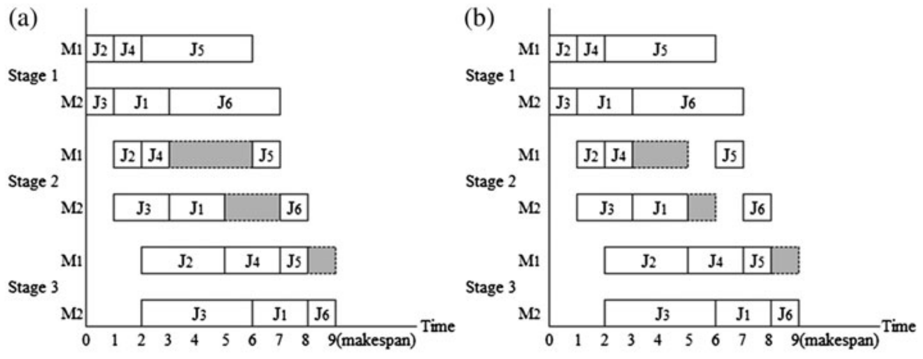

Figure 3. An example of the difference between idle time and free slack. (a) Idle time in a schedule; (b) free slack in a schedule.

$$\mathrm{PD} = M_{\mathrm{AS}}/M_{\mathrm{PS}} - 1 \tag{12}$$

- Number of replications: 100. Since the initial network condition greatly affects the performance of a BPN, 100 BPNs with a specific number of hidden neurons are trained under different initial conditions. To identify the optimal BPN for PD estimation, only the one with minimum MSE is kept for further evaluation.
- Number of epochs per replication: 10,000.
- Training examples: The training examples of the BPN are generated by the procedure discussed in Section 5.2.

Both the number of replications and the number of epochs per replication are chosen empirically. These values are capable of generating the BPNs with satisfactory performance within an acceptable training time. When the optimal BPN for PD estimation is established, the candidate solutions of EDA are roughly evaluated based on Formula (9), and the best $\alpha \times 100\%$ ($\alpha \in [0,1]$) ones are identified to construct a set of roughly good solutions Π_r.

4.1.2 *Phase II: performance evaluation using a discrete-event simulation algorithm*

The solutions of Π_r obtained in Phase I are further evaluated by a discrete-event simulation algorithm. This algorithm is applied to mimic the production process in an HFS under stochastic processing times, which is detailed as follows:

Algorithm I: The discrete-event simulation algorithm

```
Step 1   Initialise counter $n = 0$ and set the number of simulation replications $N_{\mathrm{sim}}$.
Step 2   Set the coefficient of processing time variation ($\mathrm{CPTV}_k$) for the machines at each stage $k$.
Step 3   Assign the jobs to machines using the job sequence of a candidate solution.
Step 4   Following the processing route (from the first stage to the last stage), run the completion time algorithm to obtain the actual job completion times on each machine $M_{ki}$, representing the $i$th machine at stage $k$.
Step 5   Record the simulated makespan as $M^n_{\mathrm{sim}}$, which is the maximum job completion time at the last stage.
Step 6   If $n \geq N_{\mathrm{sim}}$, return the average simulated makespan $M_{\mathrm{avg}} = \sum_{n=1}^{N_{\mathrm{sim}}} M^n_{\mathrm{sim}}/N_{\mathrm{sim}}$; otherwise set $n = n + 1$ and go to step 3.
```

Algorithm II: The completion time algorithm

```
Step 1   Identify the first unprocessed job $j$ on machine $M_{ki}$.
Step 2   Record the start time of job $j$ when it is available to be processed on machine $M_{ki}$ as scheduled.
Step 3   Generate the actual processing time of job $j$ on machine $M_{ki}$, which follows a normal distribution. The details of the normal distribution are shown in Section 5.1.
Step 4   Compute the finish time of job $j$ on machine $M_{ki}$ by adding its actual processing time to its start time.
Step 5   If all the allocated jobs on machine $M_{ki}$ have been processed, return their completion times; otherwise, go to step 1.
```

For each solution of Π_r, its performance is measured by M_{avg}. To mimic the real-life manufacturing environment, the discrete-event simulation algorithm generally adopts a large N_{sim}, inevitably leading to a time-consuming simulation process.

4.2 ***The EDA with TPSM and ASM***

To solve HFS scheduling under stochastic processing times, the EDA integrated with TPSM and ASM is described in details as follows.

4.2.1 *Representation and population initialisation*

The job permutation-based scheme has been widely used in the literature for a variety of flow shop scheduling problems (Jungwattanakit et al. 2008; Qian et al. 2009; Pan and Ruiz 2012). Therefore, it is adopted to encode individual solutions in this study. For example, as a solution to an HFS scheduling problem with 10 jobs, job sequence {10, 9, 4, 2, 5, 3, 1, 8, 7, 6} indicates the job processing order by available machines. To guarantee the diversity of the initial population, all the individuals are randomly generated. Such a method is also applied for population initialisation by Jarboui, Eddaly, and Siarry (2009) and Zhang and Li (2011).

4.2.2 *The ASM*

The probabilistic model construction depends on the genetic information of the elite set chosen from the population by a selection operator. The commonly used way to construct the elite set is to select a number of best individuals from the population (Zhang and Li 2011; Pan and Ruiz 2012). However, as the population of the EDA evolves over generations, such a selection operator makes the diversity of elite set diminished and eventually results in search stagnation (Chen et al. 2010).

