Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 8891322, 15 pages
https://doi.org/10.1155/2021/8891322

Research Article

Single-Machine Scheduling with Fixed Periodic Preventive Maintenance to Minimise the Total Weighted Completion Times

Hongming Zhou,[1] Ya-Chih Tsai,[2] Shenquan Huang,[1] Yarong Chen,[1] and Fuh-Der Chou[1]

[1]*The College of Mechanical and Electrical Engineering, Wenzhou University, Wenzhou 325035, Zhejiang, China*
[2]*Department of Hotel Management, Vanung University, Tao Yuan, Taiwan*

Correspondence should be addressed to Fuh-Der Chou; fdchou@tpts7.seed.net.tw

Received 11 September 2020; Revised 23 March 2021; Accepted 26 March 2021; Published 17 April 2021

Academic Editor: Bekir Sahin

The single-machine scheduling problem with fixed periodic preventive maintenance, in which preventive maintenance is implemented periodically to maintain good machine operational status and decrease the cost caused by sudden machine failure, is studied in this paper. The adopted objective function is to minimise the total weighted completion time, which is representative of the minimisation of the global holding/inventory cost in the system. This problem is proven to be NP-hard; a position-based mixed integer programming model and an efficient heuristic algorithm with local improvement strategy are developed for the total weighted completion time problem. To evaluate the performances of the proposed heuristic algorithms, two new lower bounds are further developed. Computational experiments show that the proposed heuristic can rapidly achieve optimal results for small-sized problems and obtain near-optimal solutions with tight average relative percentage deviation for large-sized problems.

1. Introduction

Production scheduling is concerned with the allocation of limited resources (machines or operators) to tasks in the manufacturing systems of enterprises to optimise certain objective functions such that their competitive position is maintained or improved in fast-changing markets. In traditional scheduling theory, it is usually assumed that machines can operate continuously during a scheduling planning horizon. However, uncertain machine breakdown or preventive maintenance activities cannot be ignored in the real world, as they cause the machines to be unavailable and production continuity to be slowed or stopped. As a result, the overall performance of the system will be decreased. Therefore, the constraint of machine unavailability taken into account during production scheduling is very important, and it makes scheduling problems more complex and realistic.

Preventive maintenance (PM) causes machine unavailability, as mentioned above, but it has gradually become a common practice in many companies as a priori maintenance measure to prevent potential malfunctions and critical nonavailability of a system. For this reason, many researchers and practitioners have gradually regarded production scheduling jointly with PMas a common and significant issue. In the literature, most scheduling models assume that the intervals between maintenance activities are known in advance and that maintenance time (unavailability time) is a constant. This assumption is reasonable in the case that PM is determined by engineering staff in advance. To date, such models have received considerable attention from researchers, and comprehensive reviews of this type of research have also been provided [1–4].

Regarding the scheduling problem with PM activities, the single-machine problem with a single interval of machine nonavailability is a basic model, denoted as$1, h_1 \| \mathrm{Obj}$, where h_1 indicates the PM activity occurs once in the planning horizon, and *Obj* is the objective function adopted in scheduling problems. Adiri et al. [5] and Lee and Liman

[6] proved the single-machine problem with the total completion time (denoted as $1, h_1 \| \sum C_i$) to be NP-hard. They showed that the SPT rule has a tight worst-case error bound of 2/7. For the same problem, Sadifi et al. [7] showed their proposed approximation algorithm MSPT (modified SPT) had a worst-case error bound of3/17. Breit [8] provided a parametric O ($n\log n$)-algorithm H and the best error bound of this algorithm was 7.4%. Another related flow-time objective function is the weighted completion time, which also has been discussed in the literature, because weights can measure the importance of processing jobs, or quantify the stocking cost per a unit of time of jobs in a system. Kacem et al. [9] studied the problem of $1, h_1 \| \sum w_i C_i$. They proposed a branch-and-bound (BAB) method, a mixed integer programming model (MIP), and a dynamic programming (DP) method. For the same problem, Kacem and Paschos [10] modified the WSPT rule slightly and provided a polynomial $(3 - \sqrt{5})/2$-differential algorithm.

All the research cited above addressed cases with a single PM activity in the planning horizon. Without loss of generality, it is necessary to consider multiple PM activities, as maintenance is scheduled periodically in many manufacturing systems. That is, there is usually more than one maintenance period in the planning horizon. Liao and Chen [11] extended a single PM activity to multiple and periodic maintenance activities for a single-machine scheduling problem; they proposed a BAB algorithm and an efficient heuristic for minimising the maximum tardiness. For the objective of minimising the number of tardy jobs, Chen [12] proposed an efficient heuristic based on Moore's algorithm [13] and the BAB method to solve the problem. Lee and Kim [14] proposed a two-phase heuristic algorithm for the same problem to minimise the number of tardy jobs. Batun and Azizoglu [15] proposed a BAB algorithm to solve the same problem, in terms of total flow time. To minimise the makespan, Perez-Gonzalez and Framinan [16] developed an MIP model and two-phase heuristic algorithm based on bin-packing dispatching rules to obtain optimal or near-optimal solutions. Zhou et al. [17] also proposed different heuristic algorithms with a local improvement mechanism for the same problem.

In the related research, a variety of production scheduling problems exist that consider different maintenance types in different manufacturing shop floors. Relative to fixed PM activity with known starting time and duration, flexible maintenance is another attractive research topic where the maintenance start time is not fixed in advance. In some cases of steel processing or semiconductor manufacturing, the maintenance start time depends on a specific threshold value such as maximum continuous working time or maximum dirt. This kind of assumption can be found in Qi et al. [18], Mosheiov and Sarig [19], Cui and Lu [20], Su and Wang [21], Pang et al. [22], Huang et al. [23], Chung et al. [24], etc. Another deterministic case is when the maintenance period $[u, v]$ is arranged in advance and maintenance operations should be undertaken within the period $[u, v]$, where the maintenance time (w) is not longer than the maintenance period, i.e., $w \leq (v - u)$. Some contributions under this assumption for different scheduling problems can be found in Chen [25], Yang et al. [4], etc. All studies mentioned above assumed that the maintenance time was a constant. More recently, some researchers have taken variable maintenance time into account, because in some realistic applications the maintenance time either increases or decreases depending on the machine conditions or the maintenance skill improvement. Bock et al. [26] considered the case of job-dependent machine deterioration, where the maintenance time is dependent on the amount by which the maintenance level is increased; that is, more improvement of the machine's state requires more maintenance time. Several recent studies have been conducted to address variable maintenance activity on scheduling problems [27, 28].

This study focuses on scheduling jobs on a single machine with periodic unavailability periods due to implementing PM activities; the objective is to minimise the total weighted completion times. In our study, the unavailability periods are fixed and known in advance. In the literature, a recent study by Krim et al. [29] considered the same scheduling problem. Our contribution lies in developing a different MIP model, two lower bounds, and a new heuristic algorithm while examining the effectiveness of the proposed methods in terms of various problems. In the following section, we propose an MIP model and some lower bounds, which are compared with the results by Krim et al. [29]. The results show that our model finds the optimal solution in all problems of the model by Krim et al. [29] at a lower CPU time, and some lower bounds are addressed in Section 3. In Section 4, we propose a heuristic algorithm based on the analytical properties. The proposed heuristic algorithm is examined on large instances, and the results are compared and reported in Section 5. Finally, Section 6 states our conclusions along with future research directions.

2. Mixed Integer Programming Model

In this section, we first describe the considered problem. There are n independent nonresumable jobs to be processed on a single machine. PM activities are carried out periodically, so the machine is not constantly available in the planning horizon, which is composed of an availability period of $T - \delta$ time units followed by a nonavailability period of δ time units. A nonresumable job refers to a job that cannot be finished before a maintenance activity and thus has to restart. Each availability period of $T - \delta$ is frozen, and machine idle may occur because of job nonresumablity, as shown in Figure 1. Using the three-field notation of Graham et al. [30], we denote this scheduling problem as $1, h_k|nr| \sum w_i C_i$, which has been proven to be NP-hard [29]. For the $1, h_k|nr| \sum w_i C_i$ problem, some assumptions and notations are given as follows:

(i) Assumptions:

(1) All jobs and machine are available for time $t = 0$
(2) The processing times and weights of jobs are deterministic and known in advance

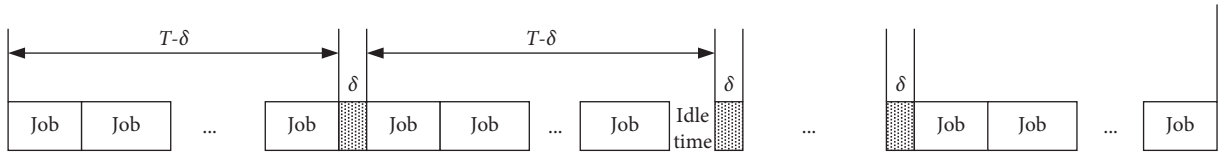

Figure 1: A feasible schedule for a single problem with fixed periodic maintenance activities.

