IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received December 6, 2018, accepted February 6, 2019, date of publication February 12, 2019, date of current version March 1, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2898905

Scheduling Jobs on a Single Machine With Dirt Cleaning Consideration to Minimize Total Completion Time

YARONG CHEN[1], LING-HUEY SU[2], YA-CHIH TSAI[3], SHENQUAN HUANG[1], AND FUH-DER CHOU[1]

[1]College of Mechanical and Electronic Engineering, Wenzhou University, Wenzhou 325035, China
[2]Industrial Engineering Department, Chung Yuan Christian University, Taoyuan 32023, Taiwan
[3]Department of Hotel Management, Vanung University, Taoyuan 32061, Taiwan

Corresponding author: Fuh-Der Chou (fdchou@tpts7.seed.net.tw)

This work was supported in part by the National Natural Science Foundation of China under Grant 51705370, and in part by the Zhejiang Province Natural Science Foundation of China under Grant LY18G010012 and Grant LY19G010007.

ABSTRACT This paper studies a single-machine scheduling problem observed in the wafer manufacturing process, where the machine must receive periodical maintenance so that the dirt generated in the process does not exceed the limit. The objective is to minimize the total completion times. A mixed binary integer programming model is formulated, and, due to its computational intractability for large problems, three effective heuristics are proposed based on our developed properties. The proposed heuristics are evaluated by comparison with exact solutions on small problems and with lower bounds on large problems. The experimental results show that the INDEX-LOE heuristic yields high-quality solutions in comparison with those obtained from the other two heuristics. Furthermore, the impacts of dirt accumulation and cleaning time are discussed in detail.

INDEX TERMS Scheduling, total completion time, machine unavailability, mixed binary integer programming, heuristic algorithms.

I. INTRODUCTION

Many researchers that have investigated single-machine scheduling problems, assumed that the machines are available all the time. However, in practical cases, machine operations have to be interrupted for some engineering needs, such as repairing, changing kits, or cleaning. This paper is motivated by a wafer fabrication process, where dirt, such as particles, organic materials, and metal-salts, on the surface of the wafer will be left in the machine during wafer processing. Once the accumulation of dirt reaches a threshold value, the wafer will be damaged. Thus, the machine has to be stopped to remove the accumulated dirt with a cleaning agent. That is, the cleaning agent cleans the machine so that the accumulated dirt does not exceed the threshold value. In the cleaning period, the machine cannot process any jobs.

This paper studies a single-machine scheduling problem with flexible maintenance; the flexible maintenance in this paper specifies that the starting time of maintenance activity is determined by the amount of dirt accumulated and the threshold value. This situation is commonly encountered in a wafer manufacturing company, and these non-available periods of a machine will affect the scheduling and the system's performance. For the problem, we consider the objective of minimizing the total completion times. This objective implies minimizing the work-in-process (WIP) inventory, which is an important internal managerial performance measure for a company. According to the notation [1], the problem is denoted $1\,|cleaning|\sum C_j$, where the first field denotes a single machine, the second field denotes the cleaning activities and the third field denotes the total completion time.

In the literature, many studies assumed that maintenance is required in fixed intervals or during some window times. Among them, the first study related to our study was conducted by Yang *et al.* [2]. Other related articles include the following (Schmidt and Sanlaville [3], Qi *et al.* [4], Schmidt [5], Liao and Chen [6], Chen [7], Chen [8], Ji *et al.* [9], Chen [10], Mosheiov and Sarig [11], Low *et al.* [12], Ma *et al.* [13], Yang *et al.* [14], Zammori *et al.* [15], Xu *et al.* [16], Yin *et al.* [17], and Yin *et al.* [18]).

The associate editor coordinating the review of this manuscript and approving it for publication was Bora Onat.

The variable maintenance defines that the duration of maintenance depends on its start time. That is, the sooner the maintenance is started, the shorter the length of the maintenance time is, which is also called deteriorating maintenance. Kubzin and Strusevich [19] added the variable maintenance restriction into the two-machine flow shop and two-machine open shop problems, where the objective is to minimize the makespan. They showed that the open-shop problem is polynomially solvable, while the flow shop problem is binary NP-hard and pseudo-polynomially solvable with dynamic programming. Mosheiov and Sidney [20] also considered the deteriorating maintenance activity. The difference is that the processing time of jobs is affected by the maintenance activity, that is, the processing time of job j will decrease if job j is processed after the maintenance activity. For the problem, they proposed polynomial time solutions for the problems with different objectives. Yin *et al.* [21] considered position-dependent deteriorating jobs and deteriorating maintenance activities simultaneously on a single-machine scheduling problem, and their objective is to jointly minimize the cost of due-date assignment, and the cost of discarding jobs. The study of Luo *et al.* [22] is inspired by the above two works by Kubzin and Strusevich [19] and Mosheiov and Sidney [20], and proposed two approximation algorithms to minimize the total weighted completion time. Xu *et al.* [23] considered two scheduling problems with a single machine and parallel machine where the maintenance is an increasing linear function and the maintenance activity has to be implemented within a prefixed interval. The objective is to minimize the makespan. They developed two approximation algorithms for the problems. Motivated by the request serving process in a wireless sensor network, Gu *et al.* [24] considered single-machine problems with the machine aging effect, where maintenance is implemented once to recover the service ability. For the problem, they adopted two objectives of minimizing the makespan and the total completion time, respectively, and proposed two dynamic programming algorithms. Luo *et al.* [25] investigated scheduling jobs and variable maintenance activity on a single machine and provided polynomial-time algorithms to solve the problems while minimizing the makespan, total completion time, maximum lateness, and number of tardy jobs. Zhu *et al.* [26] added resources into the maintenance activity in the problem with deteriorating processing time, where the greater the amount of resources allocated to the maintenance activity, the shorter the duration of maintenance. For the problem, they considered different objective functions and proved that the problems are polynomially solvable. Recently, Luo and Liu [27] extended the study of Xu *et al.* [23], where the objective is to minimize the total weighted completion, and they proposed two approximation algorithms for the problem. Ying *et al.* [28] extended the problem studied by Luo *et al.* [25] to consider four different objective functions and proposed an exact algorithm with the computational complexity $O(n^2)$ for each problem. Su and Wang [29] studied a single machine problem with multiple unavailability periods, where the machine has to be interrupted to remove dirt. Their objective is to minimize the total absolute deviation of job completion time (TADC).

Some researches attempted to consider uncertain maintenance activities in production scheduling with different optimization objectives in different manufacturing shop floors [30], [31]. Xiong *et al.* [32] consider the machine disruption may occur at a particular time in a single machine, and the maintenance time will last for a period of time with a certain probability. For the problem, they proposed different approximation approaches to minimize the expected integrated cost function including the earliness, tardiness and due date assignment cost. Yin *et al.* [33] extended the study of Xiong *et al.* [32] to consider parallel-machine scheduling problem, and provide polynomial-time approximation schemes to solve the problem of minimizing the expected total completion time.

In this work, we considered the same problem studied by Su and Wang [29]; the difference is that our objective is to minimize the total completion time. Many dominance properties are explored and used in the lower bound calculation and heuristic algorithms. The rest of this paper is organized as follows. Section 2 defines the problem, presents many properties to optimally solve the problem and proposes two lower bounds. A mixed binary integer programming model is developed in Section 3. In Section 4, we present additional properties for the heuristic algorithms and propose three heuristic algorithms. Section 5 gives the computational experiments, and finally, in Section 6, the conclusions are provided along with some future research directions.

II. PROBLEM DEFINITION

This paper considers a nonresumable single-machine scheduling problem; i.e., once a job is started, it cannot be interrupted until its completion. The setup time of the job is sequence independent and included in the processing time. There are n jobs to be processed at time zero. Each job has a processing time p_i and an amount of dirt t_i left on the machine where $i = 1, 2, \ldots, n$. A cleaning activity with time w is carried out before the accumulation of dirt reaches a threshold value T, where $t_i \leq T$. The objective is to minimize the total completion time.