To maintain the diversity of elite set, the ASM is established by using Boltzmann probability in the annealing algorithm to identify elite individuals. Instead of directly selecting a number of best individuals, the ASM accepts some worse solutions to make the EDA explore the solution space more extensively. The selection procedure of the ASM involves two steps: (1) use the TPSM to evaluate the current population under stochastic processing times and sort individuals in ascending order of their estimated fitness, and (2) following the above order, the individuals are accepted with the Boltzmann probability:

$$P(x_k^i) = e^{-f_k^i/T_k} \Big/ \sum_{x_k^i \in P_k} e^{-f_k^i/T_k} \tag{13}$$

where P_k represents the population at generation k in EDA, x_k^i denotes the ith individual of population P_k, f_k^i is the fitness of x_k^i under stochastic processing times and T_k is the annealing temperature at generation k in EDA. As the objective function is to minimise the makespan, f_k^i is defined as $1/C_{\max}^{ik}$, where $C_{\max}^{ik}$ is the makespan of x_k^i. During the annealing process, T_k decreases according to a linear cooling strategy, $T_{k+1} = \beta T_k$, where $\beta \in (0, 1)$ indicates the rate at which the temperature drops. Such process continues until $\gamma \times 100\%$ ($\gamma \in [0, 1]$) individuals have been identified.

4.2.3 *Probabilistic model*

Instead of crossover and mutation operators used in GA, EDA adopts a probabilistic model to generate the offspring. Since this model is established by probabilistic modelling of promising solutions, it has a great effect on the performance of EDA. To minimise the total flow time of the permutation flow shop scheduling problems, Jarboui, Eddaly, and Siarry (2009) established a probabilistic model considering both the order of the jobs in the sequence and similar blocks of jobs presented in the selected parents. In this model, the probability for positioning job j in the kth position of the offspring is determined by

$$\pi_{jk} = \frac{\eta_{jk} \times \mu_{j[k-1]}}{\sum_{l \in \Omega_k} (\eta_{lk} \times \mu_{l[k-1]})} \tag{14}$$

where η_{jk} is the number of times that job j is before or in position k augmented by a given constant δ_1, $\mu_{j[k-1]}$ is the number of times that job j is immediately after the job in position $k-1$ augmented by a given constant δ_2 and Ω_k is the set of the unscheduled jobs until position k.

The probabilistic model introduced by Jarboui, Eddaly, and Siarry (2009) may be further improved when applied to solve HFS scheduling problems. In this study, a schedule is obtained by consecutively assigning jobs in a job sequence to available machines. As assumed in Assumption 1 that all machines are available when jobs are released to an HFS and Assumption 4 that the parallel machines at a stage take equal time to process the same job, the first jobs on the machines at the first stage can be interchanged without affecting the individual fitness for a given job sequence. For example, suppose in an HFS with four jobs and two parallel machines at each stage, job sequences [2, 3, 4, 1] and [3, 2, 4, 1] are two individual solutions and can be decoded into two schedules, as illustrated in Figure 4. Since job 2 and job 3 are firstly processed on the machines at the first stage, such two job sequences lead to the same makespan.

To consider possible job interchanges that have no effect on the individual fitness, a new probabilistic model with a learning strategy is established, which is described as follows:

$$\pi_{jk}(g+1) = \begin{cases} (1-\eta) \times \pi_{jk}(g) + \frac{\eta \times \rho_{jk}}{\sum_{l \in \Omega_k} \rho_{lk}}, & 1 \le k \le m_1 \\ (1-\eta) \times \pi_{jk}(g) + \frac{\eta \times \rho_{jk} \times \tau_{j[k-1]}}{\sum_{l \in \Omega_k} (\rho_{lk} \times \tau_{l[k-1]})}, & m_1 < k \le n \end{cases} \tag{15}$$

where $\pi_{jk}(g)$ represents the probability for positioning job j in position k of the offspring at generation g, $\eta \in (0, 1)$ denotes the learning rate, ρ_{jk} represents the number of times that job j is before or in position k, $\tau_{j[k-1]}$ denotes the number of times that job j is immediately after the job in position $k-1$ and m_1 is the number of parallel machines at the first

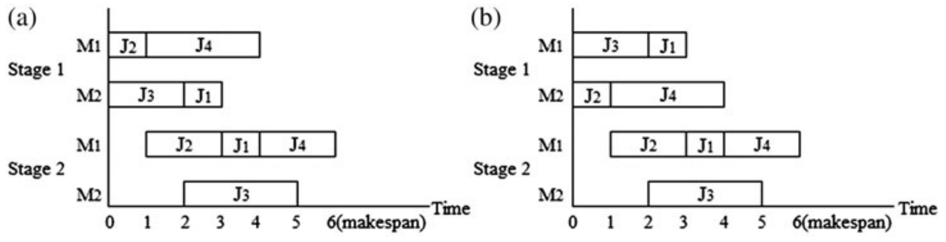

Figure 4. Gantt chart of two solutions. (a) Decoding the solution [2, 3, 4, 1]; (b) decoding the solution [3, 2, 4, 1].

stage. Thus, for each position k of a solution sequence at generation g, job j from the set of unscheduled jobs is selected with the probability of $\pi_{jk}(g)$. In this study, a population of P_s individuals are generated at each iteration by the proposed probabilistic model.

4.2.4 *Local search*

To further improve the performance of EDA, an efficient way is to add a local search method into EDA to enhance the local exploitation. The insertion operator is commonly used to develop local search methods in the flow shop literature (Jarboui, Eddaly, and Siarry 2009; Zhang and Li 2011), and hence it is adopted to generate neighbouring solutions. The insertion operator, as illustrated in Figure 5, randomly selects a job and shifts it from its original position to a new randomly chosen position.

