Minimizing Maximum Tardiness in Single Computer Numerical Control Machine Scheduling with Tool Changes

Ming-Cheng Lo, Jen-Shiang Chen and Yi-Chien Chen

Department of Business Administration, Ching Yun University, Taiwan, Republic of China

Department of Business Administration, Far East University, Taiwan, Republic of China

Department of Management Information System, Far East University, Taiwan, Republic of China

Abstract: This study considers the problem of scheduling a set of jobs on a single Computer Numerical Control (CNC) machine where the cutting tool is subject to wear, aiming to minimize the maximum tardiness. Four mixed binary integer programming models were developed to solve this problem optimally. To solve large-sized problems, two heuristics were also developed. This study presents the computational results to demonstrate the efficiency of the models and the effectiveness of the heuristics.

Key words: Scheduling, single CNC machine, tool changes, availability constraints, integer programming, heuristics

INTRODUCTION

When scheduling jobs on single machine, engineers have always assumed that the machine is continuously available. In practice, however, the machine may be shut down for various reasons, e.g., for preventive maintenance or tool changes. Preventive maintenance is often discussed in literature, but tool changes are discussed less often. For preventive maintenance activities, related literature includes (Lee, 1996; Schmidt, 1988, 2000; Sanlaville and Schmidt, 1998; Graves and Lee, 1999; Blazewicz et al., 2000; Liao and Chen, 2003; Cassady and Kutanoglu, 2003).

Fluctuations in market demands over recent decades imply that manufacturing strategy has switched from high-volume production of narrow product lines to medium-to low-volume batches of many different products. Computer Numerical Control (CNC) is a programmable automation tool that efficiently accommodates product variations and small batch production. Tool change is essential issue in CNC tool management. Previous literature on tool management has addressed this issue, but assumed that the change is due only to the part mix. In practice, however, most tool changes are caused by tool wear (Akturt et al., 2003). This study considers the problem of scheduling a set of jobs on a single CNC machine where the cutting tool is subject to wear.

Liao and Chen (2003) studied a single machine scheduling problem requiring periodic maintenance in a complete schedule. Their study attempted to minimize the maximum tardiness. However, they assumed that each maintenance activity is fixed and known in advance. That is, they studied scheduling with multiple maintenance intervals and fixed time periods between consecutive maintenance activities.

Qi et al. (1999) studied scheduling with multiple maintenance intervals and variable time between consecutive maintenance activities. Their study attempted to minimize total job completion time. Their model is similar to scheduling with tool changes problem as several tool changes can occur over a given time period and the time between tool changes can vary. Qi et al. (1999) showed that their proposed problem was strongly NP-hard and presented three heuristics and a branch and bound algorithm to solve the problem. Akturt et al. (2003) studied the same problem for scheduling with tool changes to minimize total completion time. They provided a Mixed Integer Programming (MIP) model for the exact solution of the problem and proposed several heuristics based on simple dispatch rules and genetic search. Akturt et al. (2004) considered the same problem as did and briefly described the problem and discussed its properties, complexity and solution. They provided theoretical worst-case bounds on the performance of the Shortest Processing Time first (SPT) list-scheduling heuristic and also demonstrated its empirical behavior.

This study considers a single CNC machine scheduling problem in which processing is interrupted...
due to tool wear and tool changes and in doing so, attempts to minimize the maximum tardiness. The scheduling problem has multiple maintenance intervals and varying times between consecutive maintenance activities.

**PROBLEM DESCRIPTION AND NOTATION DEFINITION**

The notation used throughout the study is defined as:

**Symbol definition**
- \( I_i \): Job number \( i \)
- \( J_k \): The job placed at the \( k \)th position
- \( B_q \): Batch number \( q \)

**Input parameters**
- \( M \): A very large positive number
- \( n \): Number of jobs for processing at time zero
- \( p_i \): The processing time of \( I_i \)
- \( d_i \): The due date of \( I_i \)
- \( T_L \): Tool life
- \( T_C \): Tool change time

