Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2014, Article ID 619254, 10 pages
http://dx.doi.org/10.1155/2014/619254

*Research Article*

# Optimal Computing Budget Allocation for Ordinal Optimization in Solving Stochastic Job Shop Scheduling Problems

**Hong-an Yang, Yangyang Lv, Changkai Xia, Shudong Sun, and Honghao Wang**

*Department of Industrial Engineering, Northwestern Polytechnical University, Xi'an 710072, China*

Correspondence should be addressed to Hong-an Yang; yhongan@nwpu.edu.cn

Received 5 January 2014; Revised 21 February 2014; Accepted 22 February 2014; Published 24 March 2014

Academic Editor: Massimo Scalia



We focus on solving Stochastic Job Shop Scheduling Problem (SJSSP) with random processing time to minimize the expected sum of earliness and tardiness costs of all jobs. To further enhance the efficiency of the simulation optimization technique of embedding Evolutionary Strategy in Ordinal Optimization (ESOO) which is based on Monte Carlo simulation, we embed Optimal Computing Budget Allocation (OCBA) technique into the exploration stage of ESOO to optimize the performance evaluation process by controlling the allocation of simulation times. However, while pursuing a good set of schedules, "super individuals," which can absorb most of the given computation while others hardly get any simulation budget, may emerge according to the allocating equation of OCBA. Consequently, the schedules cannot be evaluated exactly, and thus the probability of correct selection (PCS) tends to be low. Therefore, we modify OCBA to balance the computation allocation: (1) set a threshold of simulation times to detect "super individuals" and (2) follow an exclusion mechanism to marginalize them. Finally, the proposed approach is applied to an SJSSP comprising 8 jobs on 8 machines with random processing time in truncated normal, uniform, and exponential distributions, respectively. The results demonstrate that our method outperforms the ESOO method by achieving better solutions.

## 1. Introduction

Most classical job shop scheduling problems assume that all the problem data are fixed and known in advance. However, there are several inevitable stochastic factors in real manufacturing systems, for example, random processing time, machine breakdowns, and rush orders, among which random processing time is the most fundamental and representative uncertain factor. Therefore, research in Stochastic Job Shop Scheduling Problem (SJSSP) with random processing time has a great importance in engineering applications. Even though, research works on SJSSP are fewer than those on deterministic JSSP because of disadvantage caused by random processing time, for example, huge search space, lengthy computation time, and challenging evaluation of schedules.

In general, three models are used to denote random processing time: interval number method [1, 2], fuzzy theory [3–6], and stochastic method. However, the fluctuation and distribution of random processing time are ignored by the first two mentioned methods, which leads to the inaccurate scheduling solutions. For this reason, an increasing number of researchers use stochastic methods to denote random processing time. Independent random distributions with a known mean and variance are used to represent processing duration variability [7, 8]. Among all processing time distributions, normal, exponential, and uniform distributions are commonly found in SJSSP literatures [2, 9–12]. In order to solve these NP-hard problems, many heuristic algorithms such as genetic algorithm [13], variable neighbourhood search [14], and artificial bee colony algorithm [15] are introduced to solve SJSSP.

However, random processing time of operations contributes to the randomness of completion time for each job and performance indicators of schedules, which leads to the difficulty of evaluating feasible schedules in the evolution of these heuristic algorithms. Stochastic simulation method, for example, Monte Carlo, which relies on repeated random sampling to obtain statistic results, has been widely used to solve performance evaluation [7, 9, 11].

Zhang and Wu [15] proposed that stochastic simulation definitely increased the computational burden because of frequent evaluations, especially when used in an optimization

framework. Ho et al. [16] firstly developed Ordinal Optimization (OO) theory to obtain good enough solutions through ordinal comparison while the value of a solution was still very poor. Due to the fact that Evolutionary Strategy (ES) could optimize the sampling process in OO theory, Horng et al. [8] embedded ES in Ordinal Optimization (OO), abbreviated as ESOO, to search for a good enough schedule of SJSSP using limited computation time.

Even though ESOO could significantly reduce the computation for SJSSP, the evaluation method in the exploration stage of ESOO is similar to Monte Carlo Simulation, in which uniform computation is allocated to each schedule, regardless of whether or not it should. Thus extra computation is allocated even after the Probability of Correct Selection (PCS) has already converged. Therefore, there is potential to further enhance ESOO's efficiency by intelligently controlling evaluation process or by determining the optimal number of simulation times among different schedules according to their performances. Chen et al. [17] firstly proposed Optimal Computation Budget Allocation (OCBA) to enhance the efficiency of OO by allocating simulation times reasonably. In the recent years there are many articles about OCBA [18–20]. In order to optimize the computation allocation in the evaluation process of solving SJSSP, we firstly propose an innovative hybrid algorithm of embedding OCBA into ESOO, abbreviated as ESOO-OCBA.

However, in the OCBA [17] deduction we find the hypothesis that the simulation times of the best individual outnumber considerably the average ones is not the case in specific SJSSP environment. As a result, "super individuals," which could absorb most of the given computation in each generation while others could hardly get any simulation budget, may emerge according to the allocating equation of OCBA [17]. Thus we make improvements on OCBA to balance the computation allocation: (1) *set a threshold of simulation times to detect "super individuals,"* (2) *follow an exclusion mechanism to isolate them,* and (3) *allocate the existing computation budget to the remained individuals according to the dispatching rules of OCBA*. The proposed modification on OCBA is another contribution of this paper, which makes ESOO-OCBA suitable to solve SJSSP.

The rest of this paper is organized as follows: Section 2 defines the problem, presents a mathematical equation of the SJSSP, and improves a model for evaluating schedules. Section 3 outlines our ESOO-OCBA algorithm for finding a good enough schedule from the search space of SJSSP. Section 4 demonstrates the modifications on OCBA that we made. Section 5 shows and discusses the computational experiments and the results. Finally, in Section 6, we present our concluding remarks and discuss several future research directions.

## 2. SJSSP Formulation and Varying Evaluation Model

*2.1. SJSSP Formulation.* The SJSSP studied in this paper consists of a set of $n$ jobs, $J\ \{J_1, J_2, \ldots, J_n\}$, and a set of $m$ machines, $M = \{M_1, M_1, \ldots, M_m\}$. Every job $J_i$ $(1 \le i \le n)$ consists of $h_i$ operations, $O_i = \{O_{i,1}, O_{i,2}, \ldots, O_{i,h_i}\}$, that need to be processed in sequence. The operation $O_{i,h}$ $(1 \le h \le h_i)$ denotes the $h$th operation on job $i$ and $O_{i,h} \in O_i$ must be processed by a specified machine $M(O_{i,h}) \in M$. Similarly, the set of operations that must be processed on each machine, $M_u$ $(1 \le u \le m)$, is denoted by $OM(M_u)$.