Integrated with the insertion operator, a fast insertion-based local search (IBLS) is described as follows:

```
Algorithm III: IBLS

Step 1   Initialise counter $c = 1$.
Step 2   Apply the insertion operator to the best individual $S_c$ of the current population. All possible insertions of $S_c$ are evaluated
         by the TPSM with $\alpha = 0.02$, and the best obtained individual solution is denoted as $S^*$.
Step 3   If $S^*$ is better than the best solution $S_b$ found so far, set $S_b = S^*$.
Step 4   If $S_b$ was improved in step 2, set $c = 1$; otherwise, $c = c + 1$.
Step 5   If $c < N_g$, go to step 2; otherwise, output $S_b$.
```

In the above local search procedure, N_g denotes the number of consecutive iterations without improvement at generation g. Since the solutions obtained in the earlier generations are often of low quality, there is no need to apply the insertion operator too intensively. Therefore, to enhance the exploitation ability of EDA, an adaptive strategy is applied to adjust the parameter N_g as follows:

$$N_g = 10 \times (1 + \text{CurGen}/\text{MaxGen}) \tag{16}$$

where CurGen and MaxGen are the number of current generation and the total number of generations, respectively.

4.3 *Description of the proposed TPSB-EDA*

Based on the above description, the proposed TPSB-EDA is summarised as follows:

```
Algorithm IV: TPSB-EDA

Step 1   Initialise CurGen = 0 and set the values of $N_{\text{sim}}$, $P_s$, $T_0$, $\alpha$, $\beta$, $\gamma$, $\eta$, MaxGen.
Step 2   Randomly generate a population of $P_s$ individuals.
Step 3   Apply the TPSM to evaluate the initial population and select the best one as $S_b$.
Step 4   Select $\gamma \times P_s$ individuals with the Boltzmann probability shown in formula (13) and construct the elite set $\Pi_e$.
Step 5   Establish the probabilistic model based on $\Pi_e$ using Formula (15).
Step 6   Sample and generate $P_s$ offspring from the probabilistic model and adopt the TPSM to identify the best individual $S_c$ of the
         current population.
Step 7   Perform IBLS to $S_c$.
Step 8   If CurGen ≥ MaxGen, return $S^b$ found so far; otherwise set CurGen = CurGen + 1 and go to step 4.
```

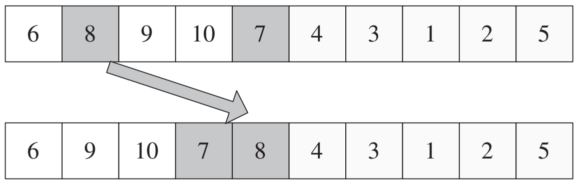

Figure 5. An example of insertion operator.

5. Computational results and analysis

5.1 *Design of experiments*

To evaluate the performance of the proposed TPSB-EDA, a prototype system is implemented in the Netbeans^{TM} development environment 7.2.1. Three experiments are designed and conducted using the developed prototype system. The first experiment establishes the optimal BPN, which is used to estimate the PD of a given schedule. The second experiment applies the Taguchi method to determine the values of the parameters in TPSB-EDA, and lastly the third experiment validates the effectiveness of TPSB-EDA under stochastic processing times.

To describe processing time uncertainty, the CPTV, defined as $\mathrm{CPTV} = \sigma/E(P)$, is adopted in this study. For instance, if $\mathrm{CPTV} = 0.2$ and the expected processing time of a job $E(P) = 10$ time units, the standard deviation σ of actual processing time is 2 time units. In the experiments above, the actual processing times are normally distributed with mean value $E(P)$ and standard deviation σ. To ensure that the actual processing times are nonnegative, the normal distribution is truncated at zero.

5.2 *Experiment I: obtaining the optimal BPN for PD Estimation*

To determine the optimal BPN for estimating the PDs of given schedules, training examples are generated first. The levels of five BPN inputs, including the number of stages, the number of jobs, the number of parallel machines per stage, CPTV and SR, are indicated in Table 1.

For each combination of BPN inputs, a training example is constructed using the following steps: (1) Generate an experimental HFS scheduling problem under stochastic processing times; (2) Randomly generate a job sequence and decode it into a schedule with a predefined SR; (3) For the same schedule, obtain the makespans under deterministic processing times and stochastic processing times, respectively; (4) Use Formula (12) to compute the PD, which is the output of BPN. This procedure continues until all the possible combinations of BPN inputs are considered. According to Table 1, a total of 4500 ($6 \times 6 \times 5 \times 5 \times 5 = 4500$) training examples are therefore generated for training, validating and testing the BPNs.

To identify the optimal BPN under normal processing times, the trial-and-error method is employed to generate a number of BPNs with different numbers of hidden neurons. The minimal MSE of these BPNs are shown in Figure 6. It clearly indicates that the optimal BPN has 12 hidden neurons. Such BPN is therefore adopted to estimate the PDs of given schedules.

5.3 *Experiment II: parameter setting*

The appropriate design of parameters has significant impact on the efficiency of meta-heuristics. The proposed TPSB-EDA has several key parameters, including P_s (the population size), T_0 (the initial temperature), β (the cooling rate), α (the parameter associated with determining roughly good solutions for further evaluation), γ (the parameter associated with updating the probabilistic model), η (the learning rate) and MaxGen (the total number of generations). To

Table 1. Levels of BPN inputs.