**Decision variables**
- \( b \): The number of batches required for processing \( n \) jobs (only for the heuristics)
- \( h_i \): The starting time of the job in the sequence position \( j \) (only for Model 1)
- \( f_i \): The finish time of the job in the sequence position \( j \) (only for Model 1)
- \( s_i \): The earliest start time of \( I_i \) (only for Model 2)
- \( C_i \): Completion time of \( I_i \) (only for Models 1 and 2)
- \( b_i \): The total batch processing time, in which the last job in the sequence position \( j \) and the immediately following position \( j \) the tool change must be taken; that is, if \( k_j = 1 \), then \( b_j \) denotes the time between the completion time of the batch in which the last job is in the sequence position \( j \) and the completion time of the previous tool change. Otherwise, if \( k_j = 0 \) then \( b_j = 0 \) (only for Models 3 and 4)
- \( e_i \): The elapsed time between the completion time of the last new tool change and the completion time of the job in sequence position \( j \) (only for Model 4)
- \( k_j \): 1 if tool is replaced immediately following position \( j \) 0 otherwise (only for Models 3 and 4)
- \( r_i \): The remaining time between the completion time of the job placed in the sequence position \( j \) and the latest starting time of the next tool change (only for Model 3)

\[ T_i \] \( \text{Tardiness of } I_i \); where \( T_i = \max \{0, C_i - d_i\} \)

\[ T_{\text{max}} \] \( \text{Max } \{T_i\} \)

\[ x_i \] 1 if \( I_i \) is scheduled at position \( j \), 0 otherwise (only for Models 1, 3 and 4)

\[ z_{ij} \] 1 if \( I_i \) precedes \( I_j \) (not necessarily immediately); 0 otherwise (only for Model 2)

Assume that \( n \) independent jobs \( I_1, I_2, ..., I_n \) are to be processed on a single CNC machine. All jobs are available at time zero and no pre-emption is allowed. The job processing times are constant and known a priori. Only one tool with a known, constant life and an unlimited availability is required. When an active tool wears out, it is replaced with a new one; the time needed for this tool change is also known and constant. The machine must stop and change tool after continuously working for a period of time. The maximum allowed continuously working time of the machine is \( T_C \) and the tool change time is \( T_C \). The relation \( T_i \geq p_i \) \( (i = 1, 2, ..., n) \) is assumed since otherwise no feasible schedule is available.

Minimizing the maximum tardiness is a basic objective studied in the scheduling literature (French, 1982). The EDD (Earliest Due Date) dispatching rule is well-known to produce an optimal schedule in the single machine case if the tool life is considered infinitely long (i.e., no tool change occurs). However, the structure of the problem changes dramatically when tool changes are introduced. The performance of the EDD rule depends on the value of \( T_C \). Notably, the EDD rule here assigns jobs to successive tools in non-decreasing order of their due dates and change tools when no current tool can perform a job.

If the jobs are considered as sharing the same tool as a batch, then a schedule can be viewed as a series of batches of jobs separated by tool changes (Fig. 1). Notably, the batch lengths may vary, because tools are changed when the current tool cannot handle the next job in the sequence. The batch length only shows the portion of the tool which has been used.

**Theorem 1:** The problem of single CNC machine scheduling with tool changes and the maximum tardiness as the criterion is strongly NP-hard.

**Proof:** Clearly the problem is strongly NP-hard since a problem that minimizes the maximum tardiness subject

![Fig. 1: Representation of a schedule as batch of jobs](image-url)
to multiple unavailability intervals and fixed start maintenance time is strongly NP-hard (Liao and Chen, 2003).

**INTEGER PROGRAMMING MODELS**

Mathematical programming formulation is a natural way to solve machine scheduling problems (Rinooy Kan, 1976). Most mathematical programming formulations of scheduling problems involve mixed Binary Integer Programming (BIP) in which some variables are binary and the others are continuous. The new development of mixed BIP techniques, along with the substantial progress in computer capacity, strongly impacts mixed BIP scheduling models. In this section, four mixed BIP models are provided for solving the proposed problem. These four models include Models 1-4.

**Model 1:** All n jobs processed in the symbol are assumed to be $J_1, J_2, \ldots, J_n$. The tool life $T_i$ is assumed to exceed the processing time of any job. Consequently, a maximum of (n-1) tool changes can occur in the planning horizon. Suppose the (n-1) tool changes in the symbol are $T_{i+1}, T_{i+2}, \ldots, T_{n+1}$. Models 1 and 2 are based on the concept of (n-1) tool changes can occur.

The binary variable $x_i$ used by Model 1 is restricted and specifies the order in which the machine processes jobs. Model 1 employs one-job-one-position to describe the single CNC machine scheduling with tool changes.