The random processing time of $O_{i,h}$ is denoted by $p_{i,h}$, which is a random variable following a given probability distribution function with mean $\overline{p}_{i,h}$ and variance $\sigma^2_{i,h}$. Let $st_{i,h}$ be the start time of $O_{i,h}$. The completion time and fixed due date of job $J_i \in J$ are denoted by $C_i$ and $d_i$, respectively. $T_i$ and $E_i$ stand for the tardiness and earliness cost of job $J_i$. Set $t_i$ and $e_i$ as the tardiness and earliness penalty per unit time for job $J_i$, respectively.

Let $\Omega$ denote the set of all the feasible and unfeasible schedules; a feasible schedule $S$ should satisfy both precedence constraint and capacity constraint simultaneously. Without loss of generality, it is assumed that all data are integers and no preemption is allowed. The goal of SJSSP is to find a feasible schedule $S \in \Omega$ that minimizes the expected sum of earliness and tardiness costs of all jobs.

Objective function:

$$\text{Minimize}_{\text{cost}} = \min_{S \in \Omega} E\left\{\sum_{i=1}^{n} (T_i + E_i)\right\}. \tag{1}$$

A feasible schedule $S \in \Omega$ should be subject to

$$T_i = t_i \times \max(0, C_i - d_i) \quad i = 1, 2, \ldots, n, \tag{2}$$

$$E_i = e_i \times \max(0, d_i - C_i) \quad i = 1, 2, \ldots, n, \tag{3}$$

$$st_{i,k} + p_{i,k} \le st_{i,k+1} \quad i = 1, 2, \ldots, n \wedge k = 1, 2, \ldots, h_i - 1 \tag{4}$$

for all operations in the set $OM(M_u)$:

$$st_{i,k} + p_{i,k} \le st_{j,h} \vee st_{j,h} + p_{j,h} \le st_{i,k} \quad i, j = 1, 2, \ldots, n \wedge k = 1, 2, \ldots, h_i \wedge h = 1, 2, \ldots, h_j. \tag{5}$$

Constraints (2) and (3) represent the tardiness and earliness cost of each job, respectively. The precedence constraint (4) ensures that, for each pair of consecutive operations $O_{i,k}$ and $O_{i,k+1}$ of the same job $i$, operation $O_{i,k+1}$ cannot be processed before operation $O_{i,k}$ is completed. The capacity constraint (5) ensures that two operations of two different jobs, $J_i$ and $J_j$, $O_{i,k}$ and $O_{j,h}$, in the set $OM(M_u)$ cannot be processed simultaneously on machine $M_u$.

*2.2. Performance Evaluation Model Improved from the Objective Function.* To obtain a good statistical estimate for a feasible schedule, a large number of simulation replications are usually required for each schedule. However, the expected objective value of a feasible schedule is available only in the form of a complex calculation via infinite simulated replications. Although infinite replications of simulation will make the objective value of (1) stable, this calculating method actually is intractable.

Therefore, depending on the amount of simulated replications, (1) can be approximated as follows:

$$\min_{S\in\Omega} F_L(S) = \frac{1}{L}\sum_{l=1}^{L}\left\{\sum_{i=1}^{n}\left[t_i T_i^l(S) + e_i E_i^l(S)\right]\right\} \quad (6)$$

for a feasible schedule $S$; $L$ represents the number of its simulation replications. $t_i T_i^l(S)$ and $e_i E_i^l(S)$ denote the tardiness and earliness costs of job $J_i$ on the $l$th replication of $S$, respectively. $F_L(S)$ denotes the average sum of tardiness and earliness costs of $S$ when the simulation length is $L$. Sufficiently large $L$ will make the objective value of (6), $F_L(S)$, sufficiently stable. Let $L_s = 10^5$ represent the sufficiently large $L$ [8]. Let $F_{L_s}(S)$ represent the objective value of $S$ computed by sufficiently exact evaluation model.

## 3. ESOO-OCBA Algorithm for SJSSP

*3.1. Embedding the OCBA Technique into ESOO.* To evaluate the performance of a feasible schedule $S$ reliably needs a complex calculation of $F_{L_s}(S)$, not to even mention the huge search space of SJSSP. ESOO could significantly reduce the computation for evaluation process [8]. However, in ESOO, uniform computation is allocated to each individual, regardless of whether or not it should. This allocation cannot meet the different demands of different individuals; thus computation allocated to each individual may be either insufficient or redundant.

Ideally, overall simulation efficiency will be improved if less computational effort is spent on simulating noncritical schedules and more is spent on critical schedules. We would like to improve the PCS in each generation by allocating computation according to the performance of each schedule. Therefore, in this paper, OCBA technique is embedded into the exploration stage of ESOO algorithm to intelligently determine the optimal number of simulation times $L$ for different individuals according to their performances.

SJSSP is a NP-hard problem, which reflects the real-world situations and always suffers from uncertain feature. Recently, many methods solving SJSSP suffer from lengthy computation budget, because evaluating the objective of a schedule is already very time-consuming not to even mention the extremely slow convergence of the heuristic techniques in searching through a huge search space. In order to overcome the drawback of consuming much computation time, we propose the ESOO-OCBA algorithm. In our ESOO-OCBA algorithm, OO theory reduces the unbearable computation and OCBA technique allocates necessary computation to each individual. The overall scheme of our algorithm is illustrated by Figure 1.

From the Figure 1, the NP-hard characteristic and the randomness of SJSSP contribute to many difficulties, for example, huge search space, performance evaluation problem, and slow convergence. OO theory contains two fundamental Ideas: (1) *ordinal comparison*, that is, ordinal is used rather than cardinal optimization in order to reduce the simulation times in evaluating schedules; (2) *goal softening* is used to decrease the degree of searching difficulty. It is proved that ordinal comparison has an exponential convergence rate [21, 22] and that goal softening can raise the Probability of Alignment (PA) exponentially [23].

ESOO-OCBA algorithm consists of two stages. (1) *The exploration stage* aims to find a subset of good samples from the search space, where samples are evaluated by a crude evaluation model. Evolutionary strategy is employed in this stage to optimize the sampling process while OCBA is used to trade off the performance stability and save the computation of each individual by allocating the simulation times reasonably. (2) *The exploitation stage* that consists of multiple subphases is used to find out the good enough individual in the good sample subset. Individuals are selected and eliminated by increasingly accurate evaluation models in each subphase. The one with the smallest $F_{L_s}(S)$ in the last subphase, $S^*$, is the good enough schedule that we seek.

*3.2. The Exploration Stage.* In exploration stage, we use ES as a whole frame of sampling process. The number of needed samples depends on the extent of goal softening, which decides the number of generations and the number of initial population. In each generation of ES, each individual could get a unique crude evaluation model developed from (6) with different simulation times $L$ allocated by OCBA. The implementation of exploration stage is as follows [8].

*Precedence-Based Representation.* We use a precedence-based presentation to define a chromosome with unpartitioned permutation of $l_i$ repetitions of each job $J_i$. The operations of the same job should satisfy its precedence constraints to form an individual standing for a feasible schedule. This encoding method is employed due to its effectiveness on generating feasible individuals and on reducing the size of search space.