BPN inputs	Levels
Number of stages	2, 4, 6, 8, 10, 12
Number of jobs	20, 25, 30, 35, 40
Number of parallel machines per stage	2, 3, 4, 5, 6
CPTV	0.1, 0.2, 0.3, 0.4, 0.5
SR	0.1, 0.2, 0.3, 0.4, 0.5

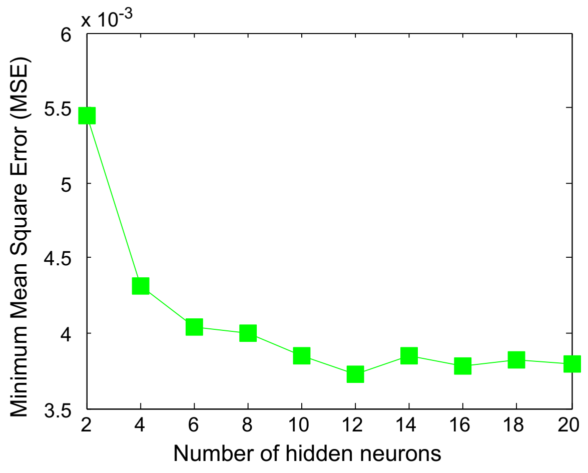

Figure 6. Minimum MSEs of BPNs with different numbers of hidden neurons.

investigate the effect of these parameters on the performance of TPSB-EDA and identify their appropriate values, we implement the Taguchi method (1986) based on a 30×8 (job × stage) HFS scheduling problem with two parallel machines at each stage. This method provides a systematic and efficient way to determine near optimum design parameters for performance and cost.

Table 2 illustrates the parameters and their factor levels considered in this study. According to the number of parameters and the number of factor levels, the orthogonal array L_{18} ($6^1 \times 3^6$) is established by MINITAB 16 for the adjustment of parameters. Instead of conducting $6^1 \times 3^6 = 4374$ experiments in a full factorial design, only 18 Taguchi experiments are required to find a suitable level of each parameter. In each Taguchi experiment, TPSB-EDA is run separately 20 times under different initial conditions and the mean simulated makespan is obtained as the response variable (RV) value. The orthogonal array and the obtained RV values are shown in Table 3.

According to Table 3, the trend of each factor level is illustrated in Figure 7. To analyse the significance rank of each parameter, the statistics of RV value of each parameter are obtained and listed in Table 4. The ranks in this table indicate the relative importance of each factor to the response. As shown in Table 4, α is the most significant one among the seven parameters and hence greatly affects the performance of TPSB-EDA. A large value of α could lead to a sufficient evaluation of offspring under stochastic processing times and build a more accurate probabilistic model. Figure 7 indicates that the improvement of solution performance is significantly weakened when α exceeds 0.3. Since a larger α usually increases the computation cost of TPSM, the parameter α of 0.3 is used to avoid long simulation time in this study. Similarly, it is necessary to consider the computation cost when determining the values of P_s and MaxGen. In addition, the values of other parameters can be directly identified according to Figure 7. Therefore, all the seven parameters are set as follows: $\alpha = 0.3$, $P_s = 150$, $T_0 = 150$, $\beta = 0.98$, $\gamma = 0.1$, $\eta = 0.1$, MaxGen = 300.

5.4 *Experiment III: analysis of TPSB-EDA*

The proposed TPSB-EDA is characterised by using TPSM and ASM to yield solutions. To validate the effectiveness of TPSB-EDA, we compare it with simulation-based EDA (SB-EDA), TPSB-EDA without ASM (TPSB-EDA_N) and DBA. The brief description of the compared algorithms is described as follows:

Table 2. Factors and their levels.

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	150	200	–	–	–
T_0	100	150	200	–	–	–
β	0.94	0.96	0.98	–	–	–
γ	0.10	0.15	0.20	–	–	–
η	0.10	0.20	0.30	–	–	–
MaxGen	100	300	500	–	–	–

Table 3. Orthogonal array L_{18} ($6^1 \times 3^6$).

	Factor							
Trial	α	P_s	T_0	β	γ	η	MaxGen	RV
1	1	1	1	1	1	1	1	752.7
2	1	2	2	2	2	2	2	746.3
3	1	3	3	3	3	3	3	753.5
4	2	1	1	2	2	3	3	747.8
5	2	2	2	3	3	1	1	744.4
6	2	3	3	1	1	2	2	736.1
7	3	1	2	1	3	2	3	740.7
8	3	2	3	2	1	3	1	739.3
9	3	3	1	3	2	1	2	738.9
10	4	1	3	3	2	2	1	742.6
11	4	2	1	1	3	3	2	739.2
12	4	3	2	2	1	1	3	735.6
13	5	1	2	3	1	3	2	738.2
14	5	2	3	1	2	1	3	739.4
15	5	3	1	2	3	2	1	738.7
16	6	1	3	2	3	1	2	740.5
17	6	2	1	3	1	2	3	736.2
18	6	3	2	1	2	3	1	739.0