(3) The machine can process only one job at a time without pre-emption

(4) For a machine, the maintenance time δ is deterministic and known in advance; after maintenance, the status of the machine becomes the initial status

(5) The fixed interval T is known in advance, and it is not large enough to process all the jobs (i.e., $\sum p_i > (T-\delta)$); however, the availability period of $(T-\delta)$ should be greater than or equal to the maximum processing time of jobs, i.e., $(T-\delta) \geq \max_{1\leq i\leq n}\{p_i\}$

(ii) Notation:

n: number of jobs

p_i: processing time of jobs i, $i = 1, 2, \ldots, n$

w_i: weight of jobs i, $i = 1, 2, \ldots, n$

δ: maintenance time

$T-\delta$: an available period, where T is a given fixed time period

M: a large integer number

X_{ik}: a binary decision variable used to judge whether job i is processed at position k, defined as follows:

$$X_{ik} = \begin{cases} 1 & \text{if job } i \text{ is scheduled at position } k \\ 0 & \text{otherwise} \end{cases}$$ where $i = 1, 2, \ldots, n$; $k = 1, 2, \ldots, n$

Y_k: a binary decision variable that is used to judge whether position k is assigned to execute maintenance, defined as follows:

$Y_k = \{1$ if maintenance is implemented at position k 0otherwise where $k = 1, 2, \ldots, n$

Q_k: sum of processing time at position k, $k = 1, 2, \ldots, n$

I_k: idle time after position k, $k = 1, 2, \ldots, n$

C_i: completion time of job i, $i = 1, 2, \ldots, n$

ST_k: start time at position k, $k = 1, 2, \ldots, n$

PT_k: processing time at position k, $k = 1, 2, \ldots, n$

CT_k: completion time at position k, $k = 1, 2, \ldots, n$.

Next, an MIP formulation is provided for the problem, and the model is constructed based on the viewpoint of each position. This is different from another MIP model that is built based on the relationship of jobs [29]. The proposed MIP model can be described as follows:

$$\text{minimise} \sum_{i=1}^{n} w_i C_i, \tag{1}$$

subject to

$$\sum_{k=1}^{n} X_{ik} = 1, \quad \forall i = 1, 2, \ldots, n, \tag{2}$$

$$\sum_{i=1}^{n} X_{ik} = 1, \quad \forall k = 1, 2, \ldots, n, \tag{3}$$

$$\mathrm{PT}_k = \sum_{i=1}^{n} (p_i \times X_{ik}), \quad \forall k = 1, 2, \ldots, n, \tag{4}$$

$$\mathrm{ST}_k \geq \mathrm{CT}_{k-1} + I_{k-1} + (\delta \times Y_{k-1}), \quad \forall k = 2, \ldots, n, \tag{5}$$

$$\mathrm{CT}_k \geq \mathrm{ST}_k + \mathrm{PT}_k, \quad \forall k = 1, 2, \ldots, n, \tag{6}$$

$$Q_1 = \sum_{i=1}^{n} (p_i \times X_{i1}), \tag{7}$$

$$Q_{k-1} + \sum_{i=1}^{n} (p_i \times X_{ik}) \leq Q_k + (M \times Y_{k-1}), \quad \forall k = 2, \ldots, n, \tag{8}$$

$$\sum_{i=1}^{n} (p_i \times X_{ik}) \leq Q_k + M \times (1 - Y_{k-1}), \quad \forall k = 2, \ldots, n, \tag{9}$$

$$Q_k + M \times (Y_{k-1} - 1) \leq \sum_{i=1}^{n} (p_i \times X_{ik}), \quad \forall k = 2, \ldots, n, \tag{10}$$

$$Q_k \leq (T - \delta), \quad \forall k = 1, 2, \ldots, n, \tag{11}$$

$$(M \times Y_k) - I_k \geq 0, \quad \forall k = 1, 2, \ldots, n, \tag{12}$$

$$M \times (1 - Y_k) - (T - \delta - Q_k - I_k) \geq 0, \quad \forall k = 1, 2, \ldots, n, \tag{13}$$

$$C_j + M \times (1 - X_{ik}) \geq \mathrm{CT}_k, \quad \forall j, k = 1, 2, \ldots, n, \tag{14}$$

$$\mathrm{ST}_1 = 0, \tag{15}$$

$$Y_n = 0, \tag{16}$$

$$I_n = 0, \tag{17}$$

Constraint (1) is the objective function that minimises the total weighted completion time of all jobs. Constraint (2) and constraint (3) ensure that each job can only be placed at one position and that each position can only be occupied by one job. Constraint (4) specifies the processing time of job that is assigned at position k. Constraint (5) ensures that the time for position k to assign a job should start no earlier than the sum of its predecessor's finish time, idle time, and possible maintenance time. Constraint (6) establishes a link between the start time and finish time for position k. Constraint (7) calculates the accumulated processing time of position 1. Constraints (8) to (10) calculate the accumulated processing time of position k, excluding position 1. Constraint (11) ensures that the accumulated processing time of position k in the sequence does not exceed the predefined available period. Constraints (12) and (13) are used to calculate the idle time immediately after position k. Constraint (14) is the completion time of each job. Constraint (15) specifies the starting time for position 1 when the sequence is equal to zero. Constraints (16) and (17) specify the maintenance time and idle time for the last position in the sequence is equal to zero, respectively.

In the proposed model, the number of constraints is $n^2 + (11 \times n) - 3$. The number of integer and binary variables is $6 \times n$ and $n^2 + n$, respectively, and n is the number of jobs. In the model proposed by Batun et al. [15] and mentioned above, the number of constraints is $2n^2 + (3 \times n)$. The number of integer and binary variables is n and $2n^2$, respectively. For even relatively modest problem sizes, there are too many integer variables in the two models for the problem to be solved; however, the MIP model is still an appropriate way to describe the problem complexity and obtain optimal solutions for a small-size problem. Form a practical viewpoint, heuristic algorithms are required to solve large-sized problem.

3. Lower Bounds

A lower bound value is an estimate of the best possible solution and is useful to evaluate the performance of heuristic algorithms when the optimal solutions cannot be found. Krim et al. [29] proposed four different lower bound procedures by polynomial relaxed problems. According to their research, we develop two different lower bound values for the considered problem in this paper. Before introducing our lower bound values, the four different lower bound procedures of Krim et al. [29] are briefly described as follows:

(i) LB1 $= \sum_{i=1}^{n} w_i \sum_{j=1}^{i} p_j$, which is based on the relaxed problem of $1// \sum_{i=1}^{n} w_i c_i$

(ii) LB2 $= p \sum_{i=1}^{n} i w_i + T \sum_{k=1}^{m-1} (k \sum_{i=1}^{n_{k+1}} w_{n_1+n_2+\cdots+n_k+i})$, which is based on the special case that the processing times of all jobs are the same

(iii) LB3 $= \sum_{i=1}^{m_1} w_i \sum_{j=1}^{i} p_j + \sum_{i=g}^{m_2} w_i (T + \sum_{j=g}^{i} p_j)$, which is based on the relaxed problem of $1, h_1// \sum_{i=1}^{n} w_i c_i$ and applies a dynamic programming algorithm to solve the $1, h_1// \sum_{i=1}^{n} w_i c_i$

(iv) LB4 $= \sum_{i=1}^{n} p_i w_i + T \times (m(m-1)/2)$, which is based on the special case that the weight values of all jobs are the same

For more detail of these lower bounds, please see the work of Krim et al. [29]. Among the four lower bounds, we find the formula of *LB2* to be questionable, and we use a counterexample of 8 jobs in which $p_j = p$, $w_j = w$, and $T = (3p + \delta)$ to illustrate the question.

For the above counterexample, the optimal sequence is as shown in Figure 2, and the optimal solution is $(15pw + 7Tw)$. However, according to the *LB2* formula where $\min_{1 \le i \le 8} p_i = p, w_i = w, m = (\sum p_i / 3p) = (8p/3p) = 3$ and x denotes the least integer greater than or equal to x, $n_1 = 3$, $n_2 = 3$, $n_3 = 2$; the calculation of *LB2* is as follows:

$$\begin{aligned} \text{LB2} = f^* &= \sum_{i \in B_1} w_i n_1 p + \sum_{i \in B_2} w_i (n_2 p + T) + \sum_{i \in B_3} w_i (n_3 p + 2T), \\ &= \sum_{i \in B_1} w \times 3 \times p + \sum_{i \in B_2} w(3p + T) + \sum_{i \in B_3} w(2p + 2T), \\ &= (22pw + 7Tw) \ge (15pw + 7Tw) \,(\text{optimal solution})\,(\text{it is conflict}). \end{aligned} \tag{18}$$

Regarding our proposed lower bound values, first we supposed an optimal sequence, as shown in Figure 3. Other notations used in Figure 3 are described below.