*Initial Population.* Each individual of the initial population is generated with a completely random method to enhance the variety of the initial population.

*Recombination.* Generate offspring from the parents by discrete recombination [24]. Use repair operator to adjust infeasible schedule when it appears [8].

*Mutation.* The insertion mutation method is adopted to breed new offspring. Delete all operations of one random selected job within a parent chromosome and then reinsert them into the remained components randomly according to the precedence constraints [8].

*Selection.* $\mu + \lambda$-selection mechanism is adopted in our approach. Select the best $\mu$ individuals from both the $\mu$ parents and $\lambda$ offspring, according to the ranking of the approximate fitness values obtained from the respective crude evaluation model of each individual.

*Termination.* The ES is stopped when the number of generations exceeds what we set; select the best $N$ individuals as the good sample set according to their performances.

*3.3. The Exploitation Stage.* In the exploitation stage, we need to find the best schedule from the $N$ schedules in the

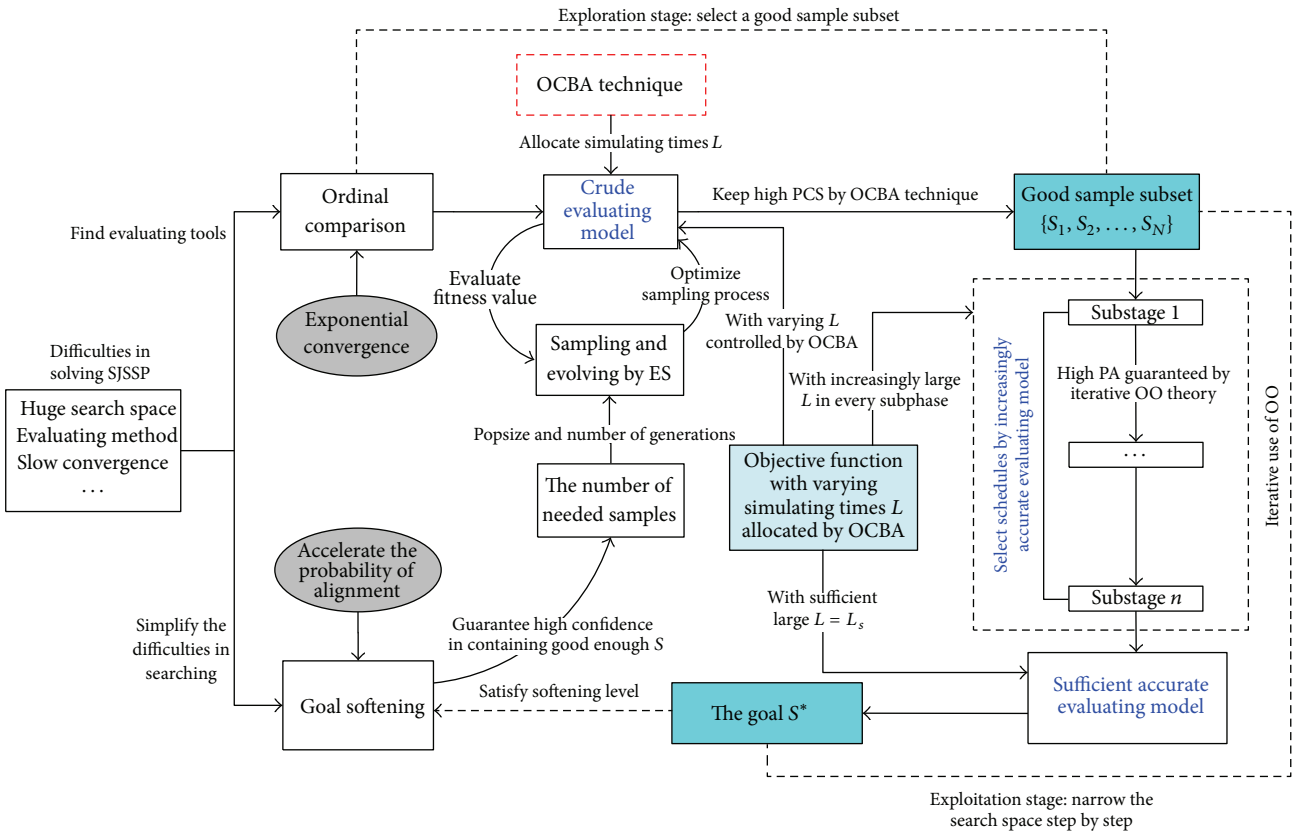


Figure 1: Overall scheme of ESOO-OCBA algorithm for SJSSP.

good sample set obtained in the exploration stage. However, evaluating each of the $N$ schedules by sufficient accurate evaluation model (objective function with $L = L_s$) costs too much computation. Thus the exploitation stage is divided into multiple subphases according to the idea of iterative use of ordinal optimization [25]. The computational complexity of exploitation stage can be dramatically decreased, as the size of the schedule set in each subphase has already been largely reduced when the evaluation model is more refined.

The implementation of exploitation stage in [8] is adopted in this paper. In each subphase, the objective function equation (6) with various simulation times $L$ is used as an increasingly accurate evaluation model, where $L$ ranges from a given fixed $L_g$ (crude model in the first subphase) to $L_s$ (sufficient accurate model in the final subphase). The remained schedules in the prior subphase are selected and some of them are eliminated according to their performances. In the last subphase, the one with the smallest $F_{L_s}(S)$ is the good enough schedule $S^*$ that we look for.

## 4. The OCBA Technique and Modifications

*4.1. The OCBA Technique.* OO theory usually allocates uniform computation to each schedule, which can hardly achieve the highest PCS within a given computation. OCBA technique [17] is employed to improve the PCS by allocating simulation times to each schedule according to their performances. We define $L_i$ as the simulation times allocated to schedule $S_i$ and set $T$ as the total given simulation times in each generation (i.e., total given computation). OCBA provides a recipe of asymptotically optimal allocation of simulation times among schedules in each generation of the exploration stage in ESOO-OCBA algorithm:

$$L_1 + L_2 + \cdots + L_k = T, \tag{7}$$

$$L_b = \sigma_b \sqrt{\sum_{i=1, i\neq b}^{k} \frac{L_i^2}{\sigma_i^2}}, \tag{8}$$

$$\frac{L_i}{L_j} = \left(\frac{\sigma_i/\delta_{b,i}}{\sigma_j/\delta_{b,j}}\right)^2, \quad \text{for } i = 1, 2, \ldots, k,\ i \neq j \neq b, \tag{9}$$

where $\sigma_i^2$ is the observation variance of schedule $S_i$ which can be approximated by sample variance and $S_b$ is the best schedule we observed. $F_{L_i}(S_i)$ is the observed performance of schedule $S_i$; $\delta_{b,i} = F_{L_b}(S_b) - F_{L_i}(S_i)$.