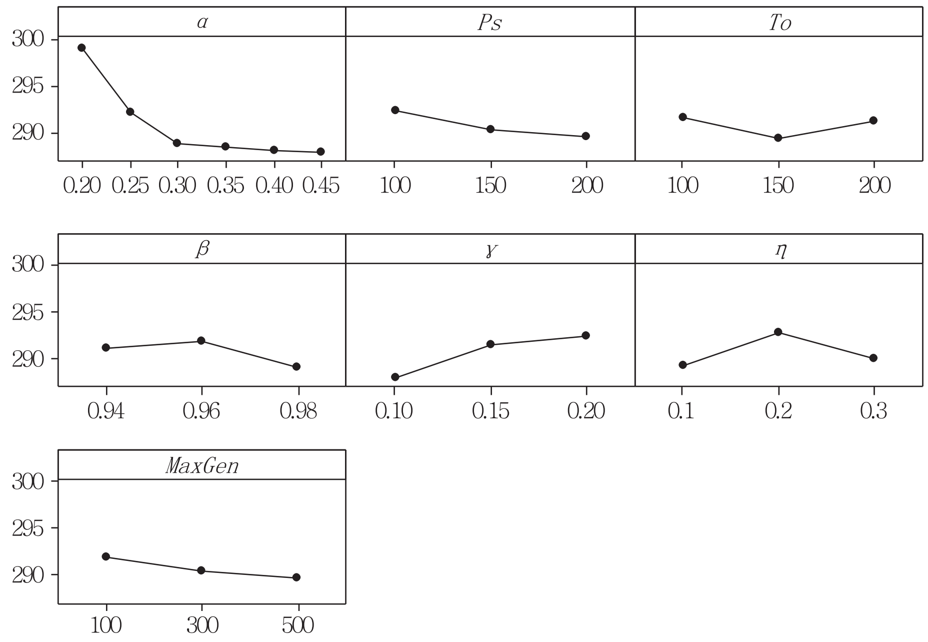

Figure 7. Factor level trend of the TPSB-EDA.

- SB-EDA: Compared with TPSB-EDA, this algorithm applies the time-consuming simulator to evaluate each candidate solution rather than a set of roughly good solutions. Although SB-EDA is superior to TPSB-EDA with respect to solution quality in nature, TPSB-EDA appears more efficient because it requires about $O(\alpha P_s)$ $(\alpha \in [0,1])$ time to evaluate candidate solutions, while such task requires $O(P_s)$ time for SB-EDA.
- TPSB-EDA_N: Instead of using the ASM, this algorithm constructs the elite set Π_e by directly selecting the $\gamma \times P_s$ best individuals of current population.

Table 4. Response value and rank of each parameter.

Level	Factor						
	α	P_s	T_0	β	γ	η	MaxGen
1	299.0	292.2	291.5	291.0	288.0	289.2	291.8
2	292.0	290.2	289.2	291.8	291.5	292.7	290.3
3	288.7	289.5	291.2	289.0	292.3	290.0	289.7
4	288.3	–	–	–	–	–	–
5	288.0	–	–	–	–	–	–
6	287.7	–	–	–	–	–	–
Delta	11.3	2.7	2.3	2.8	4.3	3.5	2.2
Rank	1	5	6	4	2	3	7

- DBA (Choi and Wang 2012): This approach employs a modified K-means clustering algorithm to decompose an HFS scheduling problem under stochastic processing times into sub-problems, which are addressed by either GA or shortest processing time heuristic.

This experiment is conducted on a test bed containing 27 HFS scheduling problems, as shown in Table 5. For each HFS scheduling problem in the test bed, the expected processing times of operations are randomly generated and the CPTVs are uniformly distributed in the interval [0.1, 0.5]. To evaluate the performance of SB-EDA, TPSB-EDA_N, TPSB-EDA and DBA, they are performed over 20 independent replications for each HFS scheduling problem. The relative performance deviation (RPD) is used to measure the performance of scheduling approaches as:

Table 5. Comparison of algorithms under normal processing times.

Problem no.	Problem size No. of jobs × no. of stages	No. of parallel machines at each stage	SB-EDA		TPSB-EDA_N		TPSB-EDA		DBA	
			ARPD	SD	ARPD	SD	ARPD	SD	ARPD	SD
1	20 × 4	2	0.15	0.12	2.12	0.54	0.41	0.25	1.87	0.54
2	20 × 4	3	0.21	0.13	2.28	0.62	0.50	0.21	1.15	0.39
3	20 × 4	4	0.00	0.00	1.81	0.55	0.11	0.10	0.30	0.21
4	20 × 8	2	0.06	0.08	3.77	1.08	1.22	0.52	3.46	1.06
5	20 × 8	3	0.23	0.15	2.14	0.72	0.37	0.24	1.67	0.54
6	20 × 8	4	0.00	0.00	2.41	0.68	0.14	0.11	1.58	0.64
7	20 × 12	2	0.26	0.16	3.96	1.13	1.35	0.57	4.61	1.35
8	20 × 12	3	0.31	0.18	2.81	0.86	0.91	0.64	2.46	0.76
9	20 × 12	4	0.25	0.14	3.36	1.04	1.11	0.45	2.41	0.74
10	30 × 4	2	0.17	0.13	4.28	1.16	0.96	0.47	4.28	1.39
11	30 × 4	3	0.27	0.19	3.07	0.97	0.65	0.35	2.51	0.91
12	30 × 4	4	0.56	0.32	3.31	0.94	0.48	0.29	1.55	0.44
13	30 × 8	2	0.18	0.12	3.65	1.13	1.45	0.81	3.37	1.13
14	30 × 8	3	0.54	0.32	3.22	1.07	0.82	0.33	3.52	1.07
15	30 × 8	4	0.34	0.18	3.05	0.91	0.97	0.48	2.64	0.91
16	30 × 12	2	0.36	0.24	4.02	1.23	1.68	0.83	5.49	1.67
17	30 × 12	3	0.29	0.17	3.57	1.05	1.07	0.64	4.17	1.15
18	30 × 12	4	0.65	0.39	3.36	1.09	0.96	0.55	3.25	0.96
19	40 × 4	2	0.23	0.21	3.91	1.27	1.63	0.97	4.23	1.37
20	40 × 4	3	0.24	0.13	3.77	1.17	0.92	0.42	3.17	0.97
21	40 × 4	4	0.61	0.36	2.65	0.93	0.45	0.16	2.54	0.86
22	40 × 8	2	0.34	0.18	3.48	1.08	1.27	0.56	3.92	1.28
23	40 × 8	3	0.54	0.35	2.67	0.87	0.76	0.61	2.78	0.89
24	40 × 8	4	0.36	0.24	2.38	0.75	0.92	0.53	2.10	0.73
25	40 × 12	2	0.39	0.20	4.06	1.12	1.46	0.59	3.56	1.05
26	40 × 12	3	0.56	0.33	3.11	0.96	1.17	0.47	2.43	0.77
27	40 × 12	4	0.42	0.27	2.37	0.82	0.95	0.32	2.06	0.69
Average			0.32	0.20	3.13	0.95	0.91	0.46	2.85	0.91