A. NOTATION AND PROBLEM SETTING

The following notations will be used throughout the study:

J_i job number $i (i = 1, 2, \ldots, n)$;

C_j completion time of the job at the j_{th} position in a given sequence $(j = 1, 2, \ldots, n)$;

$k_j 1$, if the cleaning activity is taken immediately following the j_{th} position job; 0, otherwise

$x_{ij} 1$, if J_i is scheduled at position j; 0, otherwise.

In addition, $J_{[j]}$ denotes the job in sequence position j, and $p_{[j]}$ and $t_{[j]}$ are defined accordingly.

The objective is to find a schedule that minimizes the total completion time (TC), represented as $TC = \sum_{j=1}^{n} C_j$ Denote a schedule π containing a sequence of jobs and several

cleaning activities inserted in the job sequence. In the schedule, those jobs processed between two adjacent cleaning activities form a batch, denoted B_l where $l = 1, .., L$. Thus, a schedule π can be denoted $\pi = (B_1, w, B_2, w, \ldots, B_L)$. Note that L denotes the number of batches and is a decision variable in our problem.

Theorem 1: The problem 1| cleaning $|\sum C_j$ is strongly NP-hard.

Proof: The problem 1|cleaning$|\sum C_j$ is strongly NP-hard because the special case, where each dirt is equal to its corresponding processing time, and the machine must be implemented cleaning activity after a maximum allowed dirt, i.e., 1|cleaning, $t_j = p_j|\sum C_j$, is NP-hard. (Qi *et al.* [4]).

Obviously, there are at least ($L^{'}$-1) cleaning activities in an optimal schedule. It is worth mentioning that the 1|cleaning$|\sum C_j$ problem with agreeable processing time and dirt, i.e., $p_i \le p_j$ implies that $t_i \le t_j$ is also NP-hard because the special case of 1|cleaning, $t_j = p_j|\sum C_j$ is NP-hard.

For the problem 1|cleaning $|\sum C_j$, the optimal schedule may have a larger number of batches than the schedule that contains the minimum number of batches. Therefore, the optimal schedule may need more cleaning activities to achieve a lower total completion time. The following is a simple example.

Numerical Example 1: Let $n = 4, p_1 = 1, p_2 = 3, p_3 = 1, p_4 = 3, t_1 = 1, t_2 = 2, t_3 = 1, t_4 = 2$, and $T = 3$. The schedule π_1 with the minimum number of batches is $\pi_1 = (J_1, J_2, w, J_3, J_4)$ with $L = 2$ and $TC = 18+ 2w$. Consider the schedule $\pi_2 = (J_1, J_3, w, J_2, w, J_4)$ with $L = 3$ and $TC = 16 + 3w$. Then we can see that the second schedule that encounters two cleaning activities is better than the first schedule with one cleaning activity when $w < 2$.

Therefore, in the optimal schedule, the cleaning activity may be carried out even if the machine can process more jobs within the batch.

In the following Theorem, we show that the total completion time in a schedule π containing both jobs and cleaning activities can be calculated in a more efficient way.

Lemma 1: The *TC* value of a given schedule π containing jobs and cleaning activities can be calculated as

$TC = \sum_{j=1}^{n}(n-j+1)p_{[j]} + w\sum_{j=1}^{n-1}(n-j)k_j$, where the value of k_j equals 1 if the cleaning activity is conducted immediately following the j_{th} position job; otherwise, it equals 0.

Proof: Let the number of jobs in the k_{th} batch be β_k.

$$\begin{aligned} TC &= \sum_{j=1}^{n} C_j \\ &= \sum_{j=1}^{n}(n-j+1)p_{[j]} + \sum_{k=1}^{B_{L-1}}(n - \sum_{i=1}^{k}\beta_i)w \\ &= \sum_{j=1}^{n}(n-j+1)p_{[j]} + w\sum_{j=1}^{n-1}(n-j)k_j, \end{aligned} \quad (1)$$

where the former factor (*n*-*j* + 1) is the positional weight independent of p_j and the latter factor $(n-j) \times k_j$ is the positional weight of the cleaning activity immediately following job sequence *j*.

Numerical Example 2: The data in example 1 is considered. The job sequence $\pi = (J_1, J_3, \mathrm{w}, J_2, \mathrm{w}, J_4)$ can be scheduled as follows.

$$\begin{aligned} TC &= 4 \times p_1 + 3 \times p_3 + 2 \times p_2 \\ &\quad + 1 \times p_4 + (4-2)w + (4-3)w \\ &= 16 + 3w \end{aligned}$$

According to Lemma 1, the positional weight of the cleaning activity is equal to that of the job immediately following it and thus the problem can be solved polynomially when each cleaning activity is carried out after a fixed number of jobs.

B. CALCULATION OF LOWER BOUND

Two lower bounds are proposed as benchmarks to evaluate the heuristics. Based on the fact that the *SPT* rule gives an optimal solution to the problem $1\|\sum C_j$ and that the positional index of the cleaning activity's contribution to the total completion time is in descending order due to Lemma 1, we now combine these properties with the assumption that the dirt accumulation is resumable, i.e., the dirt accumulation interrupted by a cleaning activity can resume, in order to obtain the lower bound of the total completion time *LB*(*TC*).

Denote *e* as the elapsed time between the end time of the last cleaning activity and the completion of the incumbent job. Additionally, let *Z* be the lower bound of the cleaning activity's contribution to the total completion time. The procedure to obtain *LB*(*TC*) is as follows.

Step 1. Sequence the processing time p_i in ascending order and the dirt t_i in descending order of all *n* jobs, where $i = 1$, 2,..,*n*. Set $i = 1, e = 0, Z = 0$.

Step 2. If $i > n$, then set $LB(TC) = \sum_{j=1}^{n}(n-j+1)p_j + Z$ and ***Stop***; else, go to ***Step 3***.

Step 3. If $(e + t_i) > T$, then Z = Z + w (n - i) and $e = e + t_i - T$; else, $e = e + t_i$. Set $i = i + 1$ and return to ***Step 2***.

The lower bound *LB*(*TC*) obtained with complexity $O(2n \cdot \log n)$ is very efficient but may not be so tight. It is proposed as a benchmark when evaluation of our heuristics in a very short time is essential. To evaluate the effectiveness of our heuristics precisely, a tighter lower bound obtained using IBM ILOG CPLEX Optimization Studio Ver. 12.6.1 is proposed. First, the global time limit is changed from the default value of infinity to the specified value (we set 600 seconds in our case). Then, the script code based on ILOG Script is developed to obtain the upper and the lower bounds of the objective function value, as well as the CPU time. If the CPU time does not exceed the time limit, then the upper bound obtained is the optimal solution. Otherwise, the lower bound obtained is used for evaluating the effectiveness of the proposed heuristics.