*4.2. Modifications on OCBA Technique.* However, the aforementioned OCBA is observed to be unavailable to solve the specific SJSSP in our experiments. While pursuing a good set of schedules in the exploration stage, "super individuals" may emerge according to (7)–(9), which could absorb most

of the given simulation budget in each generation while other individuals could hardly get any simulation budget.

Without enough simulation times $L$, most of the average individuals in each generation cannot be compared with each other exactly by (6). Therefore, this simulation distortion phenomenon will absolutely decrease the probability of correct selection as some inferior schedules may be selected while some good schedules may be eliminated.

We try to find the root cause of "super individuals" by dating back to the deduction of the original dispatching (7)–(9) [17]. After a series of deductions, Chen et al. [17] got (10) which could express the relationship between $L_b$, $L_i$, and $L_j$. In order to simplify (10), Chen et al. [17] assumed $L_b \gg L_i$ according to (8). Therefore (10) could be simplified as (11), and then the ratio between $L_i$ and $L_j$ was deducted from (11) which was expressed in (9). Consider the following:

$$\begin{aligned}
&\exp\left(\frac{-\delta_{b,i}^2}{2\left(\left(\sigma_b^2/L_b\right)+\left(\sigma_i^2/L_i\right)\right)}\right)\cdot\frac{\delta_{b,i}\sigma_i^2/L_i^2}{\left(\left(\sigma_b^2/L_b\right)+\left(\sigma_i^2/L_i\right)\right)^{3/2}}\\
&\quad=\exp\left(\frac{-\delta_{b,j}^2}{2\left(\left(\sigma_b^2/L_b\right)+\left(\sigma_j^2/L_j\right)\right)}\right)\\
&\qquad\cdot\frac{\delta_{b,j}\sigma_j^2/L_j^2}{\left(\left(\sigma_b^2/L_b\right)+\left(\sigma_j^2/L_j\right)\right)^{3/2}},
\end{aligned}\tag{10}$$

$$\begin{aligned}
&\exp\left(\frac{-\delta_{b,i}^2}{2\left(\sigma_i^2/L_i\right)}\right)\cdot\frac{\delta_{b,i}\sigma_i^2/L_i^2}{\left(\sigma_i^2/L_i\right)^{3/2}}\\
&\quad=\exp\left(\frac{-\delta_{b,j}^2}{2\left(\sigma_j^2/L_j\right)}\right)\cdot\frac{\delta_{b,j}\sigma_j^2/L_j^2}{\left(\sigma_j^2/L_j\right)^{3/2}}.
\end{aligned}\tag{11}$$

However, in our experiments, the performance of individual $S_i$ can be extremely similar to that of the best individual $S_b$; thus $\delta_{b,i} = F_{L_b}(S_b) - F_{L_i}(S_i)$ can be remarkably small. In this condition, according to (9), the simulation times $L_i$ of individual $S_i$ will outnumber considerably that of an average individual $S_j$. Then we can get $L_b \approx L_i$ from (8); that is, the assumption of Chen et al. [17], $L_b \gg L_i$, is not the case in the specific SJSSP environment. As (10) cannot be used directly in dispatching simulation times for its complexity, modifications are made on the original dispatching rule equations (7)–(9) to make the classical dispatching rule available in SJSSP.

In order to diminish the negative influences from "super individuals" in each generation, two steps must be taken: (1) set a threshold of simulation times to detect "super individuals" and (2) follow an exclusion mechanism to isolate them. Deducting from (7)–(9), (12) is used to allocate simulation times $L_i$ to each schedule $S_i$:

$$\begin{aligned}
L_i &= T\times\frac{\sigma_i^2\delta_{b,j}^2}{\sigma_j^2\delta_{b,i}^2}\\
&\quad\times\left[\sum_{i=1,i\neq b,i\neq j}^{k}\frac{\sigma_i^2\delta_{b,j}^2}{\sigma_j^2\delta_{b,i}^2}\right.\\
&\qquad\left.+\sigma_i\left(\sum_{i=1,i\neq b}^{k}\frac{\sigma_i^2\delta_{b,j}^4}{\sigma_j^4\delta_{b,i}^4}\right)^{1/2}+1\right]^{-1}.
\end{aligned}\tag{12}$$

Modifications are made on (12) to realize the mentioned two Steps. Firstly, we set $L_s = 10^5$ as the threshold of simulation times because it is the sufficient simulation times for evaluation. The individual will be seen as a "super individual" once it obtains more than $L_s$ simulation times. Secondly, we define all the "super individuals" as a set $\Theta$ and the scale of $\Theta$ as num and then pick out $\Theta$ from the set of all the individuals. Then from the total given $T$, deduct the simulation times which are occupied by "super individuals," where we set that each "super individual" occupies $L_s$ simulation times. Lastly, allocate the existing simulation times to the individuals except for "super individuals" according to the dispatching rules. Based on these steps, we improve (12) to (13) as follows:

$$\begin{aligned}
L_i &= \left(T-\text{num}\times L_s\right)\\
&\quad\times\frac{\sigma_i^2\delta_{b,j}^2}{\sigma_j^2\delta_{b,i}^2}\times\left[\sum_{i=1,i\neq b,i\neq j,i\notin\Theta}^{k}\frac{\sigma_i^2\delta_{b,j}^2}{\sigma_j^2\delta_{b,i}^2}\right.\\
&\qquad\left.+\sigma_i\left(\sum_{i=1,i\neq b}^{k}\frac{\sigma_i^2\delta_{b,j}^4}{\sigma_j^4\delta_{b,i}^4}\right)^{1/2}+1\right]^{-1}.
\end{aligned}\tag{13}$$

Equation (13) is one of the main contributions of our study. It can be used (i) to allocate computation in the simulation optimization of SJSSP reasonably and effectively or (ii) to make simple estimates for designing experiments in solving SJSSP.

*4.3. Implementation of the Modified OCBA.* We adopt the cost-effective sequential approach based on OCBA which is described as follows [17]: (1) $n_0$ simulation replications for each individual are conducted to get some initial information about the performance of each individual. As simulation proceeds, the sample means and sample variances of each schedule are computed from all the data that are already collected up. (2) According to this collected simulation output, an incremental computing budget, $\Delta$, is allocated to the set of all the individuals. Ideally, each new replication should bring us closer to the optimal schedules. (3) This procedure is continued until the total given $T$ is exhausted

and then (13) can be improved as follows ($T_a$ denotes the simulation time that has already been consumed):

$$L_i = \left(T_a + \Delta - \text{num} \times L_s\right) \times \frac{\sigma_i^2 \delta_{b,j}^2}{\sigma_j^2 \delta_{b,i}^2} \times \left[ \sum_{i=1, i \neq b, i \neq j, i \notin \Theta}^{k} \frac{\sigma_i^2 \delta_{b,j}^2}{\sigma_j^2 \delta_{b,i}^2} + \sigma_i \left( \sum_{i=1, i \neq b}^{k} \frac{\sigma_i^2 \delta_{b,j}^4}{\sigma_j^4 \delta_{b,i}^4} \right)^{1/2} + 1 \right]^{-1}. \quad (14)$$

The implementation of Optimal Computing Budget Allocation (OCBA) in each generation in the exploration stage of ESOO.