$$\text{RPD} = \frac{(S_{\text{alg}} - S_{\text{best}}) \times 100}{S_{\text{best}}} \tag{16}$$

where S_{alg} represents the makespan of the solution generated by a given algorithm, and S_{best} denotes the best makespan among the solutions obtained by all the compared algorithms. It is clear that the lower the RPD, the better result the algorithm produces.

Table 5 presents the experiment results of these four algorithms under stochastic processing times. The ARPD and SD represent the average and standard deviation of RPD, respectively. From the experiment results, the following conclusions can be made: (1) compared with TPSB-EDA_N and TPSB-EDA, SB-EDA yields better results in terms of ARPD and SD. Such good performance of SB-EDA results from using the simulator to evaluate all the offspring individuals; (2) there is no significant performance difference between SB-EDA and TPSB-EDA. This indicates that TPSM provides an efficient approach to evaluating the performance of schedules under stochastic processing times; (3) TPSB-EDA performs better than TPSB-EDA_N, which implies the effectiveness of the proposed ASM; (4) compared with DBA, TPSB-EDA shows a significant reduction in makespan. Such reduction underscores the superiority of the proposed TPSB-EDA.

6. Conclusion

This paper presented a novel TPSB-EDA for makespan minimisation of HFS scheduling problems under stochastic processing times. The proposed TPSB-EDA employs a TPSM to reduce the computation cost in evaluating offspring individuals of EDA. In this model, a set of roughly good offspring individuals are first identified based on an optimal BPN, and then evaluated by a discrete-event simulation algorithm. Moreover, to preserve the population diversity of EDA, TPSB-EDA adopts an ASM, in which the Boltzmann probability in the annealing algorithm is used to select part of population to establish the probabilistic model.

TPSB-EDA has been validated on a test bed of 27 HFS scheduling problems under normally distributed processing times. We compared TPSB-EDA with three scheduling algorithms, namely SB-EDA, TPSB-EDA_N and DBA. The experiment results indicate that the TPSB-EDA provides a reasonable trade-off between solution quality and computational efficiency. The good performance of TPSB-EDA is mainly due to the efficiency of TPSM in evaluating the offspring individuals under stochastic processing times, and ASM in applying annealing selection to avoid early search stagnation.

TPSB-EDA provides great academic and practical potential to solve dynamic HFS scheduling problems. Future research can focus on investigating the effectiveness of TPSB-EDA with other criteria, such as flow time-related and tardiness-related criteria. Moreover, rather than the proposed multi-layer perceptron (MLP) network with backpropagation, some other feed-forward networks need to be further investigated in estimating the PD of a given schedule. Since the radial basis function (RBF) network can be trained relatively faster than the MLP network (Mehrsai et al. 2013), another promising research area is to adopt the RBF network for PD estimation.

Acknowledgements

We would like to thank the anonymous reviewers and the editor for their constructive and pertinent comments.

Funding

This research is supported by Key Program from National Natural Science Foundation of China [grant number 71231007], [grant number 71131004], [grant number 71301124]; Humanities and Social Sciences Foundation of the Ministry of Education of China [grant number 13YJC630165]; Fundamental Research Funds for the Central Universities [grant number 2012GSP026].

References

Ahmadizar, F., M. Ghazanfari, and S. M. T. F. Ghomi. 2010. "Group Shops Scheduling with Makespan Criterion Subject to Random Release Dates and Processing times." *Computers & Operations Research* 37 (1): 152–162.

Akturk, M. S., and E. Gorgulu. 1999. "Match-up Scheduling under a Machine Breakdown." *European Journal of Operational Research* 112 (1): 81–97.

Allaoui, H., and A. Artiba. 2004. “Integrating Simulation and Optimization to Schedule a Hybrid Flow Shop with Maintenance Constraints.” *Computers & Industrial Engineering* 47 (4): 431–450.

Arthanari, T. S., and K. G. Ramamurthy. 1971. “An Extension of Two Machines Sequencing Problem.” *Opsearch* 8: 10–22.

Aytug, H., M. A. Lawley, K. McKay, S. Mohan, and R. Uzsoy. 2005. “Executing Production Schedules in the Face of Uncertainties: A Review and Some Future Directions.” *European Journal of Operational Research* 161 (1): 86–110.