(i) m^*: number of batches for an optimal solution

(ii) B_k: kth batch, $k = 1, 2, \ldots, m^*$

(iii) n_k: number of jobs in the kth batch, $k = 1, 2, \ldots, m^*$

(iv) $\sum_{k=1}^{m^*} n_k = n$

(v) Δ_k: idle time occurred in the kth batch, $k = 1, 2, \ldots, (m^* - 1)$

(vi) $J_{[i]}$: job in the ith position of an optimal sequence

(vii) $p_{[i]}$: processing time of the job in the ith position of an optimal sequence

(viii) $w_{[i]}$: weight value of the job in the ith position of an optimal sequence

For the optimal sequence as shown in Figure 3, we can obtain the total weighted completion time as follows:

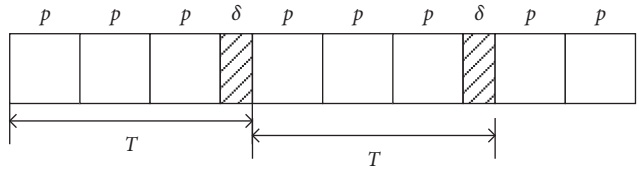

Figure 2: Optimal sequence of the counterexample.

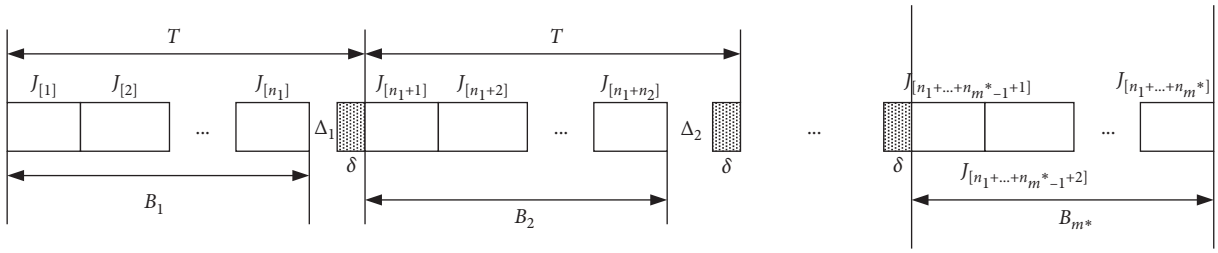

Figure 3: Schematic diagram of an optimal solution.

$$
\begin{aligned}
\mathrm{TWC} &= \mathrm{TWC}_{B_1} + \mathrm{TWC}_{B_2} + \cdots + \mathrm{TWC}_{B_{m^*}},\\
\mathrm{TWC}_{B_1} &= \sum_{i\in B_1} w_{[i]}C_{[i]} = p_{[1]}w_{[1]} + \left(p_{[1]} + p_{[2]}\right)w_{[2]} + \cdots + \left(p_{[1]} + p_{[2]} + \cdots + p_{[n_1]}\right)w_{[n_1]},\\
\mathrm{TWC}_{B_2} &= \sum_{i\in B_2} w_{[i]}C_{[i]} = \left(T + p_{[n_1+1]}\right)w_{[n_1+1]} + \left(T + p_{[n_1+1]} + p_{[n_2+1]}\right)w_{[n_1+2]}\\
&\quad + \cdots + \left(T + p_{[n_1+1]} + \cdots + p_{[n_1+n_2]}\right)w_{[n_1+n_2]}\\
&= \left(p_{[1]} + p_{[2]} + \cdots + p_{[n_1+1]} + \Delta_1 + \delta\right)w_{[n_1+1]}\\
&\quad + \left(p_{[1]} + p_{[2]} + \cdots + p_{[n_1+2]} + \Delta_1 + \delta\right)w_{[n_1+2]} + \cdots + \left(p_{[1]} + p_{[2]} + \cdots + p_{[n_1+n_2]} + \Delta_1 + \delta\right)w_{[n_1+n_2]},\\
\mathrm{TWC}_{B_3} &= \sum_{i\in B_3} w_{[i]}C_{[i]} = \left(p_{[1]} + p_{[2]} + \cdots + p_{[n_1+n_2+1]} + \Delta_1 + \Delta_2 + 2\delta\right)w_{[n_1+n_2+1]}\\
&\quad + \cdots + \left(p_{[1]} + p_{[2]} + \cdots + p_{[n_1+n_2+n_3]} + \Delta_1 + \Delta_2 + 2\delta\right)w_{[n_1+n_2+n_3]},\\
&\vdots\\
\mathrm{TWC}_{B_{m^*}} &= \sum_{i\in B_{m^*}} w_{[i]}C_{[i]}\\
&= \left[(m^*-1)T + p_{[n_1+n_2+\cdots+n_{m^*-1}+1]}\right]w_{[n_1+n_2+\cdots+n_{m^*-1}+1]} + \cdots\\
&\quad + \left[(m^*-1)T + p_{[n_1+n_2+\cdots+n_{m^*-1}+1]} + \cdots + p_{[n_1+n_2+\cdots+n_{m^*}]}\right]\\
w_{[n_1+n_2+\cdots+n_{m^*}]} &= \left[\left(p_{[1]} + p_{[2]} + \cdots + p_{\left[\left(\sum_{k=1}^{m^*-1} n_k\right)+1\right]} + \sum_{k=1}^{m^*-1}\Delta_k + (m^*-1)\delta\right]\right.\\
&\quad \left. w_{\left[\left(\sum_{k=1}^{m^*-1} n_k\right)+1\right]} + \cdots + \left[p_{[1]} + p_{[2]} + \cdots + p_{[n]} + \sum_{k=1}^{m^*-1}\Delta_k + (m^*-1)\delta\right]w_{[n]},\right.\\
\mathrm{TWC} &= \mathrm{TWC}_{B_1} + \mathrm{TWC}_{B_2} + \cdots + \mathrm{TWC}_{B_{m^*}} = \sum_{i=1}^{n} w_{[i]}\sum_{j=1}^{i} p_{[i]}\\
&\quad + \sum_{k=2}^{m^*}\Delta_{k-1}\sum_{b=1}^{n_k} w_{\left[\sum_{x=1}^{k-1} n_x + b\right]} + \delta\sum_{k=1}^{m^*}(k-1)\sum_{b=1}^{n_k} w_{\left[\sum_{x=1}^{k-1} n_x + b\right]}.
\end{aligned}
\tag{19}
$$

According to Figure 3 and the formula mentioned above, it is obvious that the number of batches (m^*) is an important factor influencing the schedule results, and the obvious estimation for the number of batches is $m' = \sum p_i/T - \delta$ [29]. In this paper, we adopt another lower bound concept provided by Martello and Toth [31] for the bin-packing problem to estimate the number of batches. We let $k_{\max} = T - \delta/2$, where $x$denotes the greatest integer less than or equal to x and $1 \le a \le k_{\max}$. We put the jobs where their processing times satisfy the condition of $p_i \ge (T - \delta - a)$ into set A. The other jobs where their processing times satisfy the condition of $p_i \le a$ are put into set B. We let N_A be the number of jobs in set A. Now, for each a, the waste time is calculated based on the formula of $\text{Waste}_a = \max[0, N_A (T - \delta) - \sum_{i\in A} p_i - \sum_{i\in B} p_i]$, and the maximum waste is obtained by $\text{Waste}_{\max} = \text{Max}_{1\le a\le k_{\max}}(\text{Waste}_a)$. Considering maximum waste, the estimation of the number of batches will be $m'' = \sum p_i + \text{Waste}_{\max}/T - \delta$. For m', m'', and m^*, the relation among them is $m' \le m'' \le m^*$.

For a job sequence $(J'_{[1]}, J'_{[2]}, \ldots, J'_{[n]})$ where all jobs are sorted according to WSPT rule, i.e., $(p_{[1]}'/w_{[1]}') \le (p_{[2]}'/w_{[2]}') \le \cdots \le (p_{[n]}'/w_{[n]}')$, we use the estimation (m'') as a substitute for the best number of batches (m^*) and ignore the impact of the total weighted idle time, i.e., $\sum_{k=2}^{m^*} \Delta_{k-1} \sum_{b=1}^{n_k} w_{[\sum_{x=1}^{k-1} n_x + b]}$ on the objective function, i.e., $\sum_{k=2}^{m^*} \Delta_{k-1} \sum_{b=1}^{n_k} w_{[\sum_{x=1}^{k-1} n_x + b]} = 0$. Then, the new lower bound (*NewLB*) is derived based on equation (18) as below, and the value of NewLB is less than or equal to the optimal solution (TWC).$\text{NewLB} = \text{TWC}' = \sum_{i=1}^{n} w_{[i]}' \sum_{j=1}^{i} p_{[i]}' + \delta \sum_{k=1}^{m''} (k-1) \sum_{b=1}^{n_k'} w_{[\sum_{x=1}^{k-1} n_x' + b]}' \le \text{TWC}$, where n_k' is the minimum number of jobs in B_k. The estimation of n_k' is as follows:

$$n_k' = \begin{cases} \dfrac{T - \delta}{p_{\max}}, & 1 \le k \le (m'' - 1), \\ \max\left(1, \dfrac{\sum p_j - (T - \delta)(m'' - 1))}{p_{\max}}\right), & k = m'', \end{cases} \tag{20}$$

where $p_{\max} = \max(p_i)$. To illustrate the proposed lower bound values, an instance with 5 jobs is given; the information for the instance is shown in Table 1, and the optimal solution of 194 is obtained by the MIP model.