*Step 1.* Perform $l$th simulation replications for all individuals; $g = 0$; $L_1^g = L_2^g = \cdots = L_k^g = n_0$.

*Step 2.* If $\sum_{i=1}^{k} L_i^g \geq T$, stop.

*Step 3.* Increase the computing budget (i.e., number of additional simulation times by $L$ and compute the new budget allocation, $L_1^{g+1}, L_2^{g+1}, \ldots, L_k^{g+1}$, using (14).

*Step 4.* Perform additional $\max(0, L_i^{g+1} - L_i^g)$ simulations for schedule, $i, i = 1, \ldots, k$, $g = g + 1$. Go to Step 2.

In the OCBA steps above, $g$ is the iteration number and $k$ is the number of initial population. What needs to be remarked is the best individual $b$ which may change from iteration to iteration.

## 5. Computational Results and Discussion

*5.1. SJSSP Test Instance with Three Processing Time Distributions.* In order to demonstrate the computational quality and efficiency of our ESOO-OCBA algorithm, numerical experiments on SJSSP comprising 8 jobs on 8 machines [8] have been carried out. $(a_{i,h}, \overline{p}_{i,h}, \sigma_{i,h}^2)$ is given in Table 1, which is used to denote the operating environment of operation $O_{i,h}$. $a_{i,h}$ denotes the processing sequence of $O_{i,h}$, $\overline{p}_{i,h}$ and $\sigma_{i,h}^2$ denote the mean and variance of stochastic processing time $p_{i,h}$, respectively. The due dates $d_i$ of each job $J_i$ are given in Table 2. The tardiness penalty per unit time $t_i$ and the earliness penalty per unit time $e_i$ for each job $i$ are set to 1.

Three distributions of random processing time on the machines are used to test the computational efficiency and the obtained schedules quality of our algorithm. The first distribution is truncated normal distribution with mean $\overline{p}_{i,h}$ and variance $\sigma_{i,h}^2$. The second distribution is uniform distribution in the interval $[\overline{p}_{i,h} - 3 \times \sigma_{i,h}, \overline{p}_{i,h} + 3 \times \sigma_{i,h}]$. The third distribution is exponential distribution with mean $\overline{p}_{i,h}$.

In the exploration stage of ESOO-OCBA, we set the number of initial population as $\mu = 1000$ and the number of offspring as $\lambda = 2000$. In order to compare the efficiency between ESOO-OCBA and ESOO [8], the same total given simulation times in each generation of the exploration stage are set as $T = 368 \times \lambda$, and the number of generations is set as $k_{\max} = 100$. It is well understood that a small initial simulation time of each individual, $n_0$, can contribute to more flexibility for better allocation of the computing budget. Nevertheless, if $n_0$ is too large, we may waste too much computation budget in simulating nonpromising designs. Intuitively, if the total computing budget, $T$, is very large, the effect of $n_0$ should be less important. In the dispatching process of OCBA technique in each generation, we know that $T$ is very large and so we set $n_0 = 33$.

In addition, the selection of incremental computing budget, $\Delta$, is typically problem-specific. A large $\Delta$ can lead to waste of computation time to obtain an unnecessarily high confidence level. Nevertheless, if $\Delta$ is too small, we need to carry out the budget allocation problem many times. So according to the total computing budget, $T$, we set $\Delta = 66000$. $L_s = 10^5$ is set as the threshold for detecting "super individual." We start from randomly generating $\mu$ individuals as the parent population. After an evolution of $k_{\max}$ generations, we rank all the remained $\mu + \lambda = 3000$ individuals (parents and offspring) based on their performances and select the best $N = 1000$ individuals.

In the exploitation stage of ESOO-OCBA, we adopt all the parameters used in related works [8]. Table 3 shows the number of subphases, the simulation length, and the number of candidate schedules in each subphase. In the last subphase, we compute the exact object value $F_{L_s}(S)$ of the $N_6 = 7$ candidate schedules. The one with the smallest $F_{L_s}(S)$ is the good enough schedule $S^*$ that we look for.

*5.2. Test Results of Modified OCBA.* In order to show the advantages of our modifications on OCBA, we choose a random generation from the exploration stage of ESOO-OCBA in solving SJSSP with truncated normal distributed processing time. We compare our modified OCBA with classical OCBA [17] by allocating simulation times to all individuals ($\lambda = 2000$), respectively. The test results are shown in Figures 2, 3, and 4.

Figure 2 shows the general results of the realized allocation by the two analyzed OCBA techniques considering the different individuals, allocated simulation times, and performances as comparison criteria. Figure 3 describes the relation between the simulation times and the individual performances in detail and also helps to understand the differences between classical OCBA and modified OCBA. Figure 4 illustrates the distribution of simulation times of each individual, demonstrating the improvements of modified OCBA compared with traditional OCBA in dispatching simulation times.

In Figure 3, we can see that, with the individual performance (objective value) rising, the simulation times allocated to the better individuals increase under the influence of the allocation mechanisms in both classical and modified OCBA. This increment meets the demand of more simulation times when better individuals need to be evaluated exactly in the evaluation process.

Also, for most of the individuals, more simulation times are allocated by our modified OCBA than by classical OCBA. The reason of this phenomenon lies in the mentioned "super

Table 1: Operation environment vector of SJSSP with 8 jobs and 8 machines.

| | $M_1$ | $M_2$ | $M_3$ | $M_4$ | $M_5$ | $M_6$ | $M_7$ | $M_8$ |
|---|---|---|---|---|---|---|---|---|
| $J_1$ | 3,70,140 | 2,80,160 | 1,90,180 | 6,50,100 | 4,40,80 | 8,60,120 | 5,70,140 | 7,50,100 |
| $J_2$ | 1,80,160 | 2,40,80 | 3,50,100 | 5,90,180 | 4,40,80 | 7,50,100 | 6,60,120 | 8,40,80 |
| $J_3$ | 1,50,100 | 2,40,80 | 3,80,160 | 5,60,120 | 4,70,140 | 6,40,80 | 8,40,80 | 7,70,140 |
| $J_4$ | 2,60,120 | 1,50,100 | 3,60,120 | 4,70,140 | 7,80,160 | 5,40,80 | 6,50,100 | 8,80,160 |
| $J_5$ | 4,50,100 | 3,50,100 | 2,70,140 | 1,40,80 | 7,50,100 | 5,60,120 | 6,90,180 | 8,60,120 |
| $J_6$ | 2,60,120 | 3,80,160 | 1,90,180 | 5,70,140 | 6,50,100 | 4,40,80 | 8,80,160 | 7,90,180 |
| $J_7$ | 1,40,80 | 3,60,120 | 4,40,80 | 2,80,160 | 5,60,120 | 7,70,140 | 8,50,100 | 6,60,120 |
| $J_8$ | 2,90,180 | 1,70,140 | 3,50,100 | 4,60,120 | 5,90,180 | 7,80,160 | 6,40,80 | 8,40,80 |