Chen, S. H., and M. C. Chen. 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.

Chen, S. H., M. C. Chen, P. C. Chang, Q. Zhang, and Y. M. Chen. 2010. “Guidelines for Developing Effective Estimation of Distribution Algorithms in Solving Single Machine Scheduling Problems.” *Expert Systems with Applications* 37 (9): 6441–6451.

Choi, H. S., J. S. Kim, and D. H. Lee. 2011. “Real-time Scheduling for Reentrant Hybrid Flow Shops: A Decision Tree Based Mechanism and Its Application to a TFT-LCD Line.” *Expert Systems with Applications* 38 (4): 3514–3521.

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

Duenas, A., D. Petrovic, and S. Petrovic. 2005. “Analysis of Performance of Fuzzy Logic-based Production Scheduling by Simulation.” In *Lecture Notes in Computer Science 3789*, edited by A. Gelbukh and H. Terashima, 234–243. Berlin: Springer.

Dugardin, F., F. Yalaoui, and L. Amodeo. 2010. “New Multi-objective Method to Solve Reentrant Hybrid Flow Shop Scheduling Problem.” *European Journal of Operational Research* 203 (1): 22–31.

Garey, M. R., and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York: Freeman.

Gholami, M., M. Zandieh, and A. Alem-Tabriz. 2009. “Scheduling Hybrid Flow Shop with Sequence-dependent Setup Times and Machines with Random Breakdowns.” *The International Journal of Advanced Manufacturing Technology* 42 (1–2): 189–201.

Gupta, J. N. D. 1988. “Two-stage, Hybrid Flowshop Scheduling Problem.” *Journal of the Operational Research Society* 39 (4): 359–364.

Hauschild, M., and M. Pelikan. 2011. “An Introduction and Survey of Estimation of Distribution Algorithms.”*Swarm and Evolutionary Computation* 1 (3): 111–128.

Hazır, Ö., M. Haouari, and E. Erel. 2010. “Robust Scheduling and Robustness Measures for the Discrete Time/Cost Trade-off Problem.” *European Journal of Operational Research* 207 (2): 633–643.

Horng, S. C., S. S. Lin, and F. Y. Yang. 2012. “Evolutionary Algorithm for Stochastic Job Shop Scheduling with Random Processing Time.” *Expert Systems with Applications* 39 (3): 3603–3610.

Jarboui, B., M. Eddaly, and P. Siarry. 2009. “An Estimation of Distribution Algorithm for Minimizing the Total Flowtime in Permutation Flowshop Scheduling Problems.” *Computers & Operations Research* 36 (9): 2638–2646.

Jungwattanakit, J., M. Reodecha, P. Chaovalitwongse, and F. Werner. 2008. “Algorithms for Flexible Flow Shop Problems with Unrelated Parallel Machines, Setup Times, and Dual Criteria.” *The International Journal of Advanced Manufacturing Technology* 37 (3–4): 354–370.

Kia, H. R., H. Davoudpour, and M. Zandieh. 2010. “Scheduling a Dynamic Flexible Flow Line with Sequence-dependent Setup times: A Simulation Analysis.” *International Journal of Production Research* 48 (14): 4019–4042.

Kianfar, K., S. Fatemi Ghomi, and B. Karimi. 2009. “New Dispatching Rules to Minimize Rejection and Tardiness Costs in a Dynamic Flexible Flow Shop.” *The International Journal of Advanced Manufacturing Technology* 45 (7–8): 759–771.

Lau, J. S. K., G. Q. Huang, H. K. L. Mak, and L. Liang. 2006. “Agent-based Modeling of Supply Chains for Distributed Scheduling.” *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems & Humans* 36 (5): 847–861.

Lawrence, S. R., and E. C. Sewell. 1997. “Heuristic, Optimal, Static, and Dynamic Schedules When Processing Times are Uncertain.” *Journal of Operations Management* 15 (1): 71–82.

Leitão, P., and F. Restivo. 2008. “A Holonic Approach to Dynamic Manufacturing Scheduling.” *Robotics and Computer-Integrated Manufacturing* 24 (5): 625–634.

Li, D. C., L. S. Chen, and Y. S. Lin. 2003. “Using Functional Virtual Population as Assistance to Learn Scheduling Knowledge in Dynamic Manufacturing Environments.” *International Journal of Production Research* 41 (17): 4011–4024.

Lin, D. Y., and S. L. Hwang. 1999. “Use of Neural Networks to Achieve Dynamic Task Allocation: A Flexible Manufacturing System Example.” *International Journal of Industrial Ergonomics* 24 (3): 281–298.

Lipi, T. F., M. D. Ahsan Akhtar Hasin, and M. D. Noor-E-Alam. 2009. “Reliability Centered Multi Objective Hybrid Flow Shop Scheduling.” *Asia-Pacific Journal of Operational Research* 26 (5): 637–653.

Luo, H., G. Q. Huang, Y. Shi, T. Qu, and Y. F. Zhang. 2012. “Hybrid Flowshop Scheduling with Family Setup Time and Inconsistent Family Formation.” *International Journal of Production Research* 50 (6): 1457–1475.

Mehrsai, A., H. R. Karimi, K. D. Thoben, and B. Scholz-Reiter. 2013. “Application of Learning Pallets for Real-time Scheduling by the Use of Radial Basis Function Network.” *Neurocomputing* 101: 82–93.

Mouelhi-Chibani, W., and H. Pierreval. 2010. “Training a Neural Network to Select Dispatching Rules in Real Time.” *Computers & Industrial Engineering* 58 (2): 249–256.