First, $k_{\max} = (T - \delta/2) = (10 - 3/2) = 3$. We let a be equal to 1, and Set $A = \{\varnothing\}$, $N_A = 0$, since none of jobs satisfied the condition of $p_j \ge (10 - 3 - 1) = 6$, Set $B = \{\varnothing\}$. $\text{Waste}_1 = \max[0, \quad N_A (T - \delta) - \sum_{i\in A} p_i - \sum_{i\in B} p_i] = \max[0, 0 - 0 - 0] = 0$.

We let a be equal to 2, Set $A = \{J_1, J_2, J_3, J_4\}$, $N_A = 4$ since these jobs satisfied the condition of $p_j \ge (10 - 3 - 2) = 5$, Set $B = \{\varnothing\}$.$\text{Waste}_2 = \max[0, N_A (T - \delta) - \sum_{i\in A} p_i - \sum_{i\in B} p_i] = \max[0, (4 \times 7) - 20 - 0] = 8$. For a = 3, Set $A = \{J_1, J_2, J_3, J_4, J_5\}$, $N_A = 5$ since these jobs satisfied the condition of $p_j \ge (10 - 3 - 3) = 4$. Set $B = \{\varnothing\}$. $\text{Waste}_3 = \max[0, (5 \times 7) - 24 - 0] = 11$. The maximum waste time is$\max[0, 8, 11] = 11$, and the estimation of the number of batches is $m'' = (\sum p_j + \text{Waste}_{\max}/T - \delta) = (24 + 11/7) = 5$. For each batch, we can obtain the minimum number of jobs according to equation (19), which is $n_1' = n_2' = n_3' = n_4' = n_5' = 1$. Next, the jobs are sorted based on the WSPT rule, and *NewLB* is obtained as follows:

$\text{NewLB} = \sum_{i=1}^{n} w_{[i]}' \sum_{j=1}^{i} p_{[i]}' + \delta \sum_{k=1}^{m''} (k-1) \sum_{b=1}^{n_k'} w_{[\sum_{x=1}^{k-1} n_x' + b]}'$
$= 123 + 3 \times [(4 \times 1) + (3 \times 1) + (2 \times 2) + (1 \times 3)] = 123 + (3 \times 14) = 165$

Recalling *LB4*, the estimation is composed of $\sum_{i=1}^{n} p_i w_i$ and $T \times (m(m-1)/2)$; now replace$m$ with m''; we can derive $\text{LB5} = \sum_{i=1}^{n} p_i w_i + T \times (m''(m'' - 1)/2)$. For this example, $\text{LB5} = 54 + 10 \times (5 \times 4/2) = 154$. Table 2 lists the results obtained by the five lower bounds for this five-job example where the performance of lower bound is evaluated by the related percentage gap (*RPG*) from the optimal solution.

Table 1: An instance with 5 jobs, $T = 10$, and $\delta = 3$.

	J_1	J_2	J_3	J_4	J_5
Processing times (p_i)	5	5	5	5	4
Weights (w_i)	1	2	3	4	1

4. Heuristic Algorithms

Despite the relative success of the two MIP models mentioned above for finding optimal solution reporting in computational experiments, the models are still incapable of solving medium and large instances. It is necessary to develop efficient heuristic algorithms. Many existing heuristic algorithms in the literature use some specific optimality properties as a basis for constructing better schedule solutions [18, 32]. Additionally, some local improvement strategies such as insert and swap methods are integrated in heuristic algorithms to gain better solutions, especially when developing metaheuristic algorithms (Li et al. [33], Imran Hossain et al. [34]). Inspired by this idea, we adopt three important optimality properties proposed by Krim et al. [29] in this paper to construct a heuristic algorithm to find optimal or near-optimal solutions for large-sizes problems. Additionally, a local improvement strategy is combined with the heuristic algorithm.

First, the three important optimality properties addressed by Krim et al. [29] are briefly described as follows.

Property 1. In any optimal solution, the jobs are scheduled in increasing order of p_i/w_i (*WSPT* rule) in each batch.

Property 2. In any optimal solution, $W_{B_k} \ge W_{B_{k+1}} 1 \le k \le (m^* - 1)$ where $W_{B_k} = \sum_{i\in B_k} w_i$.

Property 3. In any optimal solution, for each idle time Δ_k in the batch B_k and J_h with smallest processing time in the batch B_l, $k + 1 \le l \le m^*$, the condition of $\Delta_k < p_h$ for $k \in \{1, \ldots, (m^* - 1)\}$ is always satisfied.

Krim et al. [29] proposed nine heuristic algorithms, and the basic framework of these algorithms contains the following three stages: (1) a sequence of jobs is generated by dispatching rules; (2) given the sequence of jobs, an initial

Table 2: Comparison of the five lower bounds according to the 5-job example.

	LB1	*LB3*	*LB4*	*LB5*	*NewLB*
Lower bound values	123	158	114	154	165
RPG (%)	36.598	18.557	41.237	20.619	14.948

schedule with *PM* is produced by batching rules; and (3) Properties 1 and 2 are applied to improve the initial schedule to obtain final schedule. The detailed steps of the heuristic algorithms can be found in the study by Krim et al. [29]. According to their computational results, the *WSPTBF* and *WSPTFF* algorithms had better performances among the nine algorithms. In WSPTBF and WSPTFF algorithms, the job sequence is generated by *WSPT* rule, whereas the initial schedule was, respectively, generated by best fit (*BF*) and first fit (*FF*) batching rules, which are well-known batching rules for solving bin-packing problems [32].

In this paper, we develop two-phase heuristic algorithm with framework similar to that of the heuristic algorithms of Krim et al. [29]. Phase 1 consists of a sorting step using a *WSPT* dispatching rule to determine a sequence of jobs, followed by assigning steps using a full-batching rule that determines the jobs in each batch (i.e., the interval of $T - \delta$). The full-batching rule is used widely to solve the bin-packing problem that tries to minimise the number of batches [32]. As a result, an initial schedule solution is obtained. Next, Phase 2 aims to improve the initial schedule by the local improvement strategy, which includes insert and swap procedures. The insert procedure attempts to satisfy Property 3 by inserting jobs to the forward batches. That is, if a job α, $\alpha \in B_{k+l}$ and batch B_k exist in which the condition of $(\sum_{i \in B_k} p_i) + p_\alpha \leq (T - \delta)$ is satisfied, then job α is removed from batch B_{k+l} and placed into the idle period of batch B_k. The detailed steps of the insert procedure are described as follows:(Algorithm 1)

The swap procedure attempts swapping jobs belonging to two different batches so that the current idle time is decreased or the total weights (W_k) of batch k are increased. After swapping two jobs, Property 1 still needs to be satisfied. The detailed steps of the swap procedure are described as follows.(Algorithm 2)

Finally, a complete heuristic algorithm combined with local improvement strategy including insert and swap procedures, named *WSPTFB + LIS,* is described below.(Algorithm 3)

A preliminary test is conducted using the 12-job instance of Krim et al. [29] to compare our proposed heuristic algorithm with *WSPTBF* and *WSPTFF*. Tables 3 and 4 show the information of the 12-job instance and the results obtained by the heuristic algorithms, respectively. In Table 4, the symbol * indicates the solution is also optimal. For this instance, *WSPTFF* had a better initial solution than the other algorithms, and the initial solutions obtained by the *WSPTBF* and *WSPTFB + LIS* algorithms are the same, equal to 11131; however, the *WSPTFB + LIS* algorithm had better final solution. This result occurred because, without violating the three optimality properties, our heuristic algorithm could reallocate jobs between batches to obtain a better schedule by the local improvement strategy. The other algorithms (*WSPTBF* and *WSPTFF*) only tried to re-sort the sequences of jobs (or batches) to improve the solution quality, and the content (jobs) of each batch did not change.

5. Computational Experiments

This section consists of three experiments. Experiment 1 focuses on comparing the computational efficiency of the two MIP models provided by Krim et al. [29] and ours. Experiment 2 aims to evaluate different lower bounds. Finally, the performances of the proposed heuristic algorithms are compared in Experiment 3. Since the benchmark data of Krim et al. [29] is not available at the following URL: https://sakai.uphf.fr/wiki/site/1b6900a5-38f1-4003-94e9-317898cbf168/instancesmaintenancewct.html, we generate a set of problems to carry out the computational experiments. The test-bed contains small-sized and large-sized problems. Small-sized problems consist of 5, 6, 7, . . ., 14, 15 jobs, and large-sized problems consider 20, 25, 30, 35, 40, . . ., 90, 100, 200, 300, 400, 500, 1000 jobs. The processing times of jobs from a discrete uniform distribution in the range [1, 5, 15, 20] are generated, respectively. The weight of jobs is generated from a uniform distribution in the range [1, 10]. T and $\delta$are set to [30, 50] and [3, 10], respectively. 10 instances were generated for each combination of problem size (n), processing time (p_i), T and δ; thus, a total of 2240 $(28 \times 2 \times 2 \times 2 \times 10)$ instances were randomly generated and tested. To make the generated instances be reasonable, the following two restricted conditions should be involved: (1) $\max(p_i) \leq (T - \delta)$; (2) $\sum p_i > (T - \delta)$. All experiments were conducted on a PC with an Intel Xeon E-2124 3.4 GHz CPU and 32 GB RAM.