III. MIXED BINARY INTEGER PROGRAMMING (MBIP) MODEL

In this section, an *MBIP* model for optimally solving the problem $1\,|cleaning|\sum C_j$ is formulated

as follows:

$$\text{Minimize} \sum_{j=1}^{n} (n-j+1)p_{[j]} + w\sum_{j=1}^{n-1}(n-j)k_j \quad (2)$$

$$\textbf{Subject to}: \sum_{j=1}^{n} x_{ij} = 1 \quad \forall\, i = 1, 2, 3, \ldots, n \quad (3)$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad \forall\, j = 1, 2, 3, \ldots, n \quad (4)$$

$$p_{[j]} = \sum_{i=1}^{n} \left(p_i \ldots x_{ij}\right) \quad \forall\, j = 1, 2, 3, \ldots, n \quad (5)$$

$$C_1 = \sum_{i=1}^{n} \left(p_i \ldots x_{i1}\right) \quad (6)$$

$$C_{j-1} + \sum_{i=1}^{n} \left(p_i \ldots x_{ij}\right) + w \le C_j + M\left(1 - k_{j-1}\right) \quad \forall\, j = 2, 3, 4, \ldots, n \quad (7)$$

$$C_{j-1} + \sum_{i=1}^{n} \left(p_i \ldots x_{ij}\right) \le C_j + M \ldots k_{j-1} \quad \forall\, j = 2, 3, 4, \ldots, n \quad (8)$$

$$t_{[1]} = \sum_{i=1}^{n} \left(t_i \ldots x_{i1}\right) \quad (9)$$

$$t_{[j-1]} + \sum_{i=1}^{n} \left(t_i \ldots x_{ij}\right) \le t_{[j]} + M \ldots k_{j-1} \quad \forall\, j = 2, 3, 4, \ldots, n \quad (10)$$

$$\sum_{i=1}^{n} \left(t_i \ldots x_{ij}\right) \le t_{[j]} + M \ldots \left(1 - k_{j-1}\right) \quad \forall\, j = 2, 3, 4, \ldots, n \quad (11)$$

$$T \ge t_{[j]} - M \ldots \left(1 - k_j\right) \quad \forall\, j = 1, 2, 3, \ldots, n-1 \quad (12)$$

$$T \ge t_{[j]} - M \ldots k_j \quad \forall\, j = 1, 2, 3, \ldots, n-1 \quad (13)$$

$$t_{[j]} \ge 0, \quad C_j \ge 0 \quad \forall\, j = 1, 2, 3, \ldots, n \quad (14)$$

$$x_{ij} \text{ is binary}, \quad \forall i = 1, 2, 3, \ldots, n; \; j = 1, 2, 3, \ldots, n \quad (15)$$

$$k_j \text{ is binary}, \quad \forall\, j = 1, 2, 3, \ldots, n-1 \quad (16)$$

The model mentioned above is modified from the one proposed by Su and Wang [29], in which their objective function minimizes the total absolute deviation of the job completion time, whereas in our model, the objective function (2) describes the minimum *TC* according to Lemma 1. Constraints (3) and (4) dictate that each job must be placed at one position, and that each position can only perform one job. Constraint (5) defines the processing time of $J_{[j]}$. Constraint (6) specifies the completion time of the job at the first position. If a cleaning activity is performed immediately after the $(j\text{-}1)_{th}$ job, then the cleaning time w is added to the completion time of the j_{th} job. Constraints (7) and (8) together define the completion times of the jobs processed after the first one by combining the binary variable k_{j-1} with an extremely large positive number M. Constraints (9-11) define the dirt accumulation between the completion of the last cleaning activity and the completion of the j_{th} job. Constraints (12-13) ensure that the accumulation of dirt in each batch cannot exceed the threshold value. Finally, constraints (14-16) define the nonnegativity of $t_{[j]}$ and C_j, and the binary restrictions for x_{ij} and k_j.

IV. HEURISTIC ALGORITHMS

Since our considered problem is strongly NP-hard, it is difficult to apply the mixed BIP to optimally solve large-scale problems due to the considerable computational burden. Therefore, three heuristics are proposed. We first introduce some properties of the optimal schedule for the heuristics.

A. SOLUTION PROPERTIES

Recall that the shortest processing time (*SPT*) rule solves the problem $1||\sum C_j$. For the $1\,|cleaning|\sum C_j$ problem, we derive the following property to solve it.

Property 1: There exists an optimal schedule in which jobs in each batch are sequenced in *SPT* order.

Property 2: Consider a sequence where $J_{[j]}$ is to be shifted to the position immediately preceding $J_{[i]}$, where $i < j$, without violating the dirt constraint and suppose that there are k cleaning activities between $J_{[i]}$ and $J_{[j]}$. If $\Delta = (j-i)\,p_{[j]} - \sum_{u=i}^{j-1} p_{[u]}kw < 0$, then *TC* is decreased by this shift.

Proof: When $J_{[j]}$ is to be moved immediately before $J_{[i]}$, the change in the cost is

$$\begin{aligned}\Delta_1 &= (n-i+1)\,p_{[j]} - (n-j+1)p_{[j]} \\ &= (j-i)p_{[j]}\end{aligned}$$

The jobs from the i_{th} position to the $(j-1)_{th}$position are moved backward one position as $\Delta_2 = \sum_{u=1}^{j-1}[(n-u) - (n-u+1)]p_{[u]} = -\sum_{u=i}^{j-1} p_{[u]}$

Each cleaning activity in between is moved backward one position as

$$\begin{aligned}\Delta_3 &= -kw \\ \Delta &= \Delta_1 + \Delta_2 + \Delta_3\end{aligned}$$

This completes the proof.

Property 3: Given σ the list of the remaining unscheduled jobs in *SPT* order are indexed as $J_{\sigma_1}, J_{\sigma_2}, .., J_{\sigma_r}$. If J_{σ_1} cannot be scheduled at the i_{th} job sequence position in the current batch B_l due to $\sum_{k\in B_l} t_k + t_{\sigma_1} > T$, then job $J_{\sigma_i}, i = 2,3,\ldots, r$ where $\sum_{k\in B_l} t_k + t_{\sigma_1} \le T$ and $(i-1)\,p_{\sigma_i} - \sum_{k=1}^{i-1} p_{\sigma_k} - w \le 0$ should be scheduled in B_l.

Proof: If J_{σ_i} is available to be scheduled in the current batch B_l at the ith sequence position, then job J_{σ_i} contributes $\Delta_1 = (i-1)p_{\sigma_i}$ to *TC*.

The jobs from J_{σ_1} to $J_{\sigma_{i-1}}$ in σ are moved backward one position and the total completion time is changed as

$$\Delta_2 = -\sum_{k=1}^{i-1} p_{\sigma_k}$$

The cleaning activity is moved backward one position and the total completion time is changed as

$$\begin{aligned}\Delta_3 &\ge -w \\ \Delta &= \Delta_1 + \Delta_2 + \Delta_3\end{aligned}$$

This completes the proof.

Numerical Example 3: Suppose that there are four unscheduled jobs in *SPT* order. Their processing times are $p_1 = 6$, $p_2 = 7$, $p_3 = 7$, and $p_4 = 8$, and their dirt amounts

are $t_1 = 2$, $t_2 = 2$, $t_3 = 2$, and $t_4 = 1$. Suppose the current batch is B_l and $T - \sum_{k \in B_l} t_k = 1$. To schedule with the *SPT* rule, the unscheduled jobs will be processed as schedule I as in Figure 1, but it could be better in schedule II. With $w = 5$, the total completion time is $67 + 4w$ in schedule I, which equals 87. Meanwhile, in schedule II, the total completion time is $71 + 3w$, which equals 86.

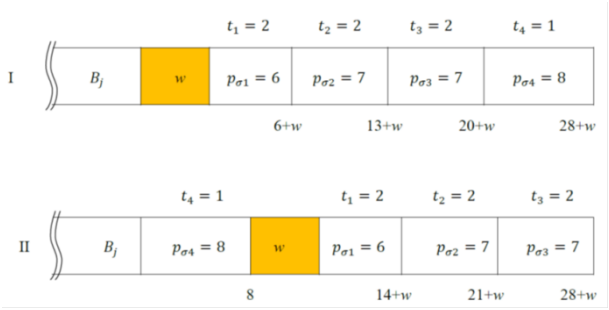

FIGURE 1. Illustration of property 3.

The ***LOE*** (last-only-empty) approach, a well-known and commonly encountered approach in practice, is introduced.

LOE: batching in which the accumulated dirt of each batch is kept full whenever possible except for the last one.

Property 4: If the sequence of all jobs is fixed, then the *LOE* rule gives an optimal solution.