Mühlenbein, H., and G. Paass. 1996. “From Recombination of Genes to the Estimation of Distributions I: Binary Parameters.” *Lecture Notes in Computer Science* 1141: 178–187.

Ouelhadj, D., and S. Petrovic. 2009. “A Survey of Dynamic Scheduling in Manufacturing Systems.” *Journal of Scheduling* 12 (4): 417–431.

Pan, Q. K., and R. Ruiz. 2012. “An Estimation of Distribution Algorithm for Lot-streaming Flow Shop Problems with Setup times.” *Omega* 40 (2): 166–180.

Priore, P., D. De La Fuente, J. Puente, and J. Parreño. 2006. “A Comparison of Machine-learning Algorithms for Dynamic Scheduling of Flexible Manufacturing Systems.” *Engineering Applications of Artificial Intelligence* 19 (3): 247–255.

Qian, B., L. Wang, D. X. Huang, W. L. Wang, and X. Wang. 2009. “An Effective Hybrid DE-based Algorithm for Multi-objective Flow Shop Scheduling with Limited Buffers.” *Computers & Operations Research* 36 (1): 209–233.

Rajabinasab, A., and S. Mansour. 2011. “Dynamic Flexible Job Shop Scheduling with Alternative Process Plans: An Agent-based Approach.” *The International Journal of Advanced Manufacturing Technology* 54 (9–12): 1091–1107.

Schmitz, G. P., and C. Aldrich. 1999. “Combinatorial Evolution of Regression Nodes in Feedforward Neural Networks.” *Neural Networks* 12 (1): 175–189.

Sha, D. Y., and C. H. Liu. 2005. “Using Data Mining for Due Date Assignment in a Dynamic Job Shop Environment.” *The International Journal of Advanced Manufacturing Technology* 25 (11–12): 1164–1174.

Shen, W. 2002. “Distributed Manufacturing Scheduling Using Intelligent Agents.” *IEEE Intelligent Systems and Their Applications* 17 (1): 88–94.

Smith, R. G. 1980. “The Contract Net Protocol: High-level Communication and Control in a Distributed Problem Solver.” *IEEE Transactions on Computers* C-29 (12): 1104–1113.

Szelke, E., and G. Márkus. 1997. “A Learning Reactive Scheduler Using CBR/L.” *Computers in Industry* 33 (1): 31–46.

Taguchi, G. 1986. *Introduction to Quality Engineering*. Tokyo: Asian Productivity Organization.

Tang, L., W. Liu, and J. Liu. 2005. “A Neural Network Model and Algorithm for the Hybrid Flow Shop Scheduling Problem in a Dynamic Environment.” *Journal of Intelligent Manufacturing* 16 (3): 361–370.

Vieira, G. E., J. W. Herrmann, and E. Lin. 2003. “Rescheduling Manufacturing Systems: A Framework of Strategies, Policies, and Methods.” *Journal of Scheduling* 6 (1): 39–62.

Wang, H. 2005. “Flexible Flow Shop Scheduling: Optimum, Heuristics and Artificial Intelligence Solutions.” *Expert Systems* 22 (2): 78–85.

Wang, K., and S. H. Choi. 2012. “A Decomposition-based Approach to Flexible Flow Shop Scheduling under Machine Breakdown.” *International Journal of Production Research* 50 (1): 215–234.

Wang, K., and S. H. Choi. 2014. “A Holonic Approach to Flexible Flow Shop Scheduling under Stochastic Processing times.” *Computers & Operations Research* 43: 157–168.

Wang, S. Y., L. Wang, M. Liu, and Y. Xu. 2013a. “An Effective Estimation of Distribution Algorithm for Solving the Distributed Permutation Flow-shop Scheduling Problem.”*International Journal of Production Economics* 145 (1): 387–396.

Wang, S. Y., L. Wang, M. Liu, and Y. Xu. 2013b. “An Enhanced Estimation of Distribution Algorithm for Solving Hybrid Flow-shop Scheduling Problem with Identical Parallel Machines.” *The International Journal of Advanced Manufacturing Technology* 68 (9–12): 2043–2056.

Wang, T. Y., and C. Y. Huang. 2007. “Applying Optimized BPN to a Chaotic Time Series Problem.” *Expert Systems with Applications* 32 (1): 193–200.

Wong, T. N., C. W. Leung, K. L. Mak, and R. Y. K. Fung. 2006. “Integrated Process Planning and Scheduling/Rescheduling – An Agent-based Approach.” *International Journal of Production Research* 44 (18–19): 3627–3655.

Xiang, W., and H. P. Lee. 2008. “Ant Colony Intelligence in Multi-agent Dynamic Manufacturing Scheduling.” *Engineering Applications of Artificial Intelligence* 21 (1): 73–85.

Zandieh, M., and M. Gholami. 2009. “An Immune Algorithm for Scheduling a Hybrid Flow Shop with Sequence-dependent Setup Times and Machines with Random Breakdowns.” *International Journal of Production Research* 47 (24): 6999–7027.

Zandieh, M., and M. A. Adibi. 2010. “Dynamic Job Shop Scheduling Using Variable Neighbourhood Search.” *International Journal of Production Research* 48 (8): 2449–2458.

Zhang, Y., and X. Li. 2011. “Estimation of Distribution Algorithm for Permutation Flow Shops with Total Flowtime Minimization.” *Computers & Industrial Engineering* 60 (4): 706–718.