Cloud theory-based simulated annealing for a single-machine past sequence setup scheduling with scenario-dependent processing times

Chin-Chia Wu¹ · Win-Chin Lin¹ · Xin-Gong Zhang^{2,6} · Dan-Yu Bai^{3,6} · Yung-Wei Tsai¹ · Tao Ren⁴ · Shuenn-Ren Cheng⁵

Received: 17 July 2020 / Accepted: 4 September 2020 / Published online: 15 October 2020
© The Author(s) 2020

Abstract

Recently, the setup times or costs have become an important research topic. The main reason is huge economic savings can be achieved when setup times are explicitly included in scheduling decisions in various real-world industrial environments. On the other hand, many real systems commonly face various uncertainties, such as working environment changes, machine breakages, a worker becomes unstable, etc. In such an environment, job-processing times should not be fixed numbers. Motivated by these situations, this article introduces a single-machine scheduling problem with sequence-dependent setup times and scenario-dependent processing times where the objective function is the total completion time. The robust version of this problem without setup times has been shown to be NP hard. To tackle this problem, a lower bound and a dominance rule are derived in a branch-and-bound method for finding an optimal solution. As for determining approximate solutions, five neighborhood schemes are proposed and embedded in the cloud theory-based simulated annealing. Finally, the performances of all proposed algorithms are determined and compared.

Keywords Machine scheduling · Setup times · Scenario-dependent processing times · Cloud theory-based simulated annealing

Introduction

In most single-machine literature studies, researchers commonly assumed job-processing times or ready times or due dates are fixed numbers, and the setup times are even ignored. However, there are significant uncertainties in many real production systems. For example, machines may break down, job operators may lack ability, the working facility may be changed, i.e., random job arrivals, as well as several other external complex factors can lead to job cancellations or altered tool quality [12]. In light of these facts, job-processing times might not be assumed as fixed numbers. On the other hand, huge economic costs can be saved when setup times are effectively considered in scheduling decisions in various industrial environments [14].

✉ Win-Chin Lin
linwc@fcu.edu.tw
Chin-Chia Wu
cchwu@fcu.edu.tw
Xin-Gong Zhang
zhangxingong@cqnu.edu.cn
Dan-Yu Bai
mikebdi@163.com
Yung-Wei Tsai
viviantai@gmail.com
Tao Ren
chinarentao@163.com
Shuenn-Ren Cheng
k0252@gcloud.csu.edu.tw

¹ Department of Statistics, Feng Chia University, No. 100, Wenhwa Road, Seatwen Dist., Taichung 40724, Taiwan

² College of Mathematics Science, Chongqing Normal University, Chongqing, People's Republic of China

³ College of Management, Dalian Maritime University, Dalian, People's Republic of China

⁴ Software College, Northeastern University, Shenyang 110819, China

⁵ Department of Business Administration, Cheng Shiu University, Kaohsiung, Taiwan

⁶ Key Lab for OCME, School of mathematical Science, Chongqing Normal University, Chongqing 400047, China

In addressing the uncertainty issues observed in the production process, researchers modeled job processing times or due dates or ready times as random parameters, or considered machines can break down randomly. Thus, stochastic scheduling problems have been inspired much attention in the researcher community, in which authors applied some probability distributions to model the uncertain factors and attempted optimality using the average sense idea. For some stochastic single machine or flowshop studies with stochastic sequence-dependent setup times or with learning effect, readers may refer to Li [16], Ertem et al. [6], Urgo and Vancza [23], Lusby et al. [17], Fu et al. [7–9] etc. However, in another stream of addressing uncertainty issues, some authors considered job-processing times as statistical data, in which the data variance might be a bit larger and the underlying distributions drawn from the data cannot be correct. Additionally, the worst-case measurement of a system is more important than the average-case measurement. In addressing this situation, Kouvelis and Yu [15] introduced a robustness method to hedge against the worst case contingency. For more practical applications with the uncertainty expressed as a part of the scenarios, readers may refer to Aloulou and Della Croce [3], Aissi et al. [4], Yang and Yu [29], and Kasperski and Zielinski [12], and so on.

Meanwhile, the literature also showed that setup times are often assumed to be either sequence independent or sequence dependent. In the former case, the setup times are ignored or included in the job-processing times. In the latter case, the setup times are assumed not only to depend on the job currently being scheduled but also the last scheduled job. This study will focus on the latter case. That is, setup times are proportionate to the length of the already scheduled jobs. This is also termed as past-sequence-dependent or p-s-d setup times. The situation occurs in high-tech manufacturing in which a batch of jobs consists of a group of electronic components mounted together on an integrated circuit (IC) board [14]. Following this approach, Wang [24–26] and Cheng et al. [5] addressed some polynomial solutions to solve single-machine scheduling models with past-sequence-dependent setup times and the exponential/or time-dependent learning effect. Wang and Li [27] examined several single-machine p-s-d setup times scheduling problems along with general position-dependent and time-dependent learning effects. Soroush [20] introduced new exact and polynomially solution strategies to minimize the sum of earliness penalties subject to no tardy jobs on a single machine. Soroush [21] addressed a bicriteria single-machine scheduling problem with job-dependent p-s-d setup times and job-dependent position-based learning effects. The objective functions are a linear composite function of a pair of performance criteria such as the makespan, the total completion time, the total lateness, the total absolute differences in completion times, and the sum of earliness, tardiness, and common due

date penalty, etc. Regarding more discussion of variants of scheduling models with p-s-d setup times, readers may refer to a third comprehensive survey paper by Allahverdi [2]. For more recent single-machine literature studies with different settings, readers may refer to Adulyasak et al. [1], Yin et al. [31–35], Xiong et al. [36], Kamran et al. [13], and Gao et al. [10], etc.

Motivated by the many applications of the past-sequence-dependent setup times by Wang [24–26] and Cheng et al. [5] and a semi-automatic assembly example of scenario-dependent processing times by Wu et al. [29], following the robustness idea of Kouvelis and Yu [15] and the p-s-d idea of Koulamas and Kyparisis [14], this study introduces a single-machine scheduling problem with p-s-d setup times and scenario-dependent processing times where the objective function is to minimize the maximum total completion time among scenarios. To solve this problem, we derive one lower bound and one property to be used in a branch-and-bound algorithm for small job cases. Then we propose four new neighborhood search methods to embed in the cloud theoretical simulated annealing [22] to compare with the common one (two-point swapping method) for the small and large job cases. The several contributions of this article are: (1) introduce a single-machine scheduling problem with p-s-d setup times and scenario-dependent processing times except the objective function is the same as that of Yang and Yu [29]. (2) To solve this model, we derive a lower bound and a property to be used in a branch-and-bound algorithm for the optimal solution for a small number of jobs. (3) We propose five variants of the cloud-theoretical simulated annealing (CSA) algorithm, including new neighborhood search methods for a small number of jobs and a large number of jobs.