Experiment 1. Comparison of the two MIP models

In this Experiment 1, we compare our model with the one provided by Krim et al. [29]. The two models are executed by IBM ILOG CPLEXOptimization Studio Version 12.7.1, and all problems are solved on the same machine for fair comparison. Additionally, the time limit of ILOG OPL is set to 7200 seconds for solving each instance. Table 5 reports the results for the two models, where MIP_k and MIP_c are, respectively, denoted as the model provided by Krim et al. [29] and our model. The "AveSol" and "AveLB" columns give the average the weighted completion time and average lower bound value over 80 instances, respectively. Another column, "*nOPT*," denotes the number of times each model finds the optimal solution. Since the MIP model may not find optimal solutions within the time limit of 7200 seconds, we record the lower bound values obtained by each MIP model to make sure the identified solutions are optimal, that is, $\text{Sol}(\text{MIP}) = \text{LB}(\text{MIP})$. For the MIP_k model, *nOPT* for 13-job instances is 79 (53), where 79 indicates the number of times each solution of MIP_k model is equal to lower bound values ofMIP_cmodel over 80 instances; furthermore, (53)

(i) Step 4: let $k = 1$ and $l = 1$
(ii) Step 4.1. If the condition of $(\sum_{i \in B_k} i \in B_k) + p_\alpha \leq (T - \delta)$ is satisfied, remove job α from batch B_{k+l} and place it into the idle period of batch B_k, let improve_flag = true, substitute the current schedule by the new one, go to Step 3
(iii) Otherwise, go to Step 4.2./ ∗ Property 3 check
(iv) Step 4.2. If $(k + l) < m$ is satisfied, let $l = l + 1$ go to Step 4.1; otherwise, go to Step 4.3
(v) Step 4.3. If $k < (m - 1)$ is satisfied, let $k = (k + 1)$, $l = 1$, go to Step 4.1; otherwise, go to Step 5

Algorithm 1: Insert procedure (Step 4 of Algorithm *WSPT + LIS*).

(i) Step 6: let $k = 1$ and $l = 1$.
(ii) Step 6.1: if there exist two jobs, $\alpha \in B_k$,$\beta \in B_{k+l}$, the conditions of $(\sum_{i \in [B_k \setminus \{\alpha\}]} p_i) + p_\beta \leq (T - \delta)$ and $(p_\alpha \times w_\alpha) < (p_\beta \times w_\beta)$ are satisfied, duplicate the current schedule π to schedule π', and go to Step 6.2. Otherwise, go to Step 6.4.
(iii) Step 6.2: job α in the position is interchangeable with job β in the position of schedule π' and resorting the jobs in batch B'_k and batch B'_{k+l} based on *WSPT* rule, respectively. Then, a new schedule (π_{new}) is generated; the corresponding objective value is TWC_{new}./ ∗ Property 1 check.
(iv) Step 6.3: if $\text{TWC}_{\text{new}} < \text{TWC}$ is satisfied, let $\pi = \pi_{\text{new}}$, $\text{TWC} = \text{TWC}_{\text{new}}$, improve_flag = true, go to Step 3. Otherwise, go to Step 6.4.
(v) Step 6.4: if $(k + l) < m$ is satisfied, let $l = (l + 1)$; go to Step 6.1; otherwise, go to Step 6.5.
(vi) Step 6.5: if $k < (m - 1)$ is satisfied, let $k = (k + 1)$, $l = 1$; go to Step 6.1; otherwise, go to Step 7.

Algorithm 2: Swap procedure (Step 6 of Algorithm *WSPT + LIS*).

(i) Step 1: input data.
(ii) Step 2: sort the jobs according to the *WSPT* rule, and assign the sorted jobs into each batch based on full-batching method. Let π, *TWC*, and improve_flag be the obtained schedule, the corresponding objective value, and true, respectively.
(iii) Step 3: if improve_flag = false, output the schedule π, *TWC*andStop. If improve_flag = true, go to Step 4.
(iv) Step 4: execute insert procedure.
(v) Step 5: if improve_flag = true, go to Step 3; otherwise, go to Step 6.
(vi) Step 6: execute swap procedure.
(vii) Step 7: if improve_flag = true, go to Step 3; otherwise, go to Step 8.
(viii) Step 8: compute $W_{B_k}, \forall k = 1, 2, \ldots, m$.
(ix) Step 9: if the schedule π satisfies Property 2, go to Step 3; otherwise, sort the batches in decreasing order of W_{B_k} and improve_flag = true; go to Step 3./ ∗ Property 2 check.

Algorithm 3: WSPTFB + LIS.

Table 3: Information of a 12-job instance of Krim et al. [29].

	J_1	J_2	J_3	J_4	J_5	J_6	J_7	J_8	J_9	J_{10}	J_{11}	J_{12}
p_i	6	15	11	17	7	13	18	11	4	10	18	5
w_i	10	14	10	14	19	11	14	18	12	19	15	15
						$T = 55$, $\delta = 1$						

Table 4: Results obtained by the heuristic algorithms.

	Initial solution	Final solution
WSPTBF	11131 (10526) $\{(J_9, J_{12}, J_5, J_{10}, J_1, J_8)(J_2, J_3, J_6)(J_{11}, J_4, J_7)\}$	10691 (10266) $\{(J_9, J_{12}, J_5, J_{10}, J_1, J_8)(J_{11}, J_4, J_7)(J_2, J_3, J_6)\}$
WSPTFF	9831 $\{(J_9, J_{12}, J_5, J_{10}, J_1, J_8, J_3)(J_2, J_6, J_{11})(J_4, J_7)\}$	9831 $\{(J_9, J_{12}, J_5, J_{10}, J_1, J_8, J_3)(J_2, J_6, J_{11})(J_4, J_7)\}$
WSPTFB + LIS	11131 $\{(J_9, J_{12}, J_5, J_{10}, J_1, J_8)(J_2, J_3, J_6)(J_{11}, J_4, J_7)\}$	9712* $\{(J_9, J_{12}, J_5, J_{10}, J_1, J_8, J_3)(J_2, J_{11}, J_4)(J_6, J_7)\}$

Remark: ( ) indicates that the value in the () proposed by Krim et al. [15] was somewhat questionable.

Table 5: Comparison results of the Krim model and our model.

n	MIP_k			MIP_c		
	AveSol	AveLB	Nopt	AveSol	AveLB	Nopt
5	637.763	637.763	80 (80)	637.763	637.763	80
6	916.075	916.075	80 (80)	916.075	916.075	80
7	1123.388	1123.388	80 (80)	1123.388	1123.388	80
8	1497.725	1497.725	80 (80)	1497.725	1497.725	80
9	1808.913	1808.913	80 (80)	1808.913	1808.913	80
10	2210.200	2210.200	80 (80)	2210.200	2210.200	80
11	2623.450	2623.450	80 (80)	2623.450	2623.450	80
12	3091.475	3091.475	80 (80)	3091.475	3091.475	80
13	3528.425	3446.013	79 (53)	3528.413	3528.413	80
14	4155.175	3469.963	77 (9)	4153.625	4153.625	80
15	4729.325	3369.713	76 (1)	4724.013	4724.013	80
20	7875.863	2942.800	20 (0)	7862.438	7622.038	33
25	12471.225	2464.225	10 (0)	12368.563	11775.163	15

means the number of times each solution of MIP_k model is equal to lower bound values of MIP_k model. From this table, we can observe that AveLB obtained by the proposed model (MIP_C) improves significantly as the number of problem size increases, which allows the MIP_C model to find better solutions within the time limit of 7200 seconds. Additionally, because the values of LB(MIP_c)are tighter than those of LB(MIP_k), using the condition of Sol(MIP_k) = LB(MIP_c), feasible solutions obtained by MIP_k model within the time limit of 7200 seconds can be proved to be optimal. For 15-job problem, Nopt value of MIP_k is increased from 1 to 76, as shown in Table 5.

Regarding computation times, the computational effort required by the MIP_k model is significantly higher than that of the MIP_c model, especially when the problem sizes increases. Table 5 and Figure 4 reveal that the efficiency of the MIP_c model is better than results obtained by the MIP_k (Krim et al. [29]). However, the two models cannot solve the NP-hard problems considered in this paper in a reasonable time, and the performances of the two models become worse as the number of jobs increases. The comparison results showed that the two models are computationally inefficient when the number of jobs is beyond 20. Therefore, it is necessary to develop heuristics to obtain near optimal solutions in reasonable time.

Experiment 2. Comparison of the lower bounds

This experiment aims to evaluate the performance of the lower bounds, so we compare the lower bounds values with the optimal solutions for small-size problems. For large-size problems, the different lower bound values are compared with the value of LB_{best}, where LB_{best} = max[LB1, LB3, LB4, LB5, NewLB]; that is, the related percentage gap (*RPG*) is adopted as follows:

RPG = RPG = [(Best − LB)/Best] ∗ 100% where Best indicates the best solutions obtained by the MIP_c model or LB_{best}.