Table 2: Due dates for each job.

| $J_i$ | $J_1$ | $J_2$ | $J_3$ | $J_4$ | $J_5$ | $J_6$ | $J_7$ | $J_8$ |
|---|---|---|---|---|---|---|---|---|
| $d_i$ | 490 | 510 | 540 | 500 | 540 | 470 | 530 | 560 |

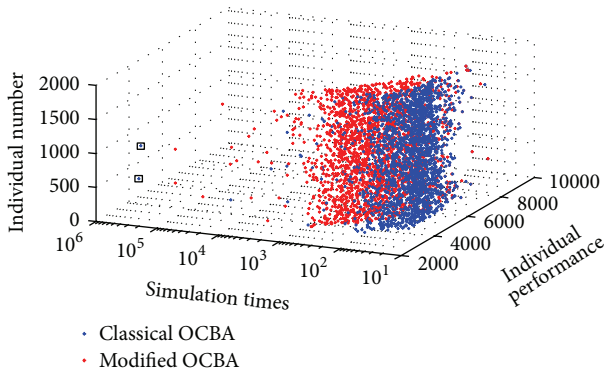


Figure 2: 3D view of the comparison between modified OCBA and classical OCBA.

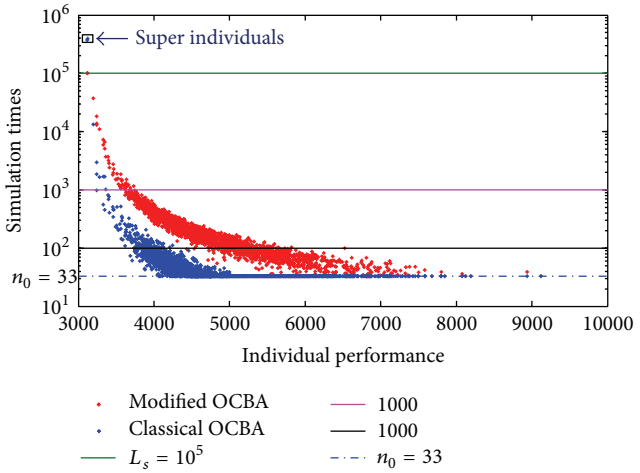


Figure 3: Relationship between individual performance and allocated simulation times.

individuals" produced by the classical OCBA dispatching rules (see (7)–(9)). Besides, for the total of individuals ($\lambda = 2000$) in the generation, the only two labeled "super individuals" absorb 85.75% of the total simulation times,

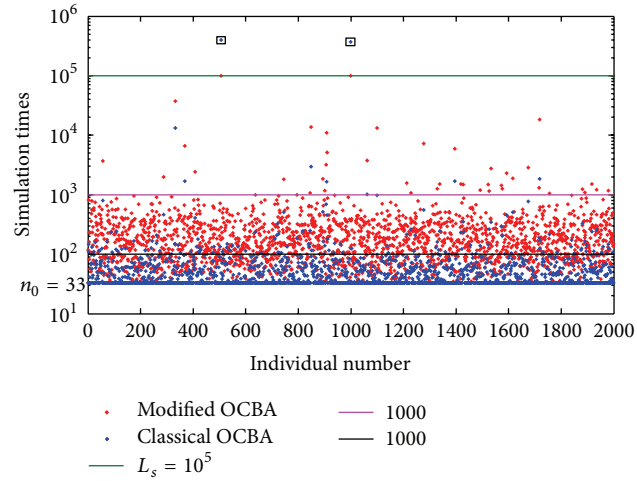


Figure 4: Distribution of allocated simulation times for individuals.

while all the other individuals are allocated with low share (14.25%) of total simulation times.

In Figure 4, a more detailed representation of the distribution of simulation times allocated to individuals can be observed: in the test results of classical OCBA, the simulation times allocated to different individuals have no apparent alterations, even if they have a huge performance gap. It is also clear that 92.85% of all the individuals (1857 of the total 2000) are allocated between 33 and 100 simulation times by classical OCBA. This lack of simulation times leads to a simulation distortion in the evaluation process, not being able to detect the variations in different individual performances, which absolutely decreases the PCS in this generation.

For modified OCBA, 97.85% of all the individuals (1957 of the total 2000) are allocated from 33 to 1000 simulation times. After we use the threshold $L_s = 10^5$ to limit the "super individuals," a larger portion of the total simulation times are allocated to average individuals (74.04%). This situation leads to high performed individuals getting much more simulation times even if they are slightly better (this effect can be observed in Figure 3). This improved allocation contributes

Table 3: Number of candidate schedules and simulation times in each subphase.

| Subphase $i$ | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| $N_i$ | 1000 | 368 | 135 | 50 | 18 | 7 |
| $L_i$ | 1000 | 2718 | 7389 | 20085 | 54598 | 100000 |

Table 4: The good enough schedule $S^*$, corresponding $F_{L_s}(S^*)$ obtained by ESOO-OCBA, and the percentage improvements relative to ESOO.