\[
\begin{align*}
\text{Minimize} & \quad T_{\text{max}} \\
\text{s.t} & \quad \sum_{i=1}^{n} x_i = 1, i = 1, 2, \ldots, n \quad (2) \\
& \quad \sum_{i=n+1} T_{i} x_i - n^2 \geq 1, j = 2, 3, \ldots, n-1 \quad (3) \\
& \quad \sum_{i=n+1} x_i = 1 \quad j = 2, 3, \ldots, n-1 \quad (4) \\
& \quad \sum_{i=n+1}^{y} x_i \leq \text{int}(y) \quad \text{where, int}(y) \text{ denote the greatest integer less than or equal to } y \quad (5) \\
& \quad x_i = 0 \quad i = n + 1, n + 2, \ldots, 2n-1; \\
& \quad j = 1, 2, \ldots, (i-n) \quad (6) \\
& \quad x_i = 0 \quad i = n + 1, n + 2, \ldots, 2n-2; \\
& \quad j = i + 1, i + 2, \ldots, 2n-1 \quad (7) \\
& \quad h_i + \sum_{j=1}^{n} p_{i,j} x_j = f_i \quad (8) \\
& \quad h_i + \sum_{j=1}^{n} p_{i,j} x_j + T_{j} \sum_{j=n+1}^{2n-1} x_j = f_i \quad (9) \\
& \quad f_{j,1} \leq T_{i} + M (1-x_{i-1,1}) \\
& \quad j = 2, 3, \ldots, 2n-2 \quad (10) \\
& \quad f_{i,j} \leq T_{i} + M (1-x_{i,j}) \\
& \quad i = n + 2, n + 3, \ldots, 2n-1; \\
& \quad j = 2 (i-n), 2 (i-n) + 1, \ldots, i; \\
& \quad j' = 2 (i-1+n), 2 (i-1+n) + 1, \ldots, i-1 \\
& \quad \text{and } j > j' \quad (12) \\
& \quad f_{j,1} \leq C_{i} + M (1-x_{i,j}) \quad i = 1, 2, \ldots, n; \\
& \quad j = 1, 2, \ldots, 2n-1 \quad (13) \\
& \quad C_{i} \cdot d_{i} \leq T_{\text{max}} - i, 2, \ldots, n \quad (14) \\
& \quad T_{\text{max}} \geq 0; h_i \geq 0; f_{i} \geq 0 \\
& \quad j = 1, 2, \ldots, 2n-1; C_{i} \geq 0 \\
& \quad i = 1, 2, \ldots, n; x_{i} \text{ is binary} \\
& \quad i, j = 1, 2, \ldots, 2n-1 \quad (15)
\end{align*}
\]

Constraint (1) describes the objective function, constraint sets (2) to (5) state in which each job is uniquely scheduled in a position for processing. Constraint sets (6) and (7) show that the situation of the tool changes must not be placed in any particular position. Constraints sets (8) and (9) are essentially definitional, while constraints (10) enforce the precedence relationships. Additionally, constraint sets (11) to (13) give the tool change starting and finishing times. Constraint set (14) defines the tardiness of jobs. Finally, the non-negativity and binary restrictions on $T_{\text{max}}, h_i, f_i$ and $C_i$ and $x_i$, respectively, are specified in (15).

**Model 2:** This model uses the binary variable $z_i$ to express the "either-or" relationship for the non-interference restrictions. Moreover, Model 2 describes the single machine problems using the concept of non-interference.

\[
\begin{align*}
\text{Minimize} & \quad T_{\text{max}} \\
\text{s.t} & \quad s_i + p_i = C_i, i = 1, 2, \ldots, n \quad (17) \\
& \quad s_i + T_{0} = C_i \\
& \quad i = n + 1, n + 2, \ldots, 2n-1 \quad (18)
\end{align*}
\]
Constraint sets (17) and (18) define the completion time of job, while constraint (16) defines the total tardiness. Constraint set (19) defines the tardiness of jobs. Moreover, constraint sets (20) and (21) meet the requirement that only one job can be processed at any time, that is, either $C_i < s_i$ or $C_i < s_i$ will hold. Incorporating binary variable $z_i$ and a very large positive number $M$, Eq. 20 and 21 together ensure that one of these two constraints holds while the other is eliminated. Furthermore, constraint sets (22) and (23) state the maintenance interval. Finally, constraint set (24) specifies the non-negativity of $C_i$, $s_i$, and $T_{max}$ and establishes the binary restrictions for $z_i$.

**Model 3:** A new binary variable $k_i$ is defined to be equal to 1 if the tool is replaced after position $j$, and 0 otherwise. The variable $r_i$ tracks the remaining time between the completion time of the job placed in the sequence position $j$ and the latest starting time of the next tool change. Model 3 uses one-job-one-position with $n$ positions to describe single CNC machine problems.