Regarding the computational time, excluding *LB3*, the average computational time required by the lower bound methods, less than 1 second, is negligible. The *LB3* method applied the dynamic programming method to obtain a lower bound value for the special case in which only one PM activity exists; therefore, *LB3* requires much more computational time as the number of jobs increases. In our computational experiment, the average computational time required by *LB3* is approximately 1700 seconds for 1000 jobs. The computation time required by all lower bound methods excluding *LB*3 is very small; therefore, we do not report it here.

Tables 6–8 show the minimum, maximum, and average *RPG*, respectively, of the lower bounds relative to the optimal solutions over 80 instances in each set of small-sized problems. From Tables 6–8, we can find that the minimum *RPG*, maximum *RPG*, and average *RPG* of *NewLB* are considerably smaller than others. Additionally, the results also supported the result of Krim et al. [29] in which *LB1* and *LB3* are quite tight compared to *LB4*. The overall average *RPG* of *NewLB* is at 5.403% on the optimality gap. For large-sized problems, Table 9 shows that *NewLB* remains the best lower bound. From the results shown in Tables 6–9, we observe that the *NewLB* is quite superior to others. To that end, *NewLB* is used to compute the performances of the heuristic algorithms in Experiment 3.

Experiment 3. Comparison of the heuristic algorithms

In this experiment, our proposed heuristic algorithm is compared with the two heuristic algorithms provided by Krim et al. [29], and we adopt the *RPD* (*R*elative *P*ercentage *D*eviation) to measure the solution quality of each instance for the heuristic algorithms. *RPD* is defined as follows: RPD = (TWC_{method} − Min/Min) × 100%, where *Min* is the optimal or lower bound value provided by MIP_c model or *NewLB* method, and TWC_{method} is the total weighted completion time obtained by each heuristic algorithm. Tables 10 and 11 provide the results of the *WSPTFB* and *WSPTFB* + *LIS* algorithms, where *WSPTFB* represents the proposed heuristic algorithm without local improvement strategy. From Tables 10 and 11, algorithm *WSPTFB* + *LIS* give notable results by local improvement strategy in which *WSPTFB* + *LIS* has found the smaller average *RPD*, and the number of optimal solutions obtained is more than *WSPTFB*. Additionally, we also applied *ANOVA* analysis to

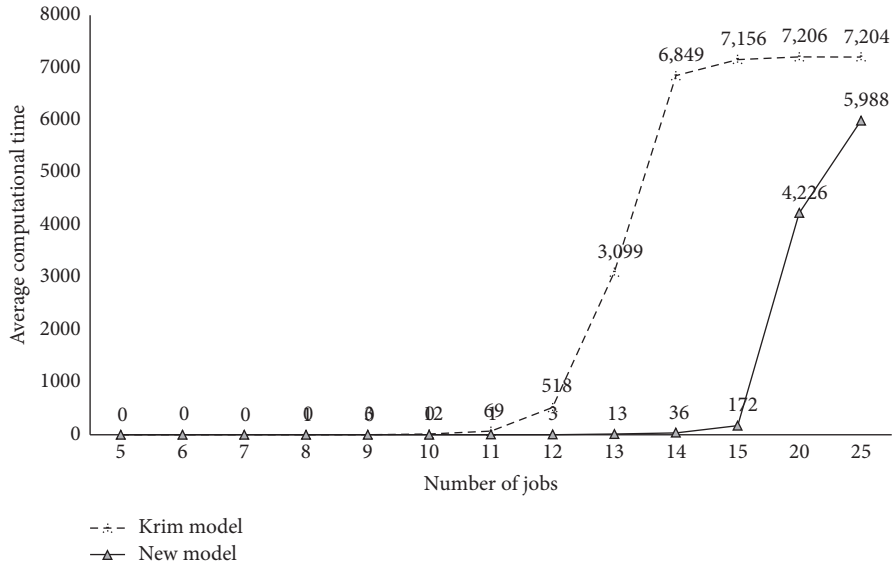

Figure 4: Average computational times required by the two models.

Table 6: Best performance for lower bounds compared to the optimal solutions.

n	Minimum RPG (%)				
	LB1	*LB3*	*LB4*	*LB5*	*NewLB*
5	0.588	0.000	28.016	24.307	0.000
6	0.612	0.000	38.782	26.099	0.000
7	0.992	0.000	44.013	44.013	0.000
8	0.539	0.000	49.653	36.025	0.000
9	0.676	0.000	51.385	49.497	0.000
10	1.585	0.000	49.195	38.452	0.000
11	1.374	0.527	53.663	50.815	0.000
12	2.260	0.888	60.482	55.103	0.000
13	0.463	0.463	50.210	42.344	0.000
14	1.718	0.000	61.760	55.880	0.000
15	3.064	1.000	64.293	51.660	0.180

Table 7: Worst performance for lower bounds compared to the optimal solutions.

n	Maximum RPG (%)				
	LB1	*LB3*	*LB4*	*LB5*	*NewLB*
5	42.424	34.294	68.235	62.199	0.000
6	42.620	34.304	66.683	64.976	22.162
7	41.723	34.467	71.655	68.750	17.949
8	48.660	41.910	76.703	71.564	20.907
9	46.659	39.082	75.132	74.592	30.586
10	43.398	37.131	78.237	77.441	22.613
11	45.370	39.557	77.747	77.747	24.820
12	40.329	32.909	78.523	78.523	20.986
13	41.050	35.127	78.993	78.993	24.603
14	38.368	29.149	80.156	80.156	22.007
15	42.546	37.534	80.655	80.655	26.603

examine the differences of the average *RPD* between the two heuristic algorithms. The results in Table 12 show that a statistically significant difference exists between the two algorithms, because p-value is less than 0.05. Based on the preliminary results, we focus only on comparing *WSPTFB* + *LIS* with other heuristic algorithms later.

To the best of our knowledge, the most recent and best algorithms proposed by Krim et al. [29] for solving the

Table 8: Average performance for lower bounds compared to the optimal solutions.

n	Average RPG (%)				
	LB1	*LB3*	*LB4*	*LB5*	*NewLB*
5	11.947	9.513	48.148	46.910	2.226
6	12.464	9.288	55.263	54.284	2.759
7	13.316	9.493	58.758	58.073	3.933
8	14.218	9.818	62.607	61.635	4.425
9	14.554	9.848	64.509	63.854	5.268
10	14.799	9.845	67.260	66.112	5.774
11	15.016	10.099	68.983	68.205	5.867
12	15.479	10.335	70.491	69.965	6.818
13	15.496	10.249	71.310	70.676	7.130
14	15.762	10.520	73.080	72.128	7.389
15	16.039	11.085	73.560	72.583	7.845
Overall average	14.463	10.008	64.906	64.039	5.403

Table 9: Average performance for lower bounds compared to the best lower bound.

n	*LB1*	*LB3*	*LB4*	*LB5*	*NewLB*
20	8.464	3.218	74.612	73.551	0.000
25	8.004	3.208	77.450	76.595	0.000
30	7.727	3.407	78.798	77.863	0.000
35	7.384	3.356	79.712	78.738	0.000
40	7.360	3.693	81.077	80.101	0.000
45	7.016	3.618	81.751	81.174	0.000
50	7.414	4.367	82.483	81.712	0.000
60	7.049	4.320	83.130	82.446	0.000
70	7.090	4.715	83.552	82.943	0.000
80	7.058	4.945	84.306	83.622	0.000
90	7.040	5.078	84.267	83.550	0.000
100	7.092	5.325	84.882	84.275	0.000
200	6.996	6.033	86.232	85.669	0.000
300	7.093	6.426	86.782	86.125	0.000
400	7.019	6.507	86.797	86.185	0.000
500	6.933	6.523	87.101	86.571	0.000
1000	7.088	6.879	87.315	86.666	0.000

Table 10: Results of *WSPTFB* and *WSPTFB* + *LIS* for small-sized problems.

n	*WSPTFB*				*WSPTFB* + *LIS*			
	Min	Max	Ave	Nopt	Min	Max	Ave	Nopt
5	0.00	39.49	3.83	53	0.00	10.87	0.22	77
6	0.00	39.41	5.11	43	0.00	11.18	0.46	72
7	0.00	37.77	5.94	34	0.00	20.81	1.18	65
8	0.00	40.74	7.07	28	0.00	13.37	1.01	58
9	0.00	33.53	8.31	19	0.00	19.81	1.48	60
10	0.00	35.07	8.98	14	0.00	22.05	2.01	54
11	0.00	36.81	10.33	13	0.00	22.93	1.91	44
12	0.00	39.06	10.85	7	0.00	18.07	1.92	35
13	0.00	29.94	9.77	7	0.00	19.13	1.51	43
14	0.00	35.57	10.62	6	0.00	13.94	2.38	26
15	0.00	39.78	10.89	2	0.00	25.27	2.30	18