Proof: According to Lemma 1, the *TC* value of a given schedule π containing jobs and cleaning activities can be calculated as

$TC = \sum_{j=1}^{n} (n - j + 1) p_{[j]} + w \sum_{j=1}^{n-1} (n - j) k_j$, where the value of k_j equals 1 if a cleaning activity occurs immediately following the j_{th} position job; otherwise, it equals 0. Since the sequence of all jobs is fixed, the term $\sum_{j=1}^{n} (n - j + 1) p_{[j]}$ is constant. Another term $\sum_{j=1}^{n-1} (n - j) k_j$ can be recalculated as $\sum_{j=1}^{n-1} (n - j) k_j = (n - 1)k_1 + (n - 2)k_2 + (n - 3)k_3 + \ldots + k_{n-1} = n \sum_{i=1}^{n-1} k_i - \sum_{i=1}^{n-1} i k_i$

To minimize the value of $n \sum_{i=1}^{n-1} k_i$, the number of batches is as small as possible; and in order to maximize $\sum_{i=1}^{n-1} i k_i$, the cleaning activity is taken as late as possible. Therefore, the *LOE* rule gives an optimal solution.

B. HEURISTC ALGORITHMS

We develop three heuristics, *SPT-LOE, SPCT-LOE,* and *INDEX-LOE*, to solve the $1|cleaning|\sum C_j$ problem. Initially, in each heuristic, a greedy algorithm is applied to sequence both the jobs and cleaning activities. An improvement algorithm is then applied. If, at any time, the processing time of a job is greater than that of the job in the following batches and the interchange these two jobs does not violate the dirt capacity constraint, then the interchange is performed. The detailed steps of the three heuristics are outlined as follows:

1) HEURISTIC SPT-LOE

The algorithm commonly used in the real world applies the *SPT* rule to sort the job and follows the *SPT* order to form the batch using the *LOE* rule. If a job is added to batch B_l, resulting in the total amount of dirt accumulation exceeding T, then the job is added to the next batch B_{l+1}. An improvement algorithm as previously described is then applied. The steps of the heuristic algorithm are as follows.

Step 1. Sequence all n jobs in *SPT* order and break any ties in favor of less dirt.

Step 2. Use the *LOE* rule to insert the cleaning activities.

Step 3. Apply the improvement algorithm to interchange the jobs in each batch with the job having the larger processing time being placed in the following batches provided that the interchange does not violate the dirt constraint. The procedure is continued until the last batch has been considered.

Step 4. Apply Property 1 to sequence the jobs in each batch in *SPT* order.

2) HEURISTIC SPCT-LOE

Since the processing time of a job and the dirt left on a machine do not agreeable, the processing time of a job is adapted by adding the estimated cleaning time proportional to the amount of dirt. The heuristic *SPCT-LOE* is outlined as follows.

***Step 1*:** Calculate the modified processing time for each job as $p_j^{'} = p_j + \left(t_j / T\right) w$, *where*$j = 1, \ldots, n$

Step 2: Find a job sequence by ordering all jobs in nondecreasing order of the modified processing time, $p_j^{'}$, and break any ties in favor of the smaller processing time.

***Step 3*:** Assign unscheduled jobs one by one to the machine until a job cannot be scheduled to the current batch due to the dirt constraint. In this case, assign a job that has not scheduled yet according to Property 3 to the last position of the current batch. If no such job is found, arrange a cleaning activity and build a new batch as the current batch. The step is continued until all jobs have been scheduled.

Step 4. Use Property 1 to sequence the jobs in each batch based on the *SPT* rule.

3) HEURISTIC INDEX-LOE

An index for each job j in each iteration is developed by the following expression:

$$I_j = (n - iteration + 1) \times p_j + (n - iteration) \times w \times \left(\frac{e + t_j}{T} + b \right),$$

where the value of e denotes the amount of accumulated dirt in the current batch before job J_j and b equals 1 if $e + t_j > T$; otherwise, it equals 0. The job with the smallest I_j is selected to be scheduled to the machine. If the selected job cannot be scheduled to the current batch due to the dirt constraint, then a cleaning activity is inserted. The steps of the *INDEX-LOE* heuristic are outlined as follows.

***Step 1*:** Set $e = 0$, Iteration $= 1$. Let S denote the set of all scheduled jobs and U the set of all unscheduled jobs. Initially, $S = \emptyset$ and $U = \{J_1, J_2, .., J_n$.

***Step 2*:** If $U = \emptyset$, go to *Step* 5. Otherwise, calculate the index of all jobs in U and find the smallest one, denoted I_j. If $e + t_j \leq T$, go to *Step* 3; otherwise, go to *Step* 4.

***Step3*:** Schedule J_j to the last position of the current batch. Let $S \leftarrow S \cup J_j$, $U \leftarrow U \backslash J_j$, $e = e + t_j$, Iteration = Iteration + 1 and go to *Step* 2.

***Step 4*:** Schedule a cleaning activity after the current batch and assign job J_j to a new batch. Let $S \leftarrow S \cup J_j$, $U \leftarrow U \backslash J_j$, $e = t_j$, Iteration = Iteration + 1 and go to *Step* 2.

***Steps* 5** and **6** are the same as *Steps* 3 and 4 in the *SPT-LOE* heuristic.

Numerical Example 4: The data in example 1 is considered. Let $w = 1$.

Iteration 1: $e = 0$, $S = \emptyset$, and $U = \{J_1, J_2, J_3, J_4\}$.

$I_1 = 4\times1+3\times1\times(1/3) = 5, I_2 = 4\times3+3\times1\times(2/3) = 14, I_3 = 4\times1+3\times1\times(1/3) = 5, I_4 = 4\times3+3\times1\times(2/3) = 14$. The smallest index is that with J_1 and J_3. Job J_1 is selected arbitrarily and is placed in position 1.

Iteration 2: $e = 1$, $S = \{J_1$, $U = \{J_2, J_3, J_4\}$.

$I_2 = 3\times3+2\times1\times(3/3) = 11, I_3 = 3\times1+2\times1\times(2/3) = 4.3$, $I_4 = 3 \times 3 + 2 \times 1 \times (3/3) = 11$. Job J_3 is placed in position 2.

Iteration 3: $e = 2, S = \{J_1, J_3, U = \{J_2, J_4\}$.

$I_2 = 2 \times 3 + 1 \times 1 \times \{[(2+2)/3] + 1\} = 8.3$, $I_4 = 2 \times 3 + 1 \times 1 \times \{[(2+2)/3] + 1\} = 8.3$. Job J_2 is selected arbitrarily and is placed in position 3, and the remaining job, J_4, is placed in position 4. This results in the sequence $\pi = (J_1, J_3, w, J_2, w, J_4)$, which is the optimal schedule.

V. COMPUTATIONAL EXPERIMENTS

The computational experiments aim to examine the performances of the proposed algorithms, including the Mixed Binary Integer Programming (MBIP) model and the heuristic algorithm. The proposed MBIP model is implemented by IBM ILOG CPLEX Optimization Studio Ver. 12.6.1 software and the heuristics as well as the lower bound are coded in C++. All experiments are run on a PC with an i3-530 CPU. Two sets of experiments are carried out. The first set aims to evaluate the efficiency of the MBIP model and the effectiveness of the heuristic algorithm for small problems. Another set evaluates the performances of the heuristics using the lower bound as a benchmark for large problems.

The parameter settings n, p_i, t_i, T, and w for small and large problems are shown in Table 1, in which the parameters α, β and γ control the amount of dirt left by each job, the length of the cleaning time, and the dirt threshold, respectively. The experimental procedure consists of a design of dirt with two settings of α ($\alpha = 0.2$ *and* 0.4) and two settings of γ ($\gamma = 2$ and 4). The two settings of $\beta(\beta = 1.5$ and 2.5), which controls the cleaning times, refer to the maintenance times of Liao and Chen [6].