\[
\text{Minimize } T_{\text{max}} \quad (25)
\]

\[
\text{S.t. } \sum_{i=1}^{n} x_i = 1 \quad i = 1, 2, \ldots, n \quad (26)
\]

\[
\sum_{i=1}^{n} x_i = 1 \quad j = 1, 2, \ldots, n \quad (27)
\]

\[
r_i + \sum_{i=1}^{n} p_i x_i = T_e \quad (28)
\]

\[
r_j + \sum_{i=1}^{n} p_i x_i \leq T_e + M(1-k_j) \quad j = 2, 3, \ldots, n \quad (29)
\]

\[
r_j + \sum_{i=1}^{n} p_i x_i \leq T_e + M(1-k_j) \quad j = 2, 3, \ldots, n \quad (30)
\]

\[
b_j \geq T_e - t_j - M(1-k_j) \quad j = 1, 2, \ldots, n-1 \quad (31)
\]

\[
f_j = \sum_{i=1}^{n} b_j \quad j = 1, 2, \ldots, n \quad (32)
\]

\[
f_j \geq \sum_{i=1}^{n} b_j \quad j = 1, 2, \ldots, n \quad (33)
\]

\[
T_{\text{max}} \geq 0; r_i \geq 0; C_i, s_i, T_e \geq 0 \quad (34)
\]

Constraint (25) describes the objective function, constraint sets (26) and (27) state in which each job is uniquely scheduled in a position for processing. Constraint (28) records the remaining life time of the tool which processes the job in the first position. Additionally, constraint sets (29) and (30) meet the requirement that the range of $r_i$ that is, either

\[
r_j + \sum_{i=1}^{n} p_i x_i \leq T_e \quad (31)
\]

will hold. Incorporating binary variable $k_j$, and a very large positive number $M$, Eq. 29 and 30 together ensure that one of these two constraints holds while the other is eliminated. Moreover, constraint sets (31) to (33) define the $b_j$, $f_j$, and $T_{max}$ respectively. Finally, constraint set (34) specifies the non-negativity of $T_{max}$, $r_j$, $f_j$ and $b_j$, and establishes the binary restrictions for $x_j$ and $k_j$.

**Model 4:** Model 4 utilizes the variable $e_j$ instead of $r_j$ as used in Model 3. The variable $e_j$ tracks the time elapsed between the completion time of the last new tool change and the completion time of the job in sequence position $j$.

\[
\text{Minimize } T_{\text{max}} \quad (35)
\]

\[
\text{S.t. } \sum_{j=1}^{n} x_j = 1 \quad i = 1, 2, \ldots, n \quad (36)
\]

\[
\sum_{j=1}^{n} x_j = 1 \quad j = 1, 2, \ldots, n \quad (37)
\]
\[
e_i = \sum_{j=1}^{n_i} p_j x_j
\]

(38)

\[
e_i + \sum_{j=1}^{n_i} p_j x_j \leq e_i + M k_i
\]

(39)

\[
\sum_{j=1}^{n_i} p_j x_j \leq M (1-k_i)
\]

(40)

\[
e_j \geq T_{ij}, j = 2, 3, \ldots, n
\]

(41)

\[
b_j \geq e_j - M (1-k_i)
\]

(42)

\[
f_i = \sum_{j=1}^{n_i} b_j + e_j + T_{ij} \sum_{j=1}^{n_i} x_j
\]

(43)

\[
f_i \sum_{j=1}^{n_i} x_j \leq T_{max}
\]

(44)

\[
T_{max} \geq 0, e_i \geq 0, f_i \geq 0, j = 1, 2, \ldots, n;
\]

\[
b_j \geq 0, j = 1, 2, \ldots, n-1;
\]

\[
x_j \text{ is binary}
\]

\[
i = 1, 2, \ldots, n, j = 1, 2, \ldots, n;
\]

\[
k_i \text{ is binary}
\]

\[
j = 1, 2, \ldots, n-1;
\]

(45)

### Table 1: The size of integer programming model

<table>
<thead>
<tr>
<th>Model</th>
<th>No. of binary variables</th>
<th>No. of constraints</th>
<th>No. of continuous variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>(4n^2 - 4n + 3)</td>
<td>(7/2 n^2 + 11/2 n)</td>
<td>5n-l</td>
</tr>
<tr>
<td>2</td>
<td>(2n^2 - 3n + 1)</td>
<td>(3n^2 - 2n - 1)</td>
<td>4n-l</td>
</tr>
<tr>
<td>3</td>
<td>(n + n - 1)</td>
<td>(7n - 1)</td>
<td>3n</td>
</tr>
<tr>
<td>4</td>
<td>(n^2 + n - 1)</td>
<td>(8n - 2)</td>
<td>3n</td>
</tr>
</tbody>
</table>

### COMPARISONS OF THE PROPOSED MODELS

Size complexity denotes the size of a problem in terms of binary variables, constraints and continuous (real) variables as a function of \(n\), the number of jobs, in the problem. Table 1 shows the size complexity of each of the four integer programming models.