| Distribution | The good enough schedule $S^*$ | ESOO-OCBA | ESOO [8] | % improvement | CPU-T (s) |
|---|---|---|---|---|---|
| Truncated normal | $O_{4,1}$ $O_{7,1}$ $O_{2,1}$ $O_{1,1}$ $O_{1,2}$ $O_{2,2}$ $O_{5,1}$ $O_{4,2}$ $O_{7,2}$ $O_{1,3}$ $O_{3,1}$ $O_{4,3}$ $O_{2,3}$ $O_{7,3}$ $O_{7,4}$ $O_{7,5}$ $O_{7,6}$ $O_{8,1}$ $O_{2,4}$ $O_{4,4}$ $O_{5,2}$ $O_{1,4}$ $O_{1,5}$ $O_{1,6}$ $O_{7,7}$ $O_{4,5}$ $O_{3,2}$ $O_{2,5}$ $O_{3,3}$ $O_{4,6}$ $O_{7,8}$ $O_{5,3}$ $O_{8,2}$ $O_{8,3}$ $O_{1,7}$ $O_{4,7}$ $O_{5,4}$ $O_{3,4}$ $O_{8,4}$ $O_{1,8}$ $O_{2,6}$ $O_{5,5}$ $O_{6,1}$ $O_{5,6}$ $O_{2,7}$ $O_{4,8}$ $O_{6,2}$ $O_{2,8}$ $O_{8,5}$ $O_{3,5}$ $O_{8,6}$ $O_{3,6}$ $O_{3,7}$ $O_{8,7}$ $O_{5,7}$ $O_{6,3}$ $O_{5,8}$ $O_{8,8}$ $O_{3,8}$ $O_{6,4}$ $O_{6,5}$ $O_{6,6}$ $O_{6,7}$ $O_{6,8}$ | **2089** | 2280 | **8.38** | **392.12** |
| Uniform | $O_{7,1}$ $O_{6,1}$ $O_{4,1}$ $O_{7,2}$ $O_{2,1}$ $O_{7,3}$ $O_{4,2}$ $O_{7,4}$ $O_{5,1}$ $O_{2,2}$ $O_{6,2}$ $O_{4,3}$ $O_{1,1}$ $O_{8,1}$ $O_{4,4}$ $O_{7,5}$ $O_{7,6}$ $O_{6,3}$ $O_{3,1}$ $O_{7,7}$ $O_{6,4}$ $O_{2,3}$ $O_{8,2}$ $O_{4,5}$ $O_{6,5}$ $O_{7,8}$ $O_{2,4}$ $O_{1,2}$ $O_{1,3}$ $O_{3,2}$ $O_{6,6}$ $O_{4,6}$ $O_{5,2}$ $O_{2,5}$ $O_{8,3}$ $O_{8,4}$ $O_{3,3}$ $O_{5,3}$ $O_{6,7}$ $O_{1,4}$ $O_{2,6}$ $O_{6,8}$ $O_{5,4}$ $O_{2,7}$ $O_{1,5}$ $O_{4,7}$ $O_{8,5}$ $O_{8,6}$ $O_{5,5}$ $O_{4,8}$ $O_{3,4}$ $O_{5,6}$ $O_{3,5}$ $O_{2,8}$ $O_{1,6}$ $O_{8,7}$ $O_{8,8}$ $O_{3,6}$ $O_{1,7}$ $O_{1,8}$ $O_{3,7}$ $O_{5,7}$ $O_{5,8}$ $O_{3,8}$ | **2452** | 2778 | **11.74** | **405.22** |
| Exponential | $O_{7,1}$ $O_{4,1}$ $O_{5,1}$ $O_{7,2}$ $O_{3,1}$ $O_{4,2}$ $O_{7,3}$ $O_{3,2}$ $O_{5,2}$ $O_{7,4}$ $O_{7,5}$ $O_{4,3}$ $O_{7,6}$ $O_{3,3}$ $O_{5,3}$ $O_{4,4}$ $O_{2,1}$ $O_{4,5}$ $O_{3,4}$ $O_{4,6}$ $O_{3,5}$ $O_{2,2}$ $O_{5,4}$ $O_{5,5}$ $O_{8,1}$ $O_{1,1}$ $O_{1,2}$ $O_{3,6}$ $O_{8,2}$ $O_{2,3}$ $O_{4,7}$ $O_{5,6}$ $O_{7,7}$ $O_{2,4}$ $O_{3,7}$ $O_{4,8}$ $O_{2,5}$ $O_{6,1}$ $O_{5,7}$ $O_{3,8}$ $O_{7,8}$ $O_{8,3}$ $O_{8,4}$ $O_{5,8}$ $O_{1,3}$ $O_{8,5}$ $O_{6,2}$ $O_{2,6}$ $O_{6,3}$ $O_{1,4}$ $O_{8,6}$ $O_{6,4}$ $O_{2,7}$ $O_{8,7}$ $O_{1,5}$ $O_{2,8}$ $O_{1,6}$ $O_{8,8}$ $O_{1,7}$ $O_{6,5}$ $O_{1,8}$ $O_{6,6}$ $O_{6,7}$ $O_{6,8}$ | **2590** | 2683 | **3.47** | **381.17** |

to a reliable evaluation which guarantees a high PCS in each generation, at the same time reflecting the whole idea of OCBA which essentially is better individuals are allocated with more simulation times.

*5.3. Test Comparisons and Performance Evaluation.* In this section, we show the test results of the proposed ESOO-OCBA algorithm and demonstrate the schedule quality comparing with the ESOO algorithm [8]. The following computational results are conducted in Visual C++ 2010 on a Dual-Core E6600/2 GB RAM/Windows XP.

Because of the random nature of the considered problem we also have repeated the simulation process for 10 simulation runs. We have found that after the 10 simulation runs the result changes a little. Table 4 shows the best objective values, the best schedule performance obtained by ESOO-OCBA and ESOO, respectively. The processing sequence of the best schedules is showed in the table; $O_{i,j}$ denotes the $j$th operation on job $i$. Data from truncated normal distribution, uniform distribution, and exponential distribution are all showed in Table 4. As can be observed, our ESOO-OCBA algorithm outperforms the ESOO algorithm for these three distributions in the quality of the results.

In order to compare our algorithm ESOO-OCBA, we adopt the same total simulation times as ESOO. However, because of the computational burden caused by the embedded OCBA technique, the overall consumed CPU times in our experiments are slightly longer than the consumed by ESOO (within 6 minutes), but still short enough to apply our algorithm in real time.

Also, as we can see from Figure 5, the convergence rate in the exploration stage of ESOO-OCBA is significantly faster than that in ESOO. Here in ESOO-OCBA and ESOO random processing time is obtained from truncated normal distribution. This result demonstrates that introducing OCBA into ESOO algorithm really improves the efficiency of ESOO.

## 6. Conclusion

To cope with the computationally intractable SJSSP, we firstly embed OCBA technique into the exploration stage of ESOO algorithm to further enhance ESOO's efficiency by intelligently allocating simulation times according to individual performance. However, "super individuals," which lead to a simulation distortion in the evaluation process, may emerge according to the classical OCBA. Then we set a threshold to constrain the simulation times allocated to

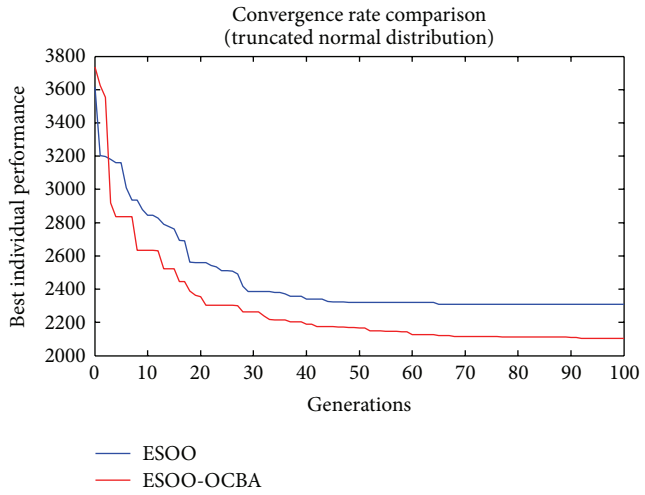


Figure 5: Convergence rate comparison between ESOO-OCBA and ESOO.

"super individuals," by which more simulation times can be allocated to other average individuals. The improvements on classical OCBA optimize the simulation times allocation mechanism, which guarantee a high probability of correct selection in each generation of the evolution in exploration stage.