The rest of this study is described as follows. The next section introduces the problem formulation. The following section presents properties and a lower bound used in the branch-and-bound algorithm. The next section introduces five neighborhoods and proposes five variants of CSA algorithms. The following section tunes the parameters values of the CSA algorithms. The next section tests and reports the performances of all proposed methods. The last section offers a conclusion and provides future suggestions.

Problem description

The study under consideration can be formulated as follows. Consider n jobs to be processed on a single machine and all jobs are available at time zero. No job can be preempted. The machine cannot be allowed to be idle and it can process one job at a time. No job can be interrupted after the start of its processing. Further, economic savings are achieved when setup times are explicitly included in scheduling decisions. On the other hand, many real systems frequently face various

uncertainties such as working environment changes, machine breakages, a worker becomes unstable, etc. Job-processing times cannot be treated as fixed numbers. Thus, following the ideas of both Koulamas and Kyparisis [14] and Yung and Yu [29], we define the scenario-based processing times of job J_j to be scheduled in the k th position and a value that depends on the sum of the processing times of the all already scheduled jobs as $p_{jk}^{(s)} = s_{[k]}^{(s)} + p_j^{(s)}$, for scenario $s = 1$ or 2 on the machine, where $s_{[1]}^{(s)} = 0$, $s_{[k]}^{(s)} = b^{k-1} \sum_{i=1}^{k-1} p_{[i]}^{(s)}$, $k = 1, 2, \dots, n$, $[\cdot]$ represents the scheduled job position, and b , named a learning rate, is a constant number with $0 < b \leq 1$.

Suppose Ω denotes all possible permutations of given n jobs. For a schedule σ , let $t^{(s)}$ be the starting times of a job scheduled the k th position in σ with the initial condition $t^{(s)} = 0$ for any job in the first position. Then, the completion times of J_j at sequence position k in schedule σ are defined as $C_{[k]}^{(s)}(\sigma) = t^{(s)} + s_{[k-1]}^{(s)} + p_{[k]}^{(s)}$. Then, the robust total completion time of σ , is defined as $\text{RpsdTC}(\sigma) = \max_{s=1,2} \sum_{1 \leq k \leq n} C_{[k]}^{(s)}(\sigma)$. In standard machine scheduling terms [18], the proposed problem can be denoted by $1/\text{psd}, p_{jk}^s/\max_{s=1,2} \sum_{1 \leq k \leq n} C_{[k]}^{(s)}(\sigma)$. The goal is to find a sequence σ^* such that $\text{RpsdTC}(\sigma^*) = \min_{\sigma \in \Omega} \{\text{RpsdTC}(\sigma)\}$.

Solution methodology

Since the same problem without p-s-d consideration has been proven to be an NP-hard problem by Yang and Yu [29], this study problem is also NP hard. Thus, we derive a lemma and a lower bound used in a branch-and-bound method for finding the optimal objective. Given two sequences $\sigma_1 = (\delta, i, j, \delta')$ and $\sigma_2 = (\delta, j, i, \delta')$ in which δ and δ' present two partial sequences, we say sequence σ_1 dominates σ_2 if the following conditions hold.

$$\begin{aligned} & \max \{C_i^{(1)}(\sigma_1) + C_j^{(1)}(\sigma_1), C_i^{(2)}(\sigma_1) + C_j^{(2)}(\sigma_1)\} \\ & < \max \{C_j^{(1)}(\sigma_2) + C_i^{(1)}(\sigma_2), C_j^{(2)}(\sigma_2) + C_i^{(2)}(\sigma_2)\}, \\ & C_j^{(1)}(\sigma_1) < C_i^{(1)}(\sigma_2), \text{ and} \\ & C_j^{(2)}(\sigma_1) < C_i^{(2)}(\sigma_2). \end{aligned}$$

The details of Lemma 1 are given as follows:

Lemma 1 Given two jobs J_i and J_j to be scheduled in adjacent positions, there is an optimal sequence that job J_i is scheduled before job J_j if $p_i^{(1)} < p_j^{(1)}$ and $p_i^{(2)} < p_j^{(2)}$ hold.

Proof Suppose $\sigma_1 = (\delta, i, j, \delta')$ and $\sigma_2 = (\delta, j, i, \delta')$, and there are r jobs scheduled in δ , the following will demonstrate σ_1 dominates σ_2 . Let $t^{(s)}$ be the starting times of a job scheduled in the $(r+1)$ th position in σ_1 and σ_2 , and $s_{[k]}^{(s)}$

$(\sigma) = b^{k-1} \sum_{i=1}^{k-1} p_{[i]}^{(s)}$. Note, $s_{[r]}^{(s)}(\sigma_1) = b^{r-1} \sum_{i=1}^{r-1} p_{[i]}^{(s)} = s_{[r]}^{(s)}(\sigma_2)$.

According to the definition of the completion time of a job, we have:

$$\begin{aligned} C_i^{(s)}(\sigma_1) &= t^{(s)} + s_{[r]}^{(s)}(\sigma_1) + p_i^{(s)}, \\ C_j^{(s)}(\sigma_1) &= C_i^{(s)}(\sigma_1) + s_{[r+1]}^{(s)}(\sigma_1) + p_j^{(s)} \\ &= C_i^{(s)}(\sigma_1) + b s_{[r]}^{(s)}(\sigma_1) + b^r p_i^{(s)} + p_j^{(s)}, \\ C_j^{(s)}(\sigma_2) &= t^{(s)} + s_{[r]}^{(s)}(\sigma_2) + p_j^{(s)}, \end{aligned}$$

and

$$\begin{aligned} C_i^{(s)}(\sigma_2) &= C_j^{(s)}(\sigma_2) + s_{[r+1]}^{(s)}(\sigma_2) + p_i^{(s)} \\ &= C_j^{(s)}(\sigma_2) + b s_{[r]}^{(s)}(\sigma_2) + b^r p_j^{(s)} + p_i^{(s)}. \end{aligned}$$

Based on the above, we have

$$C_i^{(s)}(\sigma_2) - C_j^{(s)}(\sigma_1) = b^r (p_j^{(s)} - p_i^{(s)}) > 0, \text{ for } s = 1, 2.$$

In addition,

$$\begin{aligned} C_j^{(s)}(\sigma_2) + C_i^{(s)}(\sigma_2) - C_i^{(s)}(\sigma_1) - C_j^{(s)}(\sigma_1) \\ = (1 + b^r) (p_j^{(s)} - p_i^{(s)}) > 0. \end{aligned}$$

Hence $\max_{s=1,2} \{C_i^{(s)}(\sigma_1) + C_j^{(s)}(\sigma_1)\} < \max_{s=1,2} \{C_j^{(s)}(\sigma_2) + C_i^{(s)}(\sigma_2)\}$, this completes the proof.