French (1982) stated that the speed with which integer programming problems can be solved depends on the number of variables and constraints in the problem, where the most influential factor is the number of binary variables. If two formulations have the same number of binary variables (Wilson, 1989; Liao and You, 1992) demonstrated that the next most influential element is the number of constraints. As Table 1 shows, Model 4 has the same number of binary and continuous variables as that of Model 3, but has \(n-1\) more constraints. Model 1 has \(2n^2 - 2\) more number of binary variables than Model 2, the former has \((-1/2) n^2 + (15/2) n + \sum_{j=1}^{n} \sum_{j=1}^{n} k_i j - 3\) more constraints and \(n\) more continuous variables. Models 3 and 4 are superior to Models 1 and 2 in terms of number of binary variables, constraints and continuous variables. Consequently, Model 3 is theoretically the best, followed in order by Model 4, Model 2 and Model 1.

### THE PROPOSED HEURISTICS

Although the integer programming approach can solve the problem being considered, large problems are difficult to solve without considerable computational efforts. Therefore, this study proposes two heuristic algorithms. Some properties of the optimal schedule of the proposed problem are first introduced.

**Theorem 2:** (Optimal sequence in the same batch) For two adjacent jobs in the same batch, an optimal schedule exists for which the job with the smaller due date is placed before the other job.

**Proof:** The result follows immediately from the EDD order.

Let \(b\) denote the number of batches required for processing all jobs. \(B_b\) denotes batch number \(q\), where \(q = 1, 2, \ldots, b\). The batch number is also the batch-forming sequence.
Theorem 3: (Feasible interchanging) Assume that $T_k$ is the maximum tardiness of a schedule, where $T_k > 0$ (i.e., $C_k > d_k$) and $I_k$ is in $B_k$. $B_k$ is not the first batch in the schedule. Denote by $J_i$ a job in the batch preceding $B_k$. Let $BT_i$ and $BT_i'$ be the total processing time of the batch which contains $I_k$ and $J_i$, respectively. If $BT_i - p_i - T_i > 0$ and $BT_i' - p_i' - T_i' > 0$, then a feasible schedule can be obtained by interchanging $I_k$ with $J_i$.

Proof: The proof is straightforward.

To reduce the maximum tardiness of a schedule, two jobs placed in two batches can be interchanged.

Theorem 4: (Interchange with preceding job) Suppose that $T_k$ is the maximum tardiness of a schedule, where $T_k > 0$ (i.e., $C_k > d_k$) and $I_k$ is in $B_k$. $B_k$ is not the first batch in the schedule. Denote by $J_i$ a job in the batch preceding $B_k$. Suppose that $I_k$ is not placed in $B_k$ and between $I_k$ and $J_i$ in a feasible schedule, if the following two conditions can be satisfied simultaneously, then a smaller $T_{\max}$ value can be obtained by interchanging $I_k$ with $J_i$.

(i) $C_k > d_k$ and $d_k > d_i$ or $C_k > d_i$ and $C_i < C_k$

(ii) $C_i + p_i > d_i$ or $C_i + p_i > d_i$ or $C_i + p_i > d_i$

(iii) $C_i + p_i > d_i$ and $p_i > d_i$

Proof: See Fig. 2, from Theorem 2 both jobs $I_k$ and $I_i$ are assumed to be feasibly interchangeable. Let schedule $S = (P_{S_b}, I_k, P_{S_a}, I_i, P_{S_b'}, P_{S_a})$, where $P_{S_b}, P_{S_a}, P_{S_b'}, P_{S_a}$ are partial schedules. Let $S'$ be obtained from $S$ by interchanging $I_k$ with $J_i$ that is, $S' = (P_{S_b}, I_i, P_{S_b'}, I_k, P_{S_a})$, where $P_{S_b}, P_{S_a}$ and $P_{S_b'}$ are partial schedules. Denote by $T_k(S)$ and $T_i(S')$ the tardiness of $I_k$ in schedule $S$ and $S'$, respectively (Fig. 2). Then the tardiness of $T_k(S)$, $T_i(S)$, $T_i(S')$, $T_k(S')$ and $T_i(S')$ are

$T_k(S) = \max\{C_k - d_k, 0\}$

$T_i(S) = \max\{C_i - d_i, 0\}$

$T_k(S') = \max\{C_k + p_k - d_k, 0\}$

$T_i(S') = \max\{C_i + p_i - d_i, 0\}$

To reduce the value of $T_{\max} = T_k(S)$, the following three cases must be satisfied simultaneously: (1) $T_k(S') \leq T_k(S)$; (2) $T_i(S') \leq T_i(S)$ or $T_i(S') \leq T_i(S)$; (3) $T_i(S') \leq T_i(S)$ or $T_i(S') \leq T_i(S)$. These three cases can be further discussed in detail as follows.