$1, h_k|nr|\sum w_i C_i$ problem are the *WSPTBF* and *WSPTFF* algorithms. Thus, we adopt these two algorithms as benchmarks and compare the results with those of our *WSPTFB* + *LIS* algorithm. For fair comparison, *WSPTBF* and *WSPTFF* are recoded in C++ based on their pseudo codes (Krim et al. [29]), and the same test-bed problems are examined under the same computer configuration. In Table 13, we present the minimum, maximum, and average *RPD* values and the number of optimal solutions found by each heuristic algorithm over the 80 instances of each small-size problem. From this table, we can observe that *WSPTFB* + *LIS* is able to obtain better results in all cases (552

Table 11: Results of *WSPTFB* and *WSPTFB + LIS* for large-sized problems.

n	*WSPFFB*			*WSPTFB + LIS*		
	Min	Max	Ave	Min	Max	Ave
20	0.52	75.07	25.41	0.46	34.97	12.20
25	2.95	79.32	26.30	1.35	37.30	13.73
30	4.21	68.95	28.01	2.59	39.96	14.60
35	5.36	69.84	29.41	2.81	36.98	15.22
40	6.81	72.20	29.48	3.17	43.00	15.24
45	6.03	71.60	30.90	3.37	41.48	15.60
50	6.51	77.60	30.57	3.14	42.55	15.79
60	7.17	71.42	30.69	3.77	40.72	15.61
70	7.70	73.41	30.90	4.04	38.99	15.52
80	8.59	67.68	31.68	3.62	37.89	15.71
90	8.75	73.10	31.22	3.92	40.74	15.62
100	9.99	68.32	32.42	3.93	39.12	15.67
200	9.32	70.87	33.16	4.00	40.52	15.66
300	9.75	69.95	33.07	4.12	40.26	15.40
400	10.44	71.58	33.84	4.15	39.03	15.43
500	11.68	69.42	34.15	4.23	38.91	15.57
1000	12.10	70.66	34.53	4.24	38.03	15.25

Table 12: *ANOVA* results based on the averages of the proposed heuristic algorithms.

Source	Df	Sum of squares	Mean square	F	p value
Factor	1	2103.816	2103.816	23.436	<0.001
Error	54	4847.519	89.769		
Total	55				

Table 13: Results of the heuristic algorithms compared to the optimal solutions.

n	*WSPTBF*				*WSPTFF*				*WSPTFB + LIS*			
	Min	Max	Ave	Nopt	Min	Max	Ave	Nopt	Min	Max	Ave	Nopt
5	0.00	39.49	2.89	58	0.00	24.92	1.81	59	0.00	10.87	0.22	77
6	0.00	35.88	3.70	46	0.00	39.41	2.30	49	0.00	11.18	0.46	72
7	0.00	35.24	5.42	37	0.00	31.35	2.81	48	0.00	20.81	1.18	65
8	0.00	29.11	5.98	30	0.00	27.12	2.59	41	0.00	13.37	1.01	58
9	0.00	28.79	7.09	21	0.00	24.20	3.30	38	0.00	19.81	1.48	60
10	0.00	34.00	8.18	17	0.00	22.05	4.21	26	0.00	22.05	2.01	54
11	0.00	33.80	8.65	14	0.00	24.08	4.53	19	0.00	22.93	1.91	44
12	0.00	27.03	9.46	7	0.00	22.89	3.90	19	0.00	18.07	1.92	35
13	0.00	26.32	7.90	11	0.00	19.66	3.46	20	0.00	19.13	1.51	43
14	0.00	27.67	9.13	6	0.00	16.89	4.57	14	0.00	13.94	2.38	26
15	0.00	38.93	9.69	3	0.00	26.65	4.43	6	0.00	25.72	2.30	18

optimal solutions found over 880 instances). Additionally, the average *RPD* values of *WSPTFB + LIS* are considerably smaller than those of other heuristic algorithms, which is also seen in the results presented in Table 14 in which *WSPTFB + LIS* has smaller average *RPD* values for large-size problems. Table 15 also shows that there are significant differences on the average *RPD* values among the three algorithms based on the result of *ANOVA* analysis, where p-value is less than 0.05.

In terms of computation times, Table 16 shows the proposed *WSPTFB + LIS* for which the required computational effort is slightly higher than that for the others as the number of jobs increases. This is because it performs the local improvement strategy to find better solutions. Nonetheless, the proposed *WSPTFB + LIS* can obtain very good solutions within extremely small computational times as shown in the above results, where in Table 16 the average computational effort required by our proposed algorithm is less than 2 seconds for solving 1000 jobs. This indicates that our proposed *WSPTFB + LIS* algorithm outperforms the *WSPTBF* and *WSPTFF* algorithms, and the proposed algorithm can apply to real-world problems.

Table 14: Results of heuristic algorithms compared to the lower bounds.

n	*WSPTBF*			*WSPTFF*			*WSPTFB + LIS*		
	Min	Max	Ave	Min	Max	Ave	Min	Max	Ave
20	0.52	55.97	22.90	0.52	40.49	16.09	0.46	34.97	12.20
25	2.95	66.47	24.50	1.58	49.59	16.74	1.35	37.30	13.73
30	4.21	63.59	24.85	3.07	46.05	17.42	2.59	39.96	14.60
35	5.36	57.79	26.52	3.37	45.97	17.86	2.81	36.98	15.22
40	6.56	67.72	27.11	3.50	46.63	17.97	3.17	43.00	15.24
45	6.03	67.24	27.82	3.51	44.29	18.27	3.37	41.48	15.60
50	6.51	73.43	28.24	3.53	44.65	18.88	3.14	42.55	15.79
60	7.17	60.10	27.77	4.34	44.69	18.42	3.77	40.72	15.61
70	7.70	65.09	28.11	4.91	43.94	18.19	4.04	38.99	15.52
80	7.18	63.82	28.83	4.55	42.33	18.55	3.62	37.89	15.71
90	8.75	65.95	28.51	4.28	43.15	18.46	3.92	40.74	15.62
100	9.95	61.84	29.31	4.49	42.36	18.51	3.93	39.12	15.67
200	9.16	65.63	29.86	4.75	42.01	18.31	4.00	40.52	15.66
300	8.99	63.83	29.91	4.70	41.40	17.92	4.12	40.26	15.40
400	9.87	64.63	30.48	4.64	41.08	18.04	4.15	39.03	15.43
500	10.85	64.19	30.90	4.80	40.10	18.07	4.23	38.91	15.57
1000	11.44	65.04	31.06	4.72	40.21	17.79	4.24	38.03	15.25

Table 15: *ANOVA* results based on the averages among the three heuristic algorithms.

Source	Df	Sum of squares	Mean square	F	p value
Factor	2	1526.017	763.009	10.728	<0.001
Error	81	5761.038	71.124		
Total	83				

Table 16: Average computation time of the heuristic algorithms.

n	*WSPTBF*	*WSPTFF*	*WSPTFB + LIS*
5	0.00000	0.00000	0.00000
6	0.00000	0.00000	0.00000
7	0.00000	0.00000	0.00000
8	0.00000	0.00000	0.00000
9	0.00000	0.00000	0.00000
10	0.00000	0.00000	0.00000
11	0.00000	0.00000	0.00000
12	0.00000	0.00000	0.00000
13	0.00000	0.00000	0.00000
14	0.00000	0.00000	0.00000
15	0.00000	0.00000	0.00000
20	0.00000	0.00000	0.00001
25	0.00000	0.00000	0.00001
30	0.00000	0.00001	0.00002
35	0.00001	0.00001	0.00003
40	0.00001	0.00001	0.00004
45	0.00001	0.00001	0.00006
50	0.00001	0.00001	0.00013
60	0.00001	0.00002	0.00014
70	0.00002	0.00002	0.00024
80	0.00002	0.00003	0.00031
90	0.00002	0.00003	0.00046
100	0.00003	0.00004	0.00066
200	0.00010	0.00013	0.00513
300	0.00020	0.00026	0.02055
400	0.00032	0.00041	0.06735
500	0.00046	0.00060	0.13077
1000	0.00159	0.00211	1.54310

6. Conclusions

Scheduling problems with PM activity have received increasing attention in the last decade, due to the enthusiastic adoption of the PM philosophy in many real manufacturing systems. However, not much work has been performed for problems with the objective of total weighted completion time, denoted as $1, h_k|nr|\sum w_iC_i$. For the $1, h_k|nr|\sum w_iC_i$ problem, we proposed a new MIP model, two new lower bounds, and a heuristic algorithm. The proposed heuristic algorithm adopted a bin-packing scheme that considers batch formation and batch sequence and applied the optimality properties in the local improvement strategy to efficiently obtain optimal or near optimal solutions. To the best of our knowledge, a recent new study on the same problem was conducted by Krim et al. [29]. Thus, we took their proposed methods including the MIP model, lower bound methods, and better heuristic algorithms as the benchmark approaches.