Next, a lower bound will be derived to decline the branching trees of the branch-and-bound method. It is noted that r denotes the number of scheduled jobs, n_1 denotes the number of unscheduled jobs, where $n = r + n_1$, and $p_{(r+1)}^{(s)} \leq p_{(r+2)}^{(s)} \leq \dots \leq p_{(r+n_1)}^{(s)}$ denotes a non-decreasing order of $\{p_{r+1}^{(s)}, p_{r+2}^{(s)}, \dots, p_{r+n_1}^{(s)}\}$ in the unscheduled part, $s = 1, 2$. By applying the same approach of Soroush [20, 21], Wang and Li [27], and Yang and Yu [29], a lower bound can be derived as follows:

For the completion times for the jobs scheduled on the subsequently $(r+1), \dots, (r+n_1)$, we have:

$$\begin{aligned} C_{[r+1]}^{(s)} &= t^{(s)} + b^r \sum_{k=1}^r p_{[k]}^{(s)} + p_{[r+1]}^{(s)} \geq t^{(s)} + b^r \sum_{k=1}^r p_{[k]}^{(s)} + p_{(r+1)}^{(s)} \\ C_{[r+2]}^{(s)} &= C_{[r+1]}^{(s)} + b^{r+1} \left[\sum_{k=1}^r p_{[k]}^{(s)} + p_{[k+1]}^{(s)} \right] + p_{[k+2]}^{(s)} \\ &\geq t^{(s)} + (b^r + b^{r+1}) \sum_{k=1}^r p_{[k]}^{(s)} + (1 + b^{r+1}) p_{(k+1)}^{(s)} + p_{(k+2)}^{(s)} \\ &\dots \end{aligned}$$

$$C_{[r+n_1]}^{(s)} \geq t^{(s)} + \left(\sum_{i=0}^{n_1-1} b^{r+i} \right) \sum_{k=1}^r p_{[k]}^{(s)} + \left(1 + \sum_{i=0}^{n_1-2} b^{r+i} \right) p_{(r+1)}^{(s)}$$

$$+ \dots + (1 + b^{r+n_1-1})p_{(r+n_1-1)}^{(s)} + p_{(r+n_1)}^{(s)}.$$

According to the above developments, we have

$$\begin{aligned} \sum_{k=1}^n C_{[k]}^{(s)} &\geq n_1 t^{(s)} + \left(\sum_{i=1}^{n_1} (n_1 - i + 1) b^{r+i-1} \right) \sum_{k=1}^r p_{[k]}^{(s)} \\ &\quad + \left(n_1 + \sum_{i=2}^{n_1} (n - i + 1) b^{r+i-1} \right) p_{(r+1)}^{(s)} \\ &\quad + \left(n_1 - 1 + \sum_{i=3}^{n_1} (n - i + 1) b^{r+i-1} \right) p_{(r+2)}^{(s)} \\ &\quad + \dots + (1 + b^{r+n_1-1}) p_{(r+n_1-1)}^{(s)} + p_{(r+n_1)}^{(s)}, \quad s = 1, 2. \end{aligned}$$

Therefore, we have:

$$\begin{aligned} \max_{s=1,2} \left\{ \sum_{k=1}^n C_{[k]}^{(s)} \right\} &\geq \frac{1}{2} \{ n_1 t^{(s)} + \left(\sum_{i=1}^{n_1} (n_1 - i + 1) b^{r+i-1} \right) \\ &\quad \times \sum_{k=1}^r p_{[k]}^{(s)} + \left(n_1 + \sum_{i=2}^{n_1} (n - i + 1) b^{r+i-1} \right) p_{(r+1)}^{(s)} \\ &\quad + \left(n_1 - 1 + \sum_{i=3}^{n_1} (n - i + 1) b^{r+i-1} \right) p_{(r+2)}^{(s)} \\ &\quad + \dots + (1 + b^{r+n_1-1}) p_{(r+n_1-1)}^{(s)} + p_{(r+n_1)}^{(s)} \}. \end{aligned}$$

Following the lemma of Hardy et al. [11], the above inequality can be confirmed by applying to match the elements of the vectors $\{p_{(r+1)}^{(s)}, \dots, p_{(r+n_1-1)}^{(s)}, p_{(r+n_1)}^{(s)}\}$, in the opposite order of the elements of their corresponding weight vectors $\{(n_1 + \sum_{i=2}^{n_1} (n - i + 1) b^{r+i-1}), \dots, (1 + b^{r+n_1-1}), 1\}$. Thus, the proposed lower bound (lbdd) can be yielded as follows:

$$\begin{aligned} \text{lbdd} &= \frac{1}{2} \{ n_1 t^{(s)} + \left(\sum_{i=1}^{n_1} (n_1 - i + 1) b^{r+i-1} \right) \sum_{k=1}^r p_{[k]}^{(s)} \\ &\quad + \left(n_1 + \sum_{i=2}^{n_1} (n - i + 1) b^{r+i-1} \right) p_{(r+1)}^{(s)} \\ &\quad + \left(n_1 - 1 + \sum_{i=3}^{n_1} (n - i + 1) b^{r+i-1} \right) p_{(r+2)}^{(s)} \\ &\quad + \dots + (1 + b^{r+n_1-1}) p_{(r+n_1-1)}^{(s)} + p_{(r+n_1)}^{(s)} \}. \end{aligned}$$

Five neighborhood search methods and a cloud theory-based simulated annealing

To save on cost or computational time, we provide a cloud theory-based simulated annealing (CSA) algorithm [22] searching for near-optimal solutions for the study problem.

As for other variants of the simulated-annealing algorithm, readers might refer to Sayed and Hassanien [19]. There are four common parameters including the initial temperature (T_i), the final temperature (T_f), the cooling factor (λ), and the number of improvement repetitions (N_r) when running the CSA algorithm. The first neighborhood search method, named NS1, is to switch two randomly selected jobs, J_i, J_j , where $1 \leq i < j \leq n$, up to N_r times for each level of temperature. To enlarge the diversity of neighborhood search, we propose four more neighborhood search methods (named NS2, NS3, NS4, and NS5) decorated in the CSA algorithm. These five neighborhood search methods are described as follows:

NS1: switch two randomly selected jobs, J_i, J_j , where $1 \leq i < j \leq n$.

NS2: randomly select two jobs, J_i, J_j , where $1 \leq i < j < n$, and swap both jobs with their immediately succeeding jobs.

NS3: randomly select two jobs, J_i, J_j , where $1 < i < j \leq n$, and swap both jobs with their closest preceding jobs.

NS4: randomly select two jobs, J_i, J_j , where $1 \leq i < i + 1 < j \leq n$, swap the job in front with its immediately succeeding job and swap another with its closest preceding job.

NS5: randomly select two jobs, J_i, J_j , where $1 < i < j < n$, swap the job in front with its closest preceding job and swap another with its immediately succeeding job.