For each combination of n, p_i, t_i, T, and w, ten instances are generated, yielding 1440 instances for the problem. The formula, dev(%) $= (heuristic - optimum)/optimum$, is used to determine the deviation of our heuristic solution over the optimal solution for small problem instances. On the other hand, the formula dev(%)$= (heurisitc - LowerBound)/LowerBound$ is employed to determine the deviation of our heuristics over the Lower Bound.

TABLE 1. Data for computational experiments.

	Small problem							Large problem	
n	8	10	15	20	30	40	50	100	200
p_i	$U[1, p_{max}]$, $p_{max} = 10, 100$								
t_i	$U[1, \alpha \times p_i]$, $\alpha = 0.2, 0.4$								
T	$\gamma \times t_{max}, \gamma = 2, 4$								
w	$int\left(\beta \times \frac{\sum p_i}{N}\right), \beta = 1.5, 2.5$								

A. COMPARISON OF OUR HEURISTIC WITH BIP FOR SMALL PROBLEMS

For small problems with $n = 8$, 10, 15, 20, 30, 40 and 50, the computational results are shown in Tables 2 and 3 for the processing time distributions U(1,10) and U(1,100), respectively. Both tables show the average CPU time in seconds for the MBIP model and the heuristic algorithm, as well as the average deviation of the heuristic solution over the optimal solution. Table 4 further summarized the average deviation of each heuristic over the optimal solution.

The results in Tables 2 and 3 show that the MBIP model solves all the instances within four seconds when the number of jobs n is not larger than 20. However, the required computational time increases drastically when n increases to 50. The instance of widely dispersed job processing times U(1,100) needs much more execution time than that of U(1,10). For example, when $n = 50$, the average execution time for the job processing time distribution U(1,10) is 371.553 seconds, while it is 4852.296 seconds for U(1,100). The execution times of all three heuristics are almost zero when $n \leq 50$. With regard to the quality of the heuristics, the summary results are given in Table 4. The average deviations of the three heuristics, *SPT-LOE*, *SPCT-LOE* and *INDEX-LOE*, from the optimal solution are 2.308%, 0.317%, and 0.281%, respectively, for the processing time distributions U(1,10). For the processing time distributions U(1,100), the deviations of *SPT-LOE*, *SPCT-LOE* and *INDEX-LOE* from the optimal solution are 5.252%, 0.741%, and 0.669% on average, respectively. The influences of the two processing time distributions associated with the three heuristics on the solution quality are shown in Figure 2. The heuristic solutions perform slightly worse as the dispersion of the job processing times widens. From Figure 2 or Table 4, it is clear that *INDEX-LOE* outperforms the other two heuristics, especially *SPT-LOE*, which is commonly used in practice. The impacts of the

TABLE 2. The results of $p_i = U$ [1,10] for small problems.

		α	0.2				0.4				
	p_i[1,10]	β	2		4		2		4		
		γ	1.5	2.5	1.5	2.5	1.5	2.5	1.5	2.5	Ave.
n=8	*BIP*	CPU(s)	0.102	0.114	0.111	0.120	0.123	0.097	0.100	0.094	0.108
	SPT-LOE	Dev%	2.470	1.410	4.670	2.010	2.720	0.840	7.470	2.980	3.071
	SPCT-LOE	Dev%	0.790	0.590	0.630	0.570	0.800	0.830	0.000	1.630	0.730
	INDEX-LOE	Dev%	0.880	0.240	0.210	0.570	0.750	0.590	0.000	1.170	0.551
n=10	*BIP*	CPU(s)	0.128	0.130	0.106	0.117	0.119	0.149	0.130	0.128	0.126
	SPT-LOE	Dev%	2.240	1.460	4.950	3.200	5.100	0.920	7.760	2.160	3.474
	SPCT-LOE	Dev%	0.030	0.210	0.370	0.500	0.640	0.080	1.150	0.580	0.445
	INDEX-LOE	Dev%	0.180	0.210	0.100	0.000	0.290	0.080	1.080	0.140	0.260
n=15	*BIP*	CPU(s)	0.581	0.651	0.488	0.497	0.778	0.731	0.812	0.703	0.655
	SPT-LOE	Dev%	1.610	1.160	4.880	1.060	2.570	1.550	6.390	2.770	2.749
	SPCT-LOE	Dev%	0.190	0.150	0.350	0.330	0.230	0.260	1.030	0.080	0.328
	INDEX-LOE	Dev%	0.370	0.100	0.300	0.330	0.110	0.230	0.780	0.080	0.288
n=20	*BIP*	CPU(s)	2.163	2.542	2.205	2.244	2.322	3.223	2.456	1.875	2.379
	SPT-LOE	Dev%	1.580	1.010	4.070	1.760	3.030	1.270	6.820	3.010	2.819
	SPCT-LOE	Dev%	0.060	0.200	0.210	0.120	0.640	0.440	0.450	0.390	0.314
	INDEX-LOE	Dev%	0.330	0.230	0.130	0.120	0.570	0.320	0.820	0.270	0.349
n=30	*BIP*	CPU(s)	3.469	73.714	3.233	8.247	4.977	129.241	4.531	17.131	30.568
	SPT-LOE	Dev%	0.260	0.180	1.740	0.620	0.920	0.320	2.380	1.010	0.929
	SPCT-LOE	Dev%	0.060	0.110	0.020	0.030	0.150	0.040	0.140	0.110	0.083
	INDEX-LOE	Dev%	0.030	0.070	0.010	0.030	0.100	0.040	0.180	0.100	0.070
n=40	*BIP*	CPU(s)	6.822	726.692	6.547	7.648	10.366	754.428	9.708	358.740	235.119
	SPT-LOE	Dev%	0.390	0.130	0.860	0.320	0.750	0.500	1.780	1.000	0.716
	SPCT-LOE	Dev%	0.030	0.060	0.020	0.040	0.190	0.150	0.120	0.240	0.106
	INDEX-LOE	Dev%	0.020	0.060	0.010	0.040	0.100	0.110	0.130	0.240	0.089
n=50	*BIP*	CPU(s)	28.993	16.975	1797.710	15.625	203.451	38.472	819.302	51.898	371.553
	SPT-LOE	Dev%	1.500	0.950	3.020	1.240	3.390	0.950	5.880	2.230	2.395
	SPCT-LOE	Dev%	0.090	0.180	0.080	0.120	0.460	0.200	0.410	0.170	0.214
	INDEX-LOE	Dev%	0.130	0.090	0.140	0.120	1.050	0.200	0.820	0.340	0.361

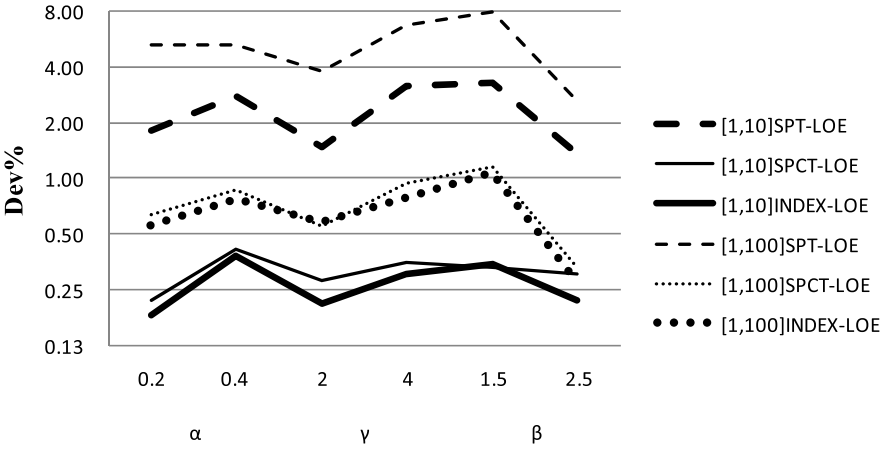

FIGURE 2. The influences of processing time distribution, α, β and γ on the solution quality for small problems.

parameters α, β and γ on the solution quality are analyzed and depicted in Figure 2.