Case 1: $T_k(S') \leq T_k(S)$.

Obviously, $T_i(S') \leq T_i(S)$ must hold.

Case 2: $T_i(S') \leq T_i(S)$ or $T_i(S') \leq T_i(S)$.

Fig. 2: Illustration of interchange with preceding job

To satisfy $T_i(S') = \max\{C_i - d_i, 0\} \leq T_k(S) = \max\{C_k - d_k, 0\}$, the following two conditions must be satisfied.

(i) $C_i \geq d_i$ or (ii) $C_i \geq d_i$ and $d_i \leq d_i$

Then, to satisfy $T_i(S') = \max\{C_i - d_i, 0\} \leq T_i(S) = \max\{C_i - d_i, 0\}$, the following two conditions must also be satisfied.

(i) $C_i \geq d_i$, (ii) $C_i \geq d_i$, $C_i \geq d_i$

and $C_i \leq C_i$

Case 3: $T_i(S') \leq T_k(S)$ or $T_i(S') \leq T_i(S)$.

To meet $T_i(S') = \max\{C_i + p_k - d_i, 0\} \leq T_k(S) = \max\{C_k - d_k, 0\}$, the following two conditions must be met.

(i) $C_i + p_k - d_i \geq d_i$

and $C_i + p_k - d_i \geq d_i$

Then, to meet $T_i(S') = \max\{C_i + p_k - d_i, 0\} \leq T_i(S) = \max\{C_k - d_k, 0\}$, the following two conditions must also be met.

(i) $C_i + p_k - d_i \leq d_i$

and $p_k \leq d_i$

In summary the conditions (46-49) and the theorem is proved.

Corollary 1: As described in Theorem 4, if $J_i$ does not exist, that is, $J_i$ is in $B_k$ and $I_k$ placed the last position of the batch $B_k$. If the first condition (i) (that is, $C_i \leq d_i$, or $C_i \geq d_i$ and $d_i \leq d_i$, or $C_i \geq d_i$, and $C_i \leq C_i$) can be satisfied, then a smaller $T_{\max}$ value can be obtained by interchanging $I_k$ with $J_i$.

Proof: The proof is straightforward.

Based on the above properties, two heuristic algorithms are proposed. These two heuristics are
Heuristic H1 and Heuristic H2. Both heuristics utilize a two-phase method. Phase I focuses on forming batches and Phase II focuses on improving the maximum tardiness. $J_{[k]}$ denotes the job placed at the kth position and $p_{[k]}$, $d_{[k]}$, $c_{[k]}$ and $T_{[k]}$ are defined accordingly.

**Heuristic H1:** Phase I of Heuristic H1 first employs the EDD rule to sequence all jobs and then follows this sequence to form batches. The steps of Heuristic H1 are outlined as follows:

**Phase I**

- **Step 1:** Sequence all jobs according to EDD rule. Set $b = 1$, $B_0 = \emptyset$, $k = 1$, $TP = 0$ and $U = \{J_{[1]}, J_{[2]}, \ldots, J_{[n]}\}$.
- **Step 2:** Delete $J_{[k]}$ from $U$ and append $J_{[k]}$ to $B_k$.
- **Step 3:** If $k = n$, then stop.
- **Step 4:** If $TP = T_{[k]}$ then set $b = b + 1$ and $TP = 0$.
- **Step 5:** Set $k = k + 1$ and $TP = TP + p_{[k]}$.
- **Step 6:** If $TP > T_{[k]}$ then set $b = b + 1$ and $TP = p_{[k]}$.
- **Step 7:** Go to Step 2.

**Phase II**

- **Step 8:** Denote by $\sigma_s$ the resulting complete schedule, which is composed of the schedule in all batches and the necessary maintenance periods from Phase I.
- **Step 9:** If $T_i = 0$ for all $i$, stop, otherwise, find $J_i$ in $B_i$ with $T_{i_{max}}$ (if a tie exists, break it arbitrarily).
- **Step 10:** If $B_k$ is the first batch, stop.
- **Step 11:** Denote by $J_i$ a job in the batch preceding $B_k$. Apply Theorem 3 to determine whether any interchange of $J_i$ and $J_k$ is feasible. Let $S_{nth}$ be the set of jobs which can feasibly interchange with $J_k$. If $S_{nth} = \emptyset$, stop.
- **Step 12:** Apply Theorem 4 and Corollary 1 to determine whether any interchange of $J_i$ and each job in $S_{nth}$ are advantageous. Let $S_{nth}$ be the set of jobs which are advantageous to interchange with $J_k$. If $S_{nth} = \emptyset$, stop.
- **Step 13:** List the possible schedules according to $S_{nth}$ and employ Theorem 2 to reset the jobs in each batch in EDD order for each schedule. Determine the maximal tardiness for each schedule. Let $\sigma_s^*$ be the schedule in which one job is the most advantageous to interchange with $J_k$. (If a tie exists, break it arbitrarily). Replace $\sigma_s$ with $\sigma_s^*$ and return to Step 9.