Figure 1 illustrates these five neighborhood search methods for $n = 8$. For each method, the first row is an incumbent schedule of job and the circled jobs are two selected jobs by the corresponding search method. The second row is a schedule after applying the search method on the incumbent schedule (or on the circled jobs).

These five neighborhood search methods are, respectively, decorated in the CSA algorithms. They are termed CSA1, CSA2, CSA3, CSA4, and CSA5. As for the initial seed used in the five variants of the CSA algorithm, we adopt the average of two scenarios of the process-

NS1:	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
NS2:	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_8
NS3:	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
NS4:	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
NS5:	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8
	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8

Fig. 1 Illustration of five neighborhood search methods

ing time $p_j^{(s)}$, $s = 1, 2$, i.e., $(p_j^{(1)} + p_j^{(2)})/2$, and then sort the sequence in a non-decreasing order of $(p_j^{(1)} + p_j^{(2)})/2$ of n jobs as the common initial seed in five CSAs. Recall the values of objective function for two sequences of σ and σ_t are $RpsdTC(\sigma) = \max_{s=1,2} \sum_{1 \leq k \leq n} C_{[k]}^{(s)}(\sigma)$ and $RpsdTC(\sigma_t) = \max_{s=1,2} \sum_{1 \leq k \leq n} C_{[k]}^{(s)}(\sigma_t)$. Let $d = (RpsdTC(\sigma_t) - RpsdTC(\sigma))/RpsdTC(\sigma)$. The procedures of the proposed CSAs are described as follows:

The procedures of CSA:

- 01: **Input** the values of T_i, T_f, λ , and N_r ;
- 02: **Input** the initial sequence σ_0 ;
- 03: **Input** $\sigma = \sigma_0, T = T_i$;
- 04: **Set** $H_e = T, E_n = T, u_0 = 1 - T, E'_n = \max \left\{ E_n + H_e - \frac{1}{3} q, \delta \right\}, 0 < q_1 < 1, 0 < \delta \leq 10^{-8}$;
- 05: **Set** $T' = E'_n \cdot \sqrt{-2 \ln(u_0)}$;
- 06: **Do while** $\{T > T_f\}$;
- 07: $j = 0$;
- 08: **Do while** $\{j \leq N_r\}$;
- 09: Perform a neighborhood method *NSI* to generate a new sequence σ_t ;
- 10: If $RpsdTC(\sigma_t) < RpsdTC(\sigma)$, update σ with σ_t ;
- 11: Otherwise, update σ with σ_t if $\exp(-d / T') < q_2$;
- 12: q_2 is a random number with $0 < q_2 < 1$;
- 13: $j = j + 1$;
- 14: **End while**;
- 15: $T = T_i \times \lambda^j$;
- 16: **End while**;
- 17: **Output** the final sequence σ for each CSA1.

mined parameters of CSA1 will be adopted in the other four CSAs. The results are summarized in Fig. 2a–c.

To tune the first parameter T_i , we fixed $\lambda = 0.98, N_r = 5$, and $T_f = 10^{-8}$. Then we test the values of T_i from 0.00005 to 0.9999 by an increment of 0.00005 per time. Figure 2a shows the *AEP* was approaching at a relatively lower point as T_i at 0.5. Thereby, the T_i was adopted at 0.5 in the CSAs.

For tuning the second parameter λ , we fixed $T_i = 0.5, N_r = 5$, and $T_f = 10^{-8}$, the examined value of λ from 0.05 to

Tuning the parameters values in CSAs

This section presents 100 generated instances by randomness in a 1 factor at a time way to calibrate the parameters of the proposed CSAs. The parameters to be tuned are the initial temperature (T_i), cooling rate (λ), and the number of improvements (N_r) for each level temperature. During the tuning procedure, we set the number of jobs at $n = 10$ and the final temperature at $T_f = 10^{-8}$. We generate the processing times of $p_j^{(1)}$ from a uniform distribution $U(1, 50)$ for scenario one ($s = 1$) and the processing times of $p_j^{(2)}$ from another uniform distribution $U(1, 100)$ for scenario two ($s = 2$), respectively. The control number of the learning rate b is set at 0.5 for each trial. The measurement adopts the average error percentage (AEP), i.e., $AEP = [(C - O^*)/O^*] \times 100$ [%], where C is the total completion time of the job found by CSA1 and O^* is the total completion time of the jobs by the branch and bound method. For simplification, all the deter-

0.95 by an increment of 0.05 per trial. As the AEPs gradually declined with λ growing, we extend the λ value to 0.99 and examine the value of λ from 0.90 to 0.99 by an increment of 0.05 per trial. Figure 2b shows the AEPs gradually declined as the λ increased and approached a lower point at $\lambda = 0.95$. Thus, a value of $\lambda = 0.99$ was adopted for the later tested experiments.

As for the last parameter (N_r), we fixed $T_i = 0.5, \lambda = 0.99$, and $T_f = 10^{-8}$, the tested range of N_r from 5 to 30 by an increment of one per time. Figure 2c shows there is a relative lowest point at $N_r = 26$. Intentionally, the N_r was set at 26 for the later simulation tests.

In light of the above developments, the parameters ($T_i = 0.5, \lambda = 0.99, N_r = 26$, and $T_f = 10^{-8}$) were finally set in the following simulation tests for all five CSA algorithms.

Fig. 2 Tuning parameters of CSA algorithms

Computational simulation experiments and analysis

This section presents two computational experiment sets to evaluate the performances of the proposed branch-and-bound method and five CSA algorithms (CSA1, CSA2, ..., and CSA5). All programs were coded in Compaq Visual Fortran version 6.6 and tested on a PC with a 2.66-GHz Intel(R) Core(TM) i4 Duo E8200 CPU and 1.99 GB RAM in Windows XP. The test instances were generated based on the combinations of the types of processing times and values of learning rate (b). Three types of processing time are described as follows:

Type 1: we consider the case where the range of variation of the processing times for scenario 1 was less than that of the processing times for scenario 2. That is, the processing times of the jobs were generated from the uniform distribution $U(1, 100)$ for scenario 1 and $U(1, 50)$ for scenario 2, respectively.

Type 2: we consider the case where the range of variation of the processing times for scenario 1 was greater than that of the processing times for scenario 2. That is, the processing times of the jobs were generated from the uniform distribution $U(1, 50)$ for scenario 1 and $U(1, 100)$ for scenario 2, respectively.

Type 3: we consider the case where the range of variation of the processing times for scenario 1 was equal to that of

the processing times for scenario 2. That is, the processing times of the jobs were generated from the uniform distribution $U(1, 100)$ for scenario 1 and $U(1, 100)$ for scenario 2, respectively.