The parameter α controls the amount of dirt left by a job on the machine. The larger the value of α is, the more dirt left by a job. Figure 2 shows that the deviation of the heuristic from the optimal solution increases as the value of α increases. The parameter β represents the influence of the cleaning time on solution quality. The larger the value of β is,

TABLE 3. The results of $p_i = U$ [1,100] for small problems.

		α	0.2				0.4				
p_i[1,100]		β	2		4		2		4		
		γ	1.5	2.5	1.5	2.5	1.5	2.5	1.5	2.5	Ave.
n=8	*BIP*	CPU(s)	0.088	0.083	0.080	0.081	0.094	0.084	0.098	0.083	0.086
	SPT-LOE	Dev%	5.200	2.650	9.770	4.020	4.420	0.940	9.480	2.750	4.904
	SPCT-LOE	Dev%	0.100	0.000	0.680	0.000	0.970	0.030	2.360	0.520	0.583
	INDEX-LOE	Dev%	0.040	0.000	0.680	0.000	1.340	0.030	1.040	0.000	0.391
n=10	*BIP*	CPU(s)	0.180	0.181	0.177	0.100	0.241	0.139	0.187	0.123	0.166
	SPT-LOE	Dev%	6.240	2.170	8.400	2.260	7.790	1.540	11.310	2.080	5.224
	SPCT-LOE	Dev%	0.340	0.060	1.460	0.270	1.800	0.230	1.990	0.000	0.769
	INDEX-LOE	Dev%	0.240	0.060	1.200	0.270	1.850	0.440	2.000	0.000	0.758
n=15	*BIP*	CPU(s)	1.438	1.245	1.316	0.786	1.587	1.155	1.169	0.863	1.195
	SPT-LOE	Dev%	5.590	2.390	14.390	3.370	7.330	2.340	14.500	4.870	6.848
	SPCT-LOE	Dev%	1.060	0.440	2.440	0.270	1.120	0.460	2.510	1.040	1.168
	INDEX-LOE	Dev%	0.950	0.230	2.120	0.270	1.610	0.430	2.110	0.910	1.079
n=20	*BIP*	CPU(s)	5.836	3.456	5.272	2.487	5.225	3.428	4.949	2.669	4.165
	SPT-LOE	Dev%	7.130	3.340	14.730	5.140	7.970	2.300	14.190	4.930	7.466
	SPCT-LOE	Dev%	0.930	0.190	1.540	0.870	1.700	0.270	1.940	0.590	1.004
	INDEX-LOE	Dev%	0.840	0.260	1.780	0.850	1.830	0.340	1.470	0.590	0.995
n=30	*BIP*	CPU(s)	611.558	100.617	72.447	28.458	513.117	162.396	58.759	28.555	196.988
	SPT-LOE	Dev%	2.350	0.960	5.190	2.240	3.100	0.990	5.010	1.950	2.724
	SPCT-LOE	Dev%	0.460	0.220	0.550	0.160	0.330	0.120	0.460	0.430	0.341
	INDEX-LOE	Dev%	0.410	0.180	0.610	0.130	0.350	0.110	0.380	0.220	0.299
n=40	*BIP*	CPU(s)	6791.530	6183.700	3210.660	1001.280	4216.980	6293.050	3361.640	691.010	3968.731
	SPT-LOE	Dev%	3.110	0.810	5.130	2.070	3.360	1.100	4.420	1.960	2.745
	SPCT-LOE	Dev%	0.320	0.130	0.640	0.210	0.500	0.130	0.620	0.210	0.345
	INDEX-LOE	Dev%	0.400	0.070	0.670	0.170	0.320	0.130	0.610	0.230	0.325
n=50	*BIP*	CPU(s)	6486.600	3070.820	4325.580	3490.970	7200.570	3267.340	7200.650	3775.840	4852.296
	SPT-LOE	Dev%	8.170	2.720	12.140	4.670	7.710	2.430	12.810	4.160	6.851
	SPCT-LOE	Dev%	1.590	0.360	1.680	0.550	1.020	0.650	1.230	0.760	0.980
	INDEX-LOE	Dev%	1.110	0.220	1.630	0.340	1.090	0.340	1.330	0.610	0.834

TABLE 4. The summary of each scenario in average.

Processing time distribution	Problem size	*SPT-LOE*	*SPCT-LOE*	*INDEX-LOE*
U[1,10]	Small	2.308%	0.317%	0.281%
	Large	1.671%	0.183%	0.296%
U[1,100]	Small	5.252%	0.741%	0.669%
	Large	6.385%	1.330%	1.088%

the slightly smaller the deviation of the heuristic over the optimum. Finally, γ controls the value of the dirt threshold. Figure 2 shows that the smaller the value of γ is, the slightly better the quality of the heuristic.

B. COMPARISON OF THE HEURISTICS WITH LOWER BOUND FOR LARGE PROBLEMS

We compare the three heuristics with the *Lower Bound* obtained by the IBM ILOG CPLEX Optimization Studio Ver. 12.6.1 for large instances with $n = 100$ and 200. The computational results are shown in Tables 5 and 6 for the processing time distributions $U(1,10)$ and $U(1,100)$, respectively. The deviations of the three heuristics over the lower bound are also summarized in Table 4.

The results in Tables 5 and 6 indicate that the average CPU times for IBM ILOG CPLEX Optimization Studio Ver. 12.6.1 to obtain the lower bound are 472.122 seconds and 573.373 seconds when $n = 100$ and 200, respectively.

TABLE 5. The results of $p_i = U[1, 10]$ for large problems.

	p_i[1,10]	α	0.2				0.4				
		β	2		4		2		4		
		γ	1.5	2.5	1.5	2.5	1.5	2.5	1.5	2.5	Ave.
n=100	*Lower Bound*	CPU(s)	308.306	330.367	366.852	366.703	601.448	601.323	601.225	600.750	472.122
	SPT-LOE	Dev%	0.930	0.380	2.250	0.810	2.160	0.870	4.500	2.100	1.750
		CPU(s)	0.001	0.001	0.001	0.000	0.001	0.001	0.001	0.001	0.001
	SPCT-LOE	Dev%	0.090	0.170	0.110	0.070	0.220	0.230	0.280	0.370	0.193
		CPU(s)	0.001	0.000	0.002	0.000	0.002	0.001	0.005	0.000	0.001
	INDEX-LOE	Dev%	0.130	0.080	0.150	0.100	0.470	0.510	0.500	0.480	0.303
		CPU(s)	0.003	0.000	0.005	0.001	0.004	0.000	0.006	0.000	0.002
n=200	*BIP*	CPU(s)	490.600	600.744	491.933	600.728	600.834	600.716	600.730	600.697	573.373
	SPT-LOE	Dev%	0.800	0.200	2.220	0.680	2.140	4.550	0.730	1.410	1.591
		CPU(s)	0.010	0.005	0.009	0.006	0.017	0.019	0.008	0.007	0.010
	SPCT-LOE	Dev%	0.070	0.150	0.070	0.140	0.150	0.310	0.220	0.280	0.174
		CPU(s)	0.005	0.002	0.037	0.003	0.005	0.056	0.003	0.004	0.014
	INDEX-LOE	Dev%	0.060	0.090	0.100	0.240	0.360	0.530	0.410	0.530	0.290
		CPU(s)	0.034	0.003	0.075	0.012	0.043	0.125	0.004	0.014	0.039

TABLE 6. The results of p_i= U[1, 100] for large problems.