Phase II of Phase II constructs an initial schedule, where jobs are sequenced in the EDD and the tool changes are included. Steps 9 and 10 check whether to stop the algorithm. Step 11 applies Theorem 3 to determine where the interchange of any two jobs is feasible. Step 12 adopts Theorem 4 and Corollary 1 to determine whether any interchange is advantageous. Step 13 obtains the most advantageous interchange.

To calculate the time complexity of Heuristic H1, carry out the bin-packing technique in Phase I in $O(n \log n)$. The complexity of Steps 9 and 10 are $O(n)$. Steps 11 and 12 require $O(n^2 \sum_{i=1}^n p_i)$. Step 2.6 needs $O(n \log n)$. Thus, the overall time complexity of the proposed heuristic is $O(n \sum_{i=1}^n p_i)$.

**Heuristic H2:** Phase I of Heuristic H2 first employs the EDD rule to sequence all jobs and then follows this sequence under the minimum number of batches to form batches. The steps of Heuristic H2 are outlined as follows:

**Phase I**

- **Step 14:** Sequence all jobs according to the EDD rule. Set $b = 1$, $B_0 = \emptyset$ and $U = \{J_{[1]}, J_{[2]}, \ldots, J_{[n]}\}$.
- **Step 15:** Set $u = \min\{i \mid J_i \in U\}$, $v = \max\{i \mid J_i \in U\}$, $k = u$, $TP = p_{[u]}$.
- **Step 16:** Delete $J_{[u]}$ from $U$ and append $J_{[u]}$ to $B_u$.
- **Step 17:** If $U = \emptyset$, then stop.
- **Step 18:** If $TP = T_{[k]}$, then set $b = b + 1$ and go to Step 15.
- **Step 19:** Set $k = k + 1$ and $TP = TP + p_{[k]}$.
- **Step 20:** If $TP > T_{[k]}$, then go to Step 16.
- **Step 21:** If $k > v$, then $TP = TP - p_{[k]}$ and go to Step 19. Otherwise, set $b = b + 1$ and go to Step 15.

**Phase II**

Phase II of Heuristic H2 is the same as Phase II of Heuristic H1.

Clearly, Phase I produces the smallest number of tool changes. To calculate the time complexity of Heuristic H2, carry out the modified bin-packing technique in Phase I in $O(n^2 \log n)$. Thus, the time complexity of the Heuristic H2 is $O(n \sum_{i=1}^n p_i)$.

**COMPUTATIONAL EXPERIMENT**

The test problems were divided into two sets, where one consisting of problems for which optimal solutions were known by solving the four optimization models in a reasonable time and the other containing problems for which optimal solutions were not known.
The numerical values of the problem parameters were generated according to the following scheme. For each test problem, the job processing times were randomly generated from a discrete uniform distribution between 5 and 15. The due dates were generated as a function of due date range, \( R \) and the tardiness factor, \( \tau \). Let \( T \) denote the sum of the processing times of all jobs. The due dates of the jobs were from a uniform distribution of integers between \( T \) (1–\( \tau \)-R/2) and \( T \) (1–\( \tau \)+R/2). In the experiment, \( \tau \) was set at 0.2 and 0.6; \( R \) assumed the values of 0.2 and 0.6; \( T_c \) was set at 10 and 18; \( T_c \) assumed the values of 2 and 4, respectively.

### Problems with known optimal solutions

The first set of problems was solved using the above four optimization models. The models were tested using a computer program coded in the ILOG OPL language. Problem parameters and four total models were generated and solved with ILOG CPLEX on an Intel P4/2.67GHz with 512MB SDRAM. The experiment was conducted using three different problem sizes, namely \( n = 6, 7 \) and 12. For each problem size, ten replications were produced for each combination of \( \tau \), \( R \), \( T_c \) and \( T_c \). Table 2 reports on the average computation time for each model in solving the associated problems.