Three different learning rate values (b) were set at 0.25, 0.5, and 0.75. For the set of small number of jobs, it was set job size at $n = 8, 10$ and 12. A set of 100 instances were randomly generated for each case. Consequently, 2700 ($n \times \text{type} \times b \times 100 = 3 \times 3 \times 3 \times 100$) problem instances were tested. The branch-and-bound algorithm was set to skip to the next set of data if the number of nodes exceeded 10^8 .

The set of experiments with a large number of jobs was the same as the small number of jobs, except the n was set at 80, 100, and 120. A set of 100 instances were randomly generated for each case. Consequently, 2700 problem instances were tested as well.

Small- n simulation results analysis

The criterion AEP (average error percentage) of test instances was employed to evaluate the ability of the five CSAs (CSA1, CSA2, CSA3, CSA4, CSA5) searching near-optimal solutions. The AEP was defined as $100[(C_k - O_k)/O_k][\%]$, where C_k was acquired from each CSA algorithm and O_k was acquired from the branch-and-bound method.

As to the implementation of the branch-and-bound method, Table 1 displays its capability. The average CPU

Table 1 Performance of the branch-and-bound method

BB						
n	Processing time	b	Node		CPU time	
			Mean	Max	Mean	Max
8	Type 1	0.25	5572	17,047	0.04	0.09
		0.5	5935	16,297	0.04	0.09
		0.75	5460	14,538	0.04	0.08
	Type 2	0.25	6466	18,223	0.04	0.11
		0.5	6076	18,017	0.04	0.09
		0.75	6847	16,597	0.04	0.09
	Type 3	0.25	5582	24,365	0.04	0.09
		0.5	6414	20,230	0.04	0.09
		0.75	6795	26,579	0.04	0.11
Mean			6127	19,099	0.04	0.10
10	Type 1	0.25	247,777	1,147,921	0.87	4.20
		0.5	262,225	1,208,042	0.92	4.45
		0.75	234,911	988,684	0.82	3.62
	Type 2	0.25	266,802	1,202,267	0.94	4.45
		0.5	239,831	970,825	0.84	3.54
		0.75	303,151	1,321,254	1.07	4.79
	Type 3	0.25	233,325	1,023,892	0.81	3.67
		0.5	265,799	1,067,328	0.93	3.88
		0.75	278,161	805,054	0.98	2.92
Mean			259,109	1,081,696	0.91	3.95
12	Type 1	0.25	16,196,948	72,813,224	106.53	505.75
		0.5	17,603,444	91,498,880	117.57	689.26
		0.75	16,001,638	77,719,040	106.78	564.11
	Type 2	0.25	16,193,736	73,246,432	83.89	421.61
		0.5	15,708,485	80,686,760	81.36	440.72
		0.75	20,333,850	89,962,840	102.67	468.44
	Type 3	0.25	15,284,177	97,307,896	80.82	551.42
		0.5	15,701,142	93,409,024	106.19	677.17
		0.75	18,114,940	85,246,752	117.36	624.86
Mean			16,793,151	84,654,538	100.35	549.26
Total mean			5,686,129	28,585,111	33.77	184.433

execution times (in seconds), presented in columns 6 and 7 of Table 1, clearly increase the number of jobs n added. The mean of nodes was also increasing as n was growing, displayed in column 4 of Table 1. In addition, most of the generated test instances (2696 out of 2700) were solvable within 10^8 nodes, except four problem instances as $n = 12$.

Table 2 illustrates the summaries of the node and CPU time for the factors n , type, and learning rate (b), respectively. As the number of jobs (n) increased, the numbers of mean nodes and mean CPU times grew. This meets a characteristic of NP-hard problems.

Concerning the performances of the proposed five CSA algorithms, Table 3 records the AEPs. The AEP of CSA1 increased slightly as n increased, on the contrary, the AEPs

Table 2 Summary of the performance of branch-and-bound method

		Node		CPU time (s)	
		Mean	Max	Mean	Max
n	8	6127	19,099	0.04	0.10
	10	259,109	1,081,696	0.91	3.95
	12	16,793,151	84,654,538	100.35	549.26
Type	2	5,618,212	27,269,397	30.44	165.71
	3	5,620,068	27,323,600	34.56	187.53
	4	5,407,044	26,096,087	30.53	159.82
b	0.25	5,382,265	27,422,363	30.44	165.71
	0.5	5,533,261	29,877,267	34.21	202.14
	0.75	6,142,861	28,455,704	36.64	185.45

Table 3 Performance of CSA algorithms for $n = 8, 10, 12$

n	Processing time	b	CSA1		CSA2		CSA3		CSA4		CSA5		FS
			Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max	
8	Type 1	0.25	0.00	0.00	0.14	1.61	0.11	1.54	0.12	1.57	0.13	1.61	100
		0.5	0.04	2.85	0.18	2.21	0.16	2.21	0.17	2.21	0.18	2.21	100
		0.75	0.03	1.23	0.16	2.49	0.17	1.65	0.15	1.65	0.16	1.65	100
	Type 2	0.25	0.03	1.40	0.09	1.17	0.09	1.17	0.10	1.91	0.07	0.71	100
		0.5	0.01	0.34	0.16	1.35	0.12	1.36	0.14	1.35	0.13	1.35	100
		0.75	0.06	2.00	0.17	1.46	0.16	1.70	0.14	1.16	0.20	2.83	100
	Type 3	0.25	0.06	1.83	0.18	2.09	0.14	0.96	0.12	1.61	0.14	1.13	100
		0.5	0.10	1.77	0.20	2.61	0.19	1.93	0.23	2.33	0.23	2.39	100
		0.75	0.10	2.50	0.25	2.50	0.31	2.65	0.26	3.66	0.21	2.18	100
	Mean		0.05		0.17		0.16		0.16		0.16		
10	Type 1	0.25	0.01	0.49	0.04	0.40	0.04	0.40	0.04	0.43	0.04	0.40	100
		0.5	0.00	0.30	0.06	0.70	0.05	0.52	0.07	1.53	0.05	0.78	100
		0.75	0.03	0.58	0.06	0.94	0.05	0.90	0.07	0.98	0.08	0.08	100
	Type 2	0.25	0.04	0.74	0.06	0.83	0.05	0.74	0.05	0.74	0.05	0.74	100
		0.5	0.02	0.74	0.07	1.11	0.07	1.11	0.06	1.11	0.07	1.11	100
		0.75	0.06	1.63	0.12	1.71	0.12	1.08	0.13	1.36	0.13	1.74	100
	Type 3	0.25	0.09	0.85	0.07	0.91	0.08	0.91	0.07	0.86	0.08	0.87	100
		0.5	0.10	1.37	0.16	1.21	0.14	1.31	0.15	1.22	0.15	1.63	100
		0.75	0.19	2.77	0.17	1.58	0.23	2.04	0.17	1.78	0.18	2.29	100
	Mean		0.06		0.09		0.09		0.09		0.09		
12	Type 1	0.25	0.02	0.65	0.03	0.44	0.03	0.35	0.03	0.50	0.03	0.38	100
		0.5	0.03	1.48	0.03	0.44	0.04	0.44	0.02	0.29	0.03	0.31	100
		0.75	0.06	1.05	0.05	0.61	0.05	0.42	0.07	1.39	0.06	0.71	100
	Type 2	0.25	0.02	0.68	0.03	0.69	0.02	0.17	0.02	0.35	0.02	0.19	99
		0.5	0.03	0.74	0.03	0.75	0.04	0.57	0.03	0.76	0.02	0.41	99
		0.75	0.12	1.74	0.11	1.27	0.06	1.26	0.06	1.09	0.09	1.16	99
	Type 3	0.25	0.17	2.27	0.07	1.53	0.06	0.59	0.07	0.71	0.06	0.50	99
		0.5	0.24	2.63	0.11	1.53	0.11	1.42	0.16	1.96	0.12	1.57	100
		0.75	0.34	3.03	0.17	1.17	0.15	1.88	0.15	0.80	0.16	0.97	100
	Mean		0.11		0.07		0.06		0.07		0.07		
	Total mean		0.07		0.11		0.11		0.11		0.11		