	p_i[1,100]	α	0.2				0.4				
		β	2		4		2		4		
		γ	1.5	2.5	1.5	2.5	1.5	2.5	1.5	2.5	Ave.
n=100	*Lower Bound*	CPU(s)	601.581	600.192	602.559	600.241	602.094	600.253	602.292	600.275	601.186
	SPT-LOE	Dev%	7.550	2.590	11.710	5.000	8.200	2.780	11.390	5.180	6.800
		CPU(s)	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	SPCT-LOE	Dev%	1.560	1.010	1.690	1.190	2.180	1.470	1.620	1.550	1.534
		CPU(s)	0.005	0.001	0.010	0.002	0.005	0.001	0.007	0.002	0.004
	INDEX-LOE	Dev%	1.480	0.750	1.680	0.960	1.580	0.920	1.800	1.270	1.305
		CPU(s)	0.004	0.001	0.009	0.002	0.002	0.001	0.009	0.001	0.004
n=200	*BIP*	CPU(s)	601.050	600.703	601.095	600.797	601.088	600.705	601.125	600.739	600.913
	SPT-LOE	Dev%	6.390	2.170	11.080	3.710	7.210	2.360	11.040	3.800	5.970
		CPU(s)	0.001	0.001	0.001	0.002	0.001	0.001	0.002	0.001	0.001
	SPCT-LOE	Dev%	1.420	0.670	1.010	0.780	1.540	1.280	1.330	0.980	1.126
		CPU(s)	0.083	0.007	0.174	0.027	0.080	0.011	0.174	0.023	0.072
	INDEX-LOE	Dev%	0.950	0.490	1.060	0.590	1.230	0.610	1.270	0.760	0.870
		CPU(s)	0.092	0.011	0.197	0.041	0.087	0.013	0.183	0.038	0.083

The instance of the widely dispersed job processing time distribution $U(1,100)$ needs more execution time and incurs greater solution deviation. The execution times of all three heuristics are less than 0.039 seconds and 0.083 seconds for the processing time distributions $U(1,10)$ and $U(1,100)$, respectively, when $n = 200$. With regard to the qualities of the heuristics, the average deviations of three heuristics, *SPT-LOE*, *SPCT-LOE* and *INDEX-LOE*, over the lower bound are 1.671%, 0.183%, and 0.296%, respectively, for the processing time distribution $U(1,10)$. For the processing time distribution $U(1,100)$, the deviations of *SPT-LOE*, *SPCT-LOE* and *INDEX-LOE*, over the optimal solution are 6.385%, 1.330%, and 1.088% on average, respectively. Since the lower bound value instead of the optimal solution is used to evaluate the performances of the heuristics, the actual values of the deviation of the heuristics from the optimal solution will be lower than those shown in Tables 4, 5 and 6.

The influences of the two processing time distributions associated with the three heuristics on the solution quality are shown in Figure 3. From Figure 3 and Table 4, it is clear that *INDEX-LOE* outperforms the other two heuristics, especially *SPT-LOE*, which is commonly used in practice. The impacts of the parameters α, β and γ on the solution quality are analyzed and depicted in Figure 3.

The influences of the parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ on the solution quality are shown in Figure 3. From Figure 3, we observe that the deviation of the heuristic from the lower bound increases as the value of $\boldsymbol{\alpha}$ increases, and the larger the value of $\boldsymbol{\beta}$ is, the smaller the deviation of the heuristic over the lower bound. Finally, the smaller the value of $\boldsymbol{\gamma}$ is, the slightly better

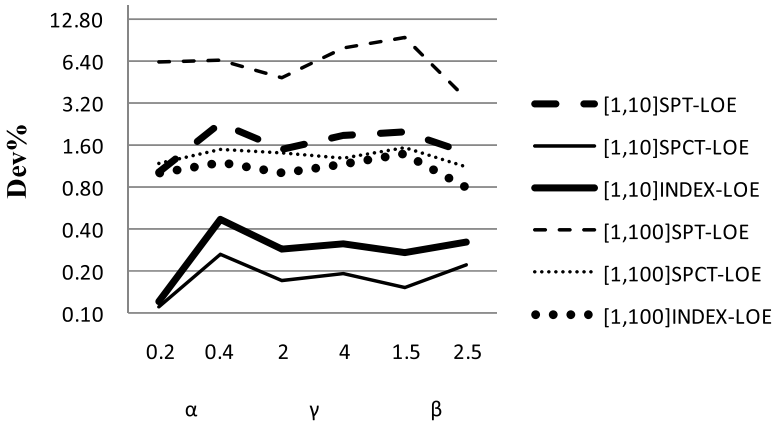

FIGURE 3. The influences of processing time distribution, α, β, and γ on the solution quality for large problems.

the quality of the heuristic. Although the three parameters $\boldsymbol{\alpha}$, $\boldsymbol{\beta}$, and $\boldsymbol{\gamma}$ have impacts on the solution quality of the heuristics, the influence is rather small. Therefore, we can assure that the heuristics are robust and pragmatic.

VI. CONCLUSION

This study focuses on the problem of minimizing the total completion time on a single machine with nonresumable jobs and machine unavailability. This problem is motivated by a wafer fabrication process, where the machine has to be stopped to remove the accumulated dirt with a cleaning agent avoiding damage to the quality of a wafer. The objective is to minimize the total completion times.

The considered problem is *NP-hard*. Many properties for optimally solving the problem are developed. Based on these properties, an MBIP model is implemented to find optimal solutions with the commercial solver of IBM ILOG CPLEX Optimization Studio Ver. 12.6.1, and a lower bound and three simple and effective heuristics are also proposed. From the computational results with problem sizes $n \leq 50$, the best heuristic *INDEX-LOE* is 0.281% and 0.669% on average over the optimal solution for the processing time distributions $U(1,10)$ and $U(1,100)$, respectively. When $n = 100$ and 200, the *INDEX-LOE* is 0.30% and 1.088% on average over the lower bound for the processing time distributions with $U(1,10)$ and $U(1,100)$, respectively. The complexities of the heuristic are very low, but the algorithms can quickly obtain a near-optimal or optimal schedule to satisfy the quick response requirement in a real world environment.

The main contribution of this paper is that another type of machine unavailability that stems from a common practice in IC manufacturing industry, which has seldom been discussed in the scheduling literature, is considered. Another direction for future research is to extend the problem to other machine environments, such as identical or unrelated parallel machines. Otherwise, considering a due date related objective function is important because meeting due dates are also a concern in practical situations. Finally, developing a meta-heuristic to solve the multiple machine problem will be worthwhile.