Based on the results of a comprehensive comparison experiment with the benchmark approaches, in terms of MIP models, the number of optimal solutions found by our MIP model was more than that of the benchmark model within the same time limit (7200 seconds). Regarding lower bound methods, *NewLB* was significantly more effective than other methods, as the average *RPG* of *NewLB* was smaller (closer to optimal solutions/best lower bounds) than that of other methods. For solving small-sized problems, approximately 63% of our proposed heuristic solutions were found to be optimal, a ratio that is higher than that of the other algorithms (28% and 39% solutions found to be optimal for the *WSPTBF* and the *WSPTFF*, respectively). Furthermore, the results for *WSPTFF* + *LIS* were significantly better than the benchmark heuristic results for the average *RPD* values (reaching lower bound values) for larger instances. The average computation time of our heuristic algorithm was less than 2 seconds. These results show that our proposed heuristic algorithm is very efficient and has better performance with respect to problems with different dimensions.

In summary, the results presented in this paper are very encouraging regarding the use of the proposed MIP model and the *WSPTFF* + *LIS* algorithm for the single-machine total weighted completion time problem with periodic preventive maintenance. In future work, it will be worth considering different types of maintenance activities, such as variable maintenance duration or flexible maintenance activities, or extending this problem to parallel machines or, flow or job shop environments, as well as including different operation constraints. Additionally, another extension may be a consideration of other criterion such as total tardiness-earliness cost, and total weighted tardiness, which are also important objective functions for system performance measures.

Finally, benchmark data sets and solutions for our proposed methods including the MIP model, *NewLB*, and *WSPTFF* + *LIS* are made available at the following URL: https://drive.google.com/file/d/1Fh5S7j6pHGHhKuy2O6WcO0yfl2eMno-t/view?usp=sharing for researchers to compare their solutions.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 51705370 and 71501143) and the Zhejiang Province Natural Science Foundation of China (Grant nos. LY18G010012 and LY19G010007).

References

[1] G. Schmidt, "Scheduling with limited machine availability," *European Journal of Operational Research*, vol. 121, no. 1, pp. 1–15, 2000.

[2] C. Y. Lee, "Machine scheduling with availability constraints," in *Handbook of Scheduling*, J. Y.-T. Leung, Ed., pp. 22.1–22.13, CRC Press, Boca Raton, FL, USA, 2004.

[3] Y. Ma, C. Chu, and C. Zuo, "A survey of scheduling with deterministic machine availability constraints," *Computers & Industrial Engineering*, vol. 58, no. 2, pp. 199–211, 2010.

[4] 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," *Computers & Operations Research*, vol. 38, no. 4, pp. 755–770, 2011.

[5] I. Adiri, J. Bruno, E. Frostig, and A. H. G. Rinnooy Kan, "Single machine flow-time scheduling with a single breakdown," *Acta Informatica*, vol. 26, no. 7, pp. 679–696, 1989.

[6] C.-Y. Lee and S. D. Liman, "Single machine flow-time scheduling with scheduled maintenance," *Acta Informatica*, vol. 29, no. 4, pp. 375–382, 1992.

[7] C. Sadfi, B. Penz, C. Rapine, J. Błażewicz, and P. Formanowicz, "An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints," *European Journal of Operational Research*, vol. 161, no. 1, pp. 3–10, 2005.

[8] J. Breit, "Improved approximation for non-preemptive single machine flow-time scheduling with an availability constraint," *European Journal of Operational Research*, vol. 183, no. 2, pp. 516–524, 2007.

[9] I. Kacem, C. Chu, and A. Souissi, "Single-machine scheduling with an availability constraint to minimize the weighted sum of the completion times," *Computers & Operations Research*, vol. 35, no. 3, pp. 827–844, 2008.

[10] I. Kacem and V. T. Paschos, "Weighted completion time minimization on a single-machine with a fixed non-availability interval: differential approximability," *Discrete Optimization*, vol. 10, no. 1, pp. 61–68, 2013.

[11] C. J. Liao and W. J. Chen, "Single-machine scheduling with periodic maintenance and nonresumable jobs," *Computers & Operations Research*, vol. 30, no. 9, pp. 1335–1347, 2003.

[12] W. Chen, "Minimizing number of tardy jobs on a single machine subject to periodic maintenance☆," *Omega*, vol. 37, no. 3, pp. 591–599, 2009.

[13] S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, Ellis Horwood, Chichester, UK, 1982.

[14] J.-Y. Lee and Y.-D. Kim, "Minimizing the number of tardy jobs in a single-machine scheduling problem with periodic maintenance," *Computers & Operations Research*, vol. 39, no. 9, pp. 2196–2205, 2012.

[15] S. Batun and M. Azizoğlu, "Single machine scheduling with preventive maintenances," *International Journal of Production Research*, vol. 47, no. 7, pp. 1753–1771, 2009.

[16] P. Perez-Gonzalez and J. M. Framinan, "Single machine scheduling with periodic machine availability," *Computers & Industrial Engineering*, vol. 123, pp. 180–188, 2018.

[17] H. Zhou, Y.-C. Tsai, F.-C. Wu, S. Huang, and F.-D. Chou, "Improved approaches to minimize the makespan on single-machine scheduling with periodic preventive maintenance activities," *Mathematical Problems in Engineering*, vol. 2020, Article ID 8548463, 13 pages, 2020.

[18] X. Qi, T. Chen, and F. Tu, "Scheduling the maintenance on a single machine," *Journal of the Operational Research Society*, vol. 50, no. 10, pp. 1071–1078, 1999.

[19] G. Mosheiov and A. Sarig, "Scheduling a maintenance activity to minimize total weighted completion-time," *Computers & Mathematics with Applications*, vol. 57, no. 4, pp. 619–623, 2009.

[20] W.-W. Cui and Z. Lu, "Minimizing the makespan on a single machine with flexible maintenances and jobs' release dates," *Computers & Operations Research*, vol. 80, pp. 11–22, 2017.

[21] L.-H. Su and H.-M. Wang, "Minimizing total absolute deviation of job completion times on a single machine with cleaning activities," *Computers & Industrial Engineering*, vol. 103, pp. 242–249, 2017.

[22] J. Pang, H. Zhou, Y.-C. Tsai, and F.-D. Chou, "A scatter simulated annealing algorithm for the bi-objective scheduling problem for the wet station of semiconductor manufacturing," *Computers & Industrial Engineering*, vol. 123, pp. 54–66, 2018.

[23] S. Huang, H. Zhou, Y.-C. Tsai, Y. Chen, and F.-D. Chou, "Minimizing the total weighted completion time of a single machine with flexible maintenance," *IEEE Access*, vol. 7, pp. 122164–122182, 2019.

[24] T.-P. Chung, Z. Xue, T. Wu, and S. C. Shih, "Minimising total completion time on single-machine scheduling with new integrated maintenance activities," *International Journal of Production Research*, vol. 57, no. 3, pp. 918–930, 2019.

[25] J. S. Chen, "Single-machine scheduling with flexible and periodic maintenance," *Journal of the Operational Research Society*, vol. 57, no. 6, pp. 703–710, 2006.

[26] S. Bock, D. Briskorn, and A. Horbach, "Scheduling flexible maintenance activities subject to job-dependent machine deterioration," *Journal of Scheduling*, vol. 15, pp. 565–578, 2012.

[27] S.-J. Yang and D.-L. Yang, "Minimizing the makespan on single-machine scheduling with aging effect and variable maintenance activities," *Omega*, vol. 38, no. 6, pp. 528–533, 2010.

[28] H.-T. Lee, D.-L. Yang, and S.-J. Yang, "Multi-machine scheduling with deterioration effects and maintenance activities for minimizing the total earliness and tardiness costs," *The International Journal of Advanced Manufacturing Technology*, vol. 66, no. 1-4, pp. 547–554, 2013.

[29] H. Krim, R. Benmansour, D. Duvivier, D. Aït-Kadi, and S. Hanafi, "Heuristics for the single machine weighted sum of completion times scheduling problem with periodic maintenance," *Computational Optimization and Applications*, vol. 75, no. 1, pp. 291–320, 2020.

[30] R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. H. G. R. Kan, "Optimization and approximation in deterministic sequencing and scheduling: a survey," *Annals of Discrete Mathematics*, vol. 5, pp. 287–326, 1979.

[31] S. Martello and P. Toth, "Lower bounds and reduction procedures for the bin packing problem," *Discrete Applied Mathematics*, vol. 28, no. 1, pp. 59–70, 1990.

[32] R. Uzsoy, "Scheduling a single batch processing machine with non-identical job sizes," *International Journal of Production Research*, vol. 32, no. 7, pp. 1615–1635, 1994.

[33] K. Li, S.-L. Yang, and H.-W. Ma, "A simulated annealing approach to minimize the maximum lateness on uniform parallel machines," *Mathematical and Computer Modelling*, vol. 53, no. 5-6, pp. 854–860, 2011.

[34] S. Imran Hossain, M. A. H. Akhand, M. I. R. Shuvo, N. Siddique, and H. Adeli, "Optimization of university course scheduling problem using particle swarm optimization with selective search," *Expert Systems With Applications*, vol. 127, pp. 9–24, 2019.