of other CSAs reduced slightly as n increased. Overall, the means of the AEPs are 0.07, 0.11, 0.11, 0.11, and 0.11 for CSA1, CSA2, CSA3, CSA4, and CSA5, respectively, for $n = 8, 10$, and 12. CSA1 performed better than the other four CSAs.

Table 4 demonstrates the means of AEPs for different values of the n , type, and learning rate (b), respectively. Table 4 shows AEP increased as the learning rate, b , p-s-d setup time control factor varied from 0.25, 0.50 to 0.75. In terms of the types of processing times, all five CSA algorithms performed best for type 3 problem instances, while performed worst for type 1 problems. Moreover, Fig. 3 demonstrates the distributions of AEPs for five CSA algorithms. The CPU expenditures were all less than 0.1 s.

To look into the significant difference among the five CSA algorithms, an ANOVA (analysis of variance) on AEPs was run on SAS (version 9.4). The normality assumption was invalid for the linear model; the p value of the Shapiro–Wilk normality test was less than 0.0001 (with value of statistic 0.8913). Accordingly, a non-parametric statistical method was applied to perform a multiple comparison between the five CSA algorithms. Based on the ranks of AEP computed from 100 instances on the 27 (n -type- $b = 3 \cdot 3 \cdot 3$) blocks of test problem instances, a Freidman test was run and showed the p value was 0.0058 (with a Chi-square value of 14.539 and degrees of freedom of 4). Based on the commonly used level of significance $\alpha = 0.05$, the test results confirmed the AEP samples did not come from the same distribution.

Table 4 Mean AEP of n , type, and b for CSA algorithms (small n)

		CSA1	CSA2	CSA3	CSA4	CSA5
n	8	0.046	0.170	0.160	0.160	0.161
	10	0.060	0.090	0.093	0.091	0.092
	12	0.114	0.070	0.062	0.068	0.066
Type	1	0.024	0.083	0.077	0.083	0.082
	2	0.043	0.094	0.081	0.082	0.087
	3	0.152	0.153	0.158	0.154	0.149
b	0.25	0.048	0.080	0.069	0.070	0.069
	0.5	0.063	0.110	0.102	0.115	0.109
	0.75	0.108	0.141	0.145	0.133	0.141

Further comparing the pairwise differences among the five CSA algorithms, we adopted the Wilcoxon–Nemenyi–McDonald–Thompson procedure (WNMT test; refer to chapter 7 in Hollander et al. [12]. Table 5 shows the sums of ranks of AEP across the 27 blocks for the five CSA algorithms. The rank sums of CSA1 to CSA5 were 58.0, 101.5, 78.0, 85.5, and 82.0, respectively. Only one pair (among ten pairs), “CSA1 vs. CSA2”, statistically differed by less than $\alpha = 0.05$. The CSA1 was the best overall performer for a small number of jobs.

Large- n simulation results analysis

As for the tests on a large number of jobs, the number of jobs was set at $n = 80, 100$, and 120 . A set of 100 instances were randomly generated for each combination of n , type and the

learning rate (b); in total 2700 problem instances were examined. To further compare the performances of five proposed CSAs for large n in general, the criterion average relative percent deviation (RPD) is employed to evaluate the relative performance of the five proposed CSA algorithms. The RPD is defined as $RPD = 100[(A_k - A_*)/A_*][\%]$, where A_k is obtained from each algorithm and A_* was the minimum value among A_k s from the five CSAs. Table 6 displays the average RPDs for the five CSAs, while Fig. 4 presents the distributions of RPD for the five CSA algorithms. Overall, Table 6 shows the CSA3, CSA4, and CSA5 produced the lowest RPD, regardless of the number of n . Table 7 presents the means of RPD for different levels of n , type, and b , respectively. In terms of the learning rate b , from Table 7, the RPDs increased as b increased from 0.25, 0.50 to 0.75. In terms of type, all five CSAs produced the largest RPD values.

Fig. 3 AEP boxplots of CSA algorithms

Table 5 Results of the WNMT pairwise comparison of performance of CSA algorithms

Pairwise comparison between algorithm	Small n		Large n	
	Pairwise rank-sum difference	Difference > 31.7 ^a	Pairwise rank-sum difference	Difference > 31.7 ^a
CSA1 vs. CSA2	58.0–101.5	Yes	130.0–66.0	Yes
CSA1 vs. CSA3	58.0–78.0	No	130.0–74.0	Yes
CSA1 vs. CSA4	58.0–85.5	No	130.0–71.0	Yes
CSA1 vs. CSA5	58.0–82.0	No	130.0–64.0	Yes
CSA2 vs. CSA3	101.5–78.0	No	66.0–74.0	No
CSA2 vs. CSA4	101.5–85.5	No	66.0–71.0	No
CSA2 vs. CSA5	101.5–82.0	No	66.0–64.0	No
CSA3 vs. CSA4	78.0–85.5	No	74.0–71.0	No
CSA3 vs. CSA5	78.0–82.0	No	74.0–64.0	No
CSA4 vs. CSA5	85.5–82.0	No	71.0–64.0	No