REFERENCES

[1] P. L. Michael, *Scheduling: Theory, Algorithms, and Systems*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1995.
[2] D.-L. Yang, C.-L. Hung, C.-J. Hsu, and M.-S. Chern, "Minimizing the makespan in a single machine scheduling problem with a flexible maintenance," *J. Chin. Inst. Ind. Eng.*, vol. 19, no. 1, pp. 63–66, 2002.
[3] E. Sanlaville and G. Schmidt, "Machine scheduling with availability constraints," *Acta Informatica*, vol. 35, no. 9, pp. 795–811, 1998.
[4] X. Qi, T. Chen, and F. Tu, "Scheduling the maintenance on a single machine," *J. Oper. Res. Soc.*, vol. 50, pp. 1071–1078, Oct. 1999.
[5] G. Schmidt, "Scheduling with limited machine availability," *Eur. J. Oper. Res.*, vol. 121, pp. 1–15, Feb. 2000.
[6] C. J. Liao and W. J. Chen, "Single-machine scheduling with periodic maintenance and nonresumable jobs," *Comput. Oper. Res.*, vol. 30, pp. 1335–1347, Aug. 2003.
[7] J. Chen, "Single-machine scheduling with flexible and periodic maintenance," *J. Oper. Res. Soc.*, vol. 57, pp. 703–710, Jun. 2006.
[8] J.-S. Chen, "Scheduling of nonresumable jobs and flexible maintenance activities on a single machine to minimize makespan," *Eur. J. Oper. Res.*, vol. 190, pp. 90–102, Oct. 2008.
[9] M. Ji, Y. He, and T. C. E. Cheng, "Single-machine scheduling with periodic maintenance to minimize makespan," *Comput. Oper. Res.*, vol. 34, pp. 1764–1770, Jun. 007.
[10] J.-S. Chen, "Optimization models for the machine scheduling problem with a single flexible maintenance activity," *Eng. Optim.*, vol. 38, no. 1, pp. 53–71, 2006.
[11] G. Mosheiov and A. Sarig, "Scheduling a maintenance activity and due-window assignment on a single machine," *Comput. Oper. Res.*, vol. 36, pp. 2541–2545, Sep. 2009.
[12] C. Low, M. Ji, C.-J. Hsu, and C.-T. Su, "Minimizing the makespan in a single machine scheduling problems with flexible and periodic maintenance," *Appl. Math. Model.*, vol. 34, pp. 334–342, Feb. 2010.
[13] Y. Ma, C. Chu, and C. Zuo, "A survey of scheduling with deterministic machine availability constraints," *Comput. Ind. Eng.*, vol. 58, pp. 199–211, Mar. 2010.
[14] S.-L. Yang, Y. Ma, D.-L. Xu, and J.-B. Yang, "Minimizing total completion time on a single machine with a flexible maintenance activity," *Comput. Oper. Res.*, vol. 38, pp. 755–770, Apr. 2011.
[15] F. Zammori, M. Braglia, and D. Castellano, "Harmony search algorithm for single-machine scheduling problem with planned maintenance," *Comput. Ind. Eng.*, vol. 76, pp. 333–346, Oct. 2014.
[16] D. Xu, L. Wan, A. Liu, and D.-L. Yang, "Single machine total completion time scheduling problem with workload-dependent maintenance duration," *Omega*, vol. 52, pp. 101–106, Apr. 2015.

[17] Y. Yin, J. Xu, T. C. E. Cheng, C.-C. Wu, and D.-J. Wang, "Approximation schemes for single-machine scheduling with a fixed maintenance activity to minimize the total amount of late work," *Naval Res. Logistics*, vol. 63, pp. 172–183, Mar. 2016.
[18] Y. Yin, T. C. E. Cheng, and D.-J. Wang, "Rescheduling on identical parallel machines with machine disruptions to minimize total completion time," *Eur. J. Oper. Res.*, vol. 252, pp. 737–749, Aug. 2016.
[19] M. A. Kubzin and V. A. Strusevich, "Planning machine maintenance in two-machine shop scheduling," *Oper. Res.*, vol. 54, no. 4, pp. 789–800, 2006.
[20] G. Mosheiov and J. Sidney, "Scheduling a deteriorating maintenance activity on a single machine," *J. Oper. Res. Soc.*, vol. 61, pp. 882–887, May 2010.
[21] Y. Yin, W.-H. Wu, T. C. E. Cheng, and C.-C. Wu, "Due-date assignment and single-machine scheduling with generalised position-dependent deteriorating jobs and deteriorating multi-maintenance activities," *Int. J. Prod. Res.*, vol. 52, no. 8, pp. 2311–2326, 2014.
[22] W. Luo, L. Chen, and G. Zhang, "Approximation algorithms for scheduling with a variable machine maintenance," in *Algorithmic Aspects in Information and Management*, vol. 6124. 2010, pp. 209–219.
[23] D. Xu, Y. Yin, and H. Li, "Scheduling jobs under increasing linear machine maintenance time," *J. Scheduling*, vol. 13, pp. 443–449, Aug. 2010.
[24] M. Gu, X. Lu, J. Gu, and Y. Zhang, "Single-machine scheduling problems with machine aging effect and an optional maintenance activity," *Appl. Math. Model.*, vol. 40, pp. 8862–8871, Nov. 2016.
[25] W. Luo, T. C. E. Cheng, and M. Ji, "Single-machine scheduling with a variable maintenance activity," *Comput. Ind. Eng.*, vol. 79, pp. 168–174, Jan. 2015.
[26] H. Zhu, M. Li, and Z. Zhou, "Machine scheduling with deteriorating and resource-dependent maintenance activity," *Comput. Ind. Eng.*, vol. 88, pp. 479–486, Oct. 2015.
[27] W. Luo and F. Liu, "On single-machine scheduling with workload-dependent maintenance duration," *Omega*, vol. 68, pp. 119–122, Apr. 2017.
[28] K.-C. Ying, C.-C. Lu, and J.-C. Chen, "Exact algorithms for single-machine scheduling problems with a variable maintenance," *Comput. Ind. Eng.*, vol. 98, pp. 427–433, Aug. 2016.
[29] L.-H. Su and H.-M. Wang, "Minimizing total absolute deviation of job completion times on a single machine with cleaning activities," *Comput. Ind. Eng.*, vol. 103, pp. 242–249, Jan. 2017.
[30] J. R. Birge and K. D. Glazebrook, "Assessing the effects of machine breakdowns in stochastic scheduling," *Oper. Res. Lett.*, vol. 7, pp. 267–271, Dec. 1988.
[31] C.-Y. Lee and G. Yu, "Single machine scheduling under potential disruption," *Oper. Res. Lett.*, vol. 35, pp. 541–548, Jul. 2007.
[32] X. Xiong, D. Wang, T. C. E. Cheng, C.-C. Wu, and Y. Yin, "Single-machine scheduling and common due date assignment with potential machine disruption," *Int. J. Prod. Res.*, vol. 56, no. 3, pp. 1345–1360, 2018.
[33] Y. Yin, Y. Wang, T. C. E. Cheng, W. Liu, and J. Li, "Parallel-machine scheduling of deteriorating jobs with potential machine disruptions," *Omega*, vol. 69, pp. 17–28, Jun. 2017.

YARONG CHEN received the B.S. and M.S. degrees in management science and engineering from Jiangsu University, Zhenjiang, Jiangsu, China, in 2003. Since 2003, she has been an Assistant and an Associate Professor with the School of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou, China. Her research interests include the simulation and optimization of complicated manufacturing systems, the integration optimization of production planning and scheduling, scheduling algorithms, and lean production.

LING-HUEY SU received the B.S. degree from National Taiwan University, Taipei, Taiwan, in 1977, the M.S. degree in operational research from the Illinois Institute of Technology, Chicago, USA, in 1980, and the Ph.D. degree in industrial engineering from Yuan Ze University, Taoyuan, Taiwan, in 1998.

She was a Lecturer, from 1980 to 1998, and an Associate Professor, from 1998 to 2008, and has been a Full Professor, since 2008, with the Industrial Engineering Department, Chung Yuan Christian University, Taoyuan, where she was the Chairwoman, from 2011 to 2013. She has authored more than 45 articles. Her research interests include production scheduling, semiconductor manufacturing management, and production planning and control. She has been the Leader of more than 20 research projects funded by the Ministry of Science and Technology, since 1998.

YA-CHIH TSAI received the M.S. and Ph.D. degrees from the Department of Industrial Engineering and Management, Yuan Ze University. She is currently an Associate Professor with the Department of Hotel Management, Vanung University. Her current research interests include production scheduling and semiconductor manufacturing management.

SHENQUAN HUANG was born in 1986. He received the B.Sc. and Ph.D. degrees from Zhejiang University, in 2008 and 2013, respectively. He is currently a Lecturer with the College of Mechanical and Electrical Engineering, Wenzhou University, China. He has been involved in production planning, manufacturing servitization, computer-integrated manufacturing, and virtual enterprises for about ten years. He has published over 15 papers in scientific journals in the related areas.

FUH-DER CHOU received the M.S. degree in industrial engineering from Chung Yuan Christian University, in 1988, and the Ph.D. degree in industrial engineering and management from National Chiao Tung University, in 1997. He is currently a Professor with the College of Mechanical and Electronic Engineering, Wenzhou University, China. His research interests include production scheduling, semiconductor manufacturing management, and group technology.