The proposed algorithm ESOO-OCBA is applied to a SJSSP comprising 8 jobs and 8 machines with random processing time in truncated normal, uniform, and exponential distributions. The simulation test results obtained by ESOO-OCBA are compared with ESOO algorithm, demonstrating that our algorithm has superior performances in the aspect of schedule quality, and our modifications on OCBA are more reasonable in allocating computation in the evaluation.

The future research on SJSSP can be conducted from the following aspects.

(1) It is worthwhile to consider other types of randomness in job shops, for example, rush orders and machine breakdowns.

(2) It is worthwhile to consider a new global computation allocation mechanism (i.e., the breadth versus depth approach [26]) as OCBA technique only allocates the computation within each generation. Ideally, the total computation can be largely reduced by allocating computation globally.

## Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

## Acknowledgments

The authors gratefully appreciate the suggestions and detailed experiment data from Shih-Cheng Horng. This research has been supported by the Graduate Starting Seed Fund of Northwestern Polytechnical University (no. Z2014105), the National Natural Science Foundation, China (no. 50705076), and the Programme of Introducing Talents of Discipline to Universities (B13044).

## References

[1] A. W. J. Kolen, J. K. Lenstra, C. H. Papadimitriou, and F. C. R. Spieksma, "Interval scheduling: a survey," *Naval Research Logistics*, vol. 54, no. 5, pp. 530–543, 2007.

[2] D. Lei, "Interval job shop scheduling problems," *International Journal of Advanced Manufacturing Technology*, vol. 60, no. 1–4, pp. 291–301, 2012.

[3] T. Itoh and H. Ishii, "Fuzzy due-date scheduling problem with fuzzy processing time," *International Transactions in Operational Research*, vol. 6, no. 6, pp. 639–647, 1999.

[4] D. Lei, "A genetic algorithm for flexible job shop scheduling with fuzzy processing time," *International Journal of Production Research*, vol. 48, no. 10, pp. 2995–3013, 2010.

[5] M. Sakawa and R. Kubota, "Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms," *European Journal of Operational Research*, vol. 120, no. 2, pp. 393–407, 2000.

[6] J. Wang, "A fuzzy robust scheduling approach for product development projects," *European Journal of Operational Research*, vol. 152, no. 1, pp. 180–194, 2004.

[7] J. C. Beck and N. Wilson, "Proactive algorithms for job shop scheduling with probabilistic durations," *Journal of Artificial Intelligence Research*, vol. 28, pp. 183–232, 2007.

[8] S.-C. Horng, S.-S. Lin, and F.-Y. Yang, "Evolutionary algorithm for stochastic job shop scheduling with random processing time," *Expert Systems with Applications*, vol. 39, no. 3, pp. 3603–3610, 2012.

[9] A. Azadeh, A. Negahban, and M. Moghaddam, "A hybrid computer simulation-artificial neural network algorithm for optimisation of dispatching rule selection in stochastic job shop scheduling problems," *International Journal of Production Research*, vol. 50, no. 2, pp. 551–566, 2012.

[10] J. Gu, X. Gu, and M. Gu, "A novel parallel quantum genetic algorithm for stochastic job shop scheduling," *Journal of Mathematical Analysis and Applications*, vol. 355, no. 1, pp. 63–81, 2009.

[11] Y. Yoshitomi and R. Yamaguchi, "A genetic algorithm and the Monte Carlo method for stochastic job-shop scheduling," *International Transactions in Operational Research*, vol. 10, no. 6, pp. 577–596, 2003.

[12] R. Zhou, A. Y. C. Nee, and H. P. Lee, "Performance of an ant colony optimisation algorithm in dynamic job shop scheduling problems," *International Journal of Production Research*, vol. 47, no. 11, pp. 2903–2920, 2009.

[13] J. Gu, M. Gu, C. Cao, and X. Gu, "A novel competitive coevolutionary quantum genetic algorithm for stochastic job shop scheduling problem," *Computers & Operations Research*, vol. 37, no. 5, pp. 927–937, 2010.

[14] M. Zandieh and M. A. Adibi, "Dynamic job shop scheduling using variable neighbourhood search," *International Journal of Production Research*, vol. 48, no. 8, pp. 2449–2458, 2010.

[15] R. Zhang and C. Wu, "An artificial bee colony algorithm for the job shop scheduling problem with random processing times," *Entropy*, vol. 13, no. 9, pp. 1708–1729, 2011.

[16] Y. C. Ho, Q. C. Zhao, and Q. S. Jia, *Ordinal Optimization: Soft Optimization for Hard Problems*, Springer, New York, NY, USA, 2007.

[17] C.-H. Chen, J. Lin, E. Yücesan, and S. E. Chick, "Simulation budget allocation for further enhancing the efficiency of ordinal optimization," *Discrete Event Dynamic Systems: Theory and Applications*, vol. 10, no. 3, pp. 251–270, 2000.

[18] C. H. Chen and L. H. Lee, *Stochastic Simulation Optimization: An Optimal Computing Budget Allocation*, World Scientific, River Edge, NJ, USA, 2010.

[19] S. C. Horng, F. Y. Yang, and S. S. Lin, "Apply PSO and OCBA to minimize the overkills and re-probes in wafer probe testing," *IEEE Transactions on Semiconductor Manufacturing*, vol. 25, no. 3, pp. 531–540, 2012.

[20] S. C. Horng, S. Y. Lin, L. H. Lee, and C. H. Chen, "Memetic algorithm for real-time combinatorial stochastic simulation optimization problems with performance analysis," *IEEE Transactions on Cybernetics*, vol. 43, no. 5, pp. 1495–1509, 2013.

[21] L. Dai, "Convergence properties of ordinal comparison in the simulation of discrete event dynamic systems," *Journal of Optimization Theory and Applications*, vol. 91, no. 2, pp. 363–388, 1996.

[22] X. Xie, "Dynamics and convergence rate of ordinal comparison of stochastic discrete-event systems," *IEEE Transactions on Automatic Control*, vol. 42, no. 4, pp. 586–590, 1997.

[23] L. H. Lee, T. W. E. Lau, and Y. C. Ho, "Explanation of goal softening in ordinal optimization," *IEEE Transactions on Automatic Control*, vol. 44, no. 1, pp. 94–99, 1999.

[24] H.-G. Beyer and H.-P. Schwefel, "Evolution strategies—a comprehensive introduction," *Natural Computing*, vol. 1, no. 1, pp. 3–52, 2002.

[25] M. Deng and Y.-C. Ho, "Iterative ordinal optimization and its applications," in *Proceedings of the 36th IEEE Conference on Decision and Control*, pp. 3562–3567, San Diego, Calif, USA, December 1997.

[26] X. Lin and L. H. Lee, "A new approach to discrete stochastic optimization problems," *European Journal of Operational Research*, vol. 172, no. 3, pp. 761–782, 2006.