^aIt was calculated approximately from a formula in Holland et al. (2014), page 316**Table 6** Performance of CSA algorithms for $n = 80, 100, 120$

n	Processing time	b	CSA1		CSA2		CSA3		CSA4		CSA5	
			Mean	Max	Mean	Max	Mean	Max	Mean	Max	Mean	Max
80	Type 1	0.25	0.00	0.14	0.00	0.00	0.00	0.00	0.00	0.01	0.00	0.00
		0.5	0.00	0.26	0.00	0.05	0.00	0.14	0.00	0.04	0.00	0.00
		0.75	0.03	0.54	0.02	0.61	0.02	0.60	0.02	0.51	0.01	0.45
	Type 2	0.25	0.00	0.11	0.00	0.10	0.00	0.00	0.00	0.00	0.00	0.00
		0.5	0.00	0.12	0.00	0.01	0.00	0.02	0.00	0.00	0.00	0.02
		0.75	0.04	1.28	0.02	0.80	0.02	0.52	0.02	0.50	0.01	0.17
	Type 3	0.25	0.38	0.69	0.02	0.33	0.02	0.21	0.03	0.23	0.03	0.35
		0.5	0.44	0.90	0.05	1.80	0.07	1.14	0.06	0.80	0.07	0.59
		0.75	0.62	1.99	0.24	1.97	0.30	2.85	0.20	1.92	0.23	2.03
	Mean		0.17		0.04		0.05		0.04		0.04	
100	Type 1	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.5	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.75	0.02	0.71	0.01	0.17	0.01	0.49	0.01	0.28	0.01	0.13
	Type 2	0.25	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.5	0.00	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.75	0.03	1.49	1.49	0.28	0.02	0.51	0.01	0.51	0.01	0.18
	Type 3	0.25	0.38	0.82	0.03	0.28	0.02	0.20	0.03	0.19	0.03	0.27
		0.5	0.44	1.40	0.06	0.51	0.09	1.42	0.08	1.04	0.05	0.82
		0.75	0.63	1.93	0.28	1.77	0.26	1.62	0.29	2.38	0.24	2.30
	Mean		0.16		0.21		0.04		0.05		0.04	
120	Type 1	0.25	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.5	0.00	0.05	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.75	0.01	0.20	0.01	0.33	0.01	0.55	0.01	0.31	0.01	0.46
	Type 2	0.25	0.00	0.11	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.5	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
		0.75	0.01	0.11	0.00	0.05	0.00	0.04	0.00	0.05	0.00	0.11
	Type 3	0.25	0.36	0.74	0.02	0.16	0.04	0.47	0.04	0.21	0.03	0.16
		0.5	0.43	2.60	0.07	0.82	0.09	0.87	0.08	0.99	0.10	0.79
		0.75	0.50	1.76	0.27	1.78	0.24	2.08	0.30	2.21	0.24	1.40
	Mean		0.15		0.04		0.04		0.05		0.04	
Total mean			0.16		0.10		0.04		0.04		0.04	

Fig. 4 RPD boxplots of CSA algorithms **Table 7** Mean RPD of n , type, and b for CSA algorithms (large n)

		CSA1	CSA2	CSA3	CSA4	CSA5
n	80	0.169	0.040	0.047	0.036	0.039
	100	0.166	0.207	0.044	0.047	0.038
	120	0.145	0.042	0.042	0.047	0.042
Type	1	0.007	0.004	0.004	0.004	0.003
	2	0.010	0.168	0.004	0.004	0.002
	3	0.464	0.117	0.125	0.123	0.114
b	0.25	0.125	0.008	0.009	0.010	0.010
	0.5	0.147	0.021	0.027	0.025	0.024
	0.75	0.209	0.261	0.097	0.095	0.084

To look into the significant differences among the five CSA algorithms, an ANOVA (analysis of variance) on RPDs was run. The normality assumption was invalid for the linear model; the p value of the Shapiro–Wilk normality test was less than 0.0001 (with value of statistic 0.5865). Accordingly, the non-parametric statistical method was applied to perform a multiple comparison between the five CSA algorithms. Based on the ranks of the RPDs computed from 100 instances on the 27 blocks of test problem instances, the Friedman test was run and showed the p value was less than 0.0001 (with a Chi-square value of 54.84 and degrees of freedom of 4). Based on the level of significance $\alpha = 0.05$, the test result confirmed the RPD samples did not come from the same distribution because [.....incomplete sentence]. Further comparing the pairwise differences among five CSA algorithms, we adopted the WNMT procedure (test). Column 4 of Table 5 shows the sum of ranks of RPD across the 27 blocks for the five algorithms. The performances among the

five CSA algorithms can be divided into two subgroups under the 0.05 level of significance. As shown in column 5 of Table 5, CSA2, CSA3, CSA4, and CSA5 with rank-sums of 66.0, 74.0, 71.0, and 64.0, respectively, were located in a better performance group while CSA1 with a rank-sum of 130.0, was located in a worse performance set. CSA1 performed statistically significantly worse than any other algorithms for large-size problem instances. Moreover, Fig. 4 demonstrated the distributions of RPDs for the five CSA algorithms.

Conclusions and suggestions

The single-machine scheduling problem has been the subject of extensive study in many variants which considered diverse processing environments and learning effect or time dependent scheduling. This was easily verified by employing the Google Scholar search function. Even during the

first half of 2020, more than 30 papers were published related to the key words “the single machine scheduling problem”. This research proposes a new variant based on the industrial setting including machine set up time and uncertainty. These characteristics lead to a complexity demanding a metaheuristic solution approach. This work developed a lower bound and a dominance property together embedded into the branch and bound method to solve the small-size problem exactly. We also provided a cloud theory-based simulated annealing (CSA) algorithm with five neighbor search methods searching for near-optimal solutions for relatively large problems. The computational simulation results in Tables 1, 3, and 6 show the branch and bound methods and the five CSA algorithms work well. Note, the proposed algorithms are not compared to other heuristics or metaheuristic algorithms because this problem was a new and unexplored one. In summary, there are three contributions of this research. (1) We introduce a single-machine scheduling problem with post-sequence-dependent setup times and scenario-dependent processing times with the objective function to minimize the maximum total completion time among scenarios. (2) To solve this model, we derive a lower bound and a property to be used in a branch-and-bound algorithm for the optimal solution. (3) We propose five variants of the cloud-theoretical simulated annealing (CSA) algorithm including new neighborhood search methods. However, the current approach assumes a deterministic parameter of uncertainty, i.e. the number of scenarios is two, the resulting schedule may not be robust if there are other types of uncertainty, for example, the uncertainty is from a continuous type of distribution. The main focus of this work is on designing five CSA algorithms which use the five neighborhoods, respectively. It would be an interesting future study to determine if CSA can obtain better results in cases where it adopts the five neighborhoods simultaneously.

Acknowledgements The authors thank the editor and two referees for their useful suggestions and helpful comments on the earlier version of our paper. This work was supported in part by the Ministry of Science and Technology of Taiwan (MOST 109-2410-H-035-019; MOST 105-2221-E-035-053-MY3), in part by the National Natural Science Foundation of China (11991022, 11971443), and in part by the National Natural Science Foundation of China (Grant Number 61873173).

Availability of data and material The data used to support the findings of this study are available from the corresponding author upon request.

Compliance with ethical standards

Conflict of interest The authors’ declares that they have no conflict of interest.

Code availability The Fortran code are available from the corresponding author upon request.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article’s Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article’s Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Adulyasak Y, Cordeau JF, Jans R (2015) Benders decomposition for production routing under demand uncertainty. *Oper Res* 63(4):851–867
2. Aissi H, Aloulou MA, Kovalyov MY (2011) Minimizing the number of late jobs on a single machine under due date uncertainty. *J Sched* 14(4):351–360
3. Allahverdi A (2015) The third comprehensive survey on scheduling problems with setup times/costs. *Eur J Oper Res* 246:345–378
4. Aloulou MA, Della Croce F (2008) Complexity of single machine scheduling problems under scenario-based uncertainty. *Oper Res Lett* 36(3):338–342
5. Cheng TCE, Lee WC, Wu CC (2010) Scheduling problems with deteriorating jobs and learning effects including proportional setup times. *Comput Ind Eng* 58:326–331
6. Ertem M, Ozcelik F, Sarac T (2019) Single machine scheduling problem with stochastic sequence-dependent setup times. *Int J Prod Res* 57(10):3273–3289
7. Fu Y, Ding J, Wang H, Wang J (2018) Two-objective stochastic flow-shop scheduling with deteriorating and learning effect in Industry 4.0-based manufacturing system. *Appl Soft Comput* 68:847–855
8. Fu Y, Wang H, Tian G, Li Z, Hu H (2019) Two-agent stochastic flow shop deteriorating scheduling via a hybrid multi-objective evolutionary algorithm. *J Intell Manuf* 30:2257–2272
9. Fu Y, Zhou MC, Guo X, Qi L (2019) Scheduling dual-objective stochastic hybrid flow shop with deteriorating jobs via bi-population evolutionary algorithm. *IEEE Trans Syst Man Cybern Syst*. <https://doi.org/10.1109/TSMC.2019.2907575>
10. Gao K, Huang Y, Sadollah A, Wang L (2020) A review of energy-efficient scheduling in intelligent production systems. *Complex Intell Syst* 6:237–249
11. Hardy GH, Littlewood JE, Polya G (1967) *Inequalities*. University Press, London
12. Hollander MD, Wolfe A, Chicken E (2014) *Nonparametric statistical methods*, 3rd edn. John Wiley & Sons, Inc, Hoboken, New Jersey
13. Kamran MA, Karimi B, Dellaert N (2020) A column-generation-heuristic-based Benders’ decomposition for solving adaptive allocation scheduling of patients in operating rooms. *Comput Ind Eng* 148 (in press)
14. Kasperski A, Zieliński P (2016) Robust discrete optimization under discrete and interval uncertainty: a survey. In: *Robustness analysis in decision aiding, optimization, and analytics*, pp 113–143
15. Koulamas C, Kyparisis GJ (2008) Single-machine scheduling problems with pastsequence-dependent setup times. *Eur J Oper Res* 187(3):1045–1049

16. Kouvelis P, Yu G (1997) Robust discrete optimization and its applications. Nonconvex optimization and its applications book, vol 14. Springer Science & Business Media
17. Li H (2017) Stochastic single-machine scheduling with learning effect. *IEEE Trans Eng Manag* 64(1):94–102
18. Lusby R, Muller LF, Petersen B (2013) A solution approach based on Benders decomposition for the preventive maintenance scheduling problem of a stochastic large-scale energy system. *J Sched* 16:605–628
19. Pinedo ML (2008) Scheduling- theory, algorithms, and systems, 3rd edn. Springer, New York
20. Sayed GI, Hassanien AE (2018) A hybrid SA-MFO algorithm for function optimization and engineering design problems. *Complex Intell Syst* 4:195–212
21. Soroush HM (2013) On the scheduling with past-sequence-dependent setup times and learning effects on a single machine. *Int J Adv Manuf Technol* 68(9):2483–2487
22. Soroush HM (2014) Scheduling in bicriteria single machine systems with past-sequence-dependent setup times and learning effects. *J Oper Res Soc* 65(7):1017–1036
23. Torabzadeh E, Zandieh M (2010) Cloud theory-based simulated annealing approach for scheduling in the two-stage assembly flow-shop. *Adv Eng Softw* 41:1238–1243
24. Urgo M, Vancza J (2019) A branch-and-bound approach for the single machine maximum lateness stochastic scheduling problem to minimize the value-at-risk. *Flex Serv Manuf J* 31:472–496
25. Wang JB (2008) Single-machine scheduling with past-sequence-dependent setup times and time-dependent learning effect. *Comput Ind Eng* 55:584–591
26. Wang JB, Li JX (2011) Single machine past-sequence-dependent setup times scheduling with general position-dependent and time-dependent learning effects. *Appl Math Model* 35(3):1388–1395
27. Wang JB, Wang D, Wang LY, Lin L, Yin N, Wang WW (2009) Single machine scheduling with exponential time-dependent learning effect and past-sequence-dependent setup times. *Comput Math Appl* 57:9–16
28. Wang XR, Wang JB, Gao WJ, Huang X (2010) Scheduling with past-sequence-dependent setup times and learning effects on a single machine. *Int J Adv Manuf Technol* 48:739–746
29. Wu CC, Gupta JND, Cheng SR, Lin BMT, Yip SH, Lin WC (2020) Robust scheduling of a two-stage assembly shop with scenario-dependent processing times. *Int J Prod Res*. <https://doi.org/10.1080/00207543.2020.1778208>
30. Xiong X, Wang D, Cheng TCE, Wu CC, Yin Y (2017) Single-machine scheduling and common due date assignment with potential machine disruption. *Int J Prod Res* 56(3):1345–1360
31. Yang J, Yu G (2002) On the robust single machine scheduling problem. *J Comb Optim* 6(1):17–33
32. Yin Y, Xu D, Wang J (2010) Some single-machine scheduling problems with past-sequence-dependent setup times and a general learning effect. *Int J Adv Manuf Technol* 48:1123–1132
33. Yin Y, Xu D, Cheng SR, Wu CC (2012) A generalization model of learning and deteriorating effects on a single-machine scheduling with past-sequence-dependent setup times. *Int J Comput Integr Manuf* 25:804–813
34. Yin Y, Cheng TCE, Xu J, Cheng SR (2013) Single-machine scheduling with past-sequence-dependent delivery times and a linear deterioration. *J Ind Manag Optim* 9:323–339
35. Yin Y, Wang W, Wang D, Cheng TCE (2017) Multi-agent single-machine scheduling and unrestricted due date assignment with a fixed machine unavailability interval. *Comput Ind Eng* 111:202–215
36. Yin Y, Li D, Wang D, Cheng TCE (2018) Single-machine serial-batch delivery scheduling with two competing agents and due date assignment. *Ann Oper Res*. <https://doi.org/10.1007/s10479-018-2839-6>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.