#### Table 2: Computational results for problems with known optimal solutions

<table>
<thead>
<tr>
<th>( n )</th>
<th>( T_c )</th>
<th>( \tau )</th>
<th>( R )</th>
<th>( \text{Model 1} )</th>
<th>( \text{Model 2} )</th>
<th>( \text{Model 3} )</th>
<th>( \text{Model 4} )</th>
<th>( \text{Average percentage error of the heuristic (%)} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>1.0234</td>
<td>1.1796</td>
<td>12.3658</td>
<td>136.2811</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>1.5408</td>
<td>1.5157</td>
<td>13.4110</td>
<td>45.1516</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>0.6</td>
<td>0.8188</td>
<td>1.4845</td>
<td>23.4406</td>
<td>222.8265</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.2</td>
<td>1.3843</td>
<td>1.8184</td>
<td>9.3111</td>
<td>50.9828</td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td>0.2</td>
<td>0.2</td>
<td>0.8032</td>
<td>1.1983</td>
<td>19.4078</td>
<td>133.4220</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>1.2966</td>
<td>2.0532</td>
<td>7.9081</td>
<td>23.1860</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>0</td>
<td>0.2</td>
<td>0.2</td>
<td>1.0918</td>
<td>1.1518</td>
<td>6.3625</td>
<td>33.9673</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>2.7062</td>
<td>1.6595</td>
<td>4.9084</td>
<td>5.7219</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.2</td>
<td>0.2</td>
<td>0.5844</td>
<td>1.1593</td>
<td>5.0125</td>
<td>17.1984</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>1.4748</td>
<td>1.9925</td>
<td>4.5985</td>
<td>8.4811</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>0</td>
<td>0.2</td>
<td>0.2</td>
<td>1.1250</td>
<td>1.1062</td>
<td>6.6077</td>
<td>24.1360</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>1.3155</td>
<td>1.3471</td>
<td>2.7689</td>
<td>12.3016</td>
</tr>
<tr>
<td>7</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>1.9889</td>
<td>1.7859</td>
<td>135.6031</td>
<td>3183.3469</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>3.2720</td>
<td>3.1578</td>
<td>120.5032</td>
<td>772.2297</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>0</td>
<td>0.2</td>
<td>0.2</td>
<td>1.5638</td>
<td>2.9002</td>
<td>315.6376</td>
<td>2885.9421</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>3.4311</td>
<td>4.0142</td>
<td>54.1469</td>
<td>265.9688</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>2.1861</td>
<td>2.1905</td>
<td>370.7091</td>
<td>3424.2705</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>2.5283</td>
<td>3.3203</td>
<td>161.2016</td>
<td>1133.5561</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>0</td>
<td>0.2</td>
<td>0.2</td>
<td>2.2469</td>
<td>3.7484</td>
<td>366.7455</td>
<td>1030.2704</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>3.2859</td>
<td>5.2328</td>
<td>533.1705</td>
<td>2376.7352</td>
</tr>
<tr>
<td>12</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>191.0266</td>
<td>180.3391</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>111.9875</td>
<td>278.7579</td>
<td>9.7936</td>
<td>0.8902</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>113.8594</td>
<td>179.9609</td>
<td>1.0201</td>
<td>0.9202</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>258.3265</td>
<td>298.7267</td>
<td>1.8804</td>
<td>0.7910</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>0</td>
<td>0.2</td>
<td>0.2</td>
<td>149.7439</td>
<td>894.2564</td>
<td>1.5590</td>
<td>0.8984</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>423.7473</td>
<td>962.4126</td>
<td>1.1010</td>
<td>0.8591</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>176.3810</td>
<td>126.0453</td>
<td>5.7800</td>
<td>0.6902</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>245.6702</td>
<td>235.5955</td>
<td>1.2667</td>
<td>0.8502</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>109.7013</td>
<td>212.9097</td>
<td>0.8289</td>
<td>0.7320</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td>0.6</td>
<td>207.9047</td>
<td>190.6234</td>
<td>1.1532</td>
<td>0.9052</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>143.6263</td>
<td>244.8625</td>
<td>0.9937</td>
<td>0.7939</td>
</tr>
</tbody>
</table>

587
The effectiveness of the two heuristic algorithms is measured by the percentage error defined as:

\[
\text{Percentage error} = \frac{T_{\text{max}} \text{ (heu)} - T_{\text{max}} \text{ (opt)}}{T_{\text{max}} \text{ (opt)}} \times 100
\]

where, \( T_{\text{max}} \text{ (heu)} \) is the maximum tardiness obtained by the heuristic algorithm and \( T_{\text{max}} \text{ (opt)} \) is the optimal \( T_{\text{max}} \) of the schedule. Table 2 also summaries the performance of the two heuristic algorithms on the average percentage error. Table 2 yields the following observations.

For the four models, the computation time increases with increasing number of jobs. Additionally, the relationship between the efficiency of the proposed models and \( \tau \) is not significant. The average CPU time of Models 3 and 4 decreases with increasing \( T_L \) and decreasing \( T_C \). The efficiency of Models 3 and 4 increases with increasing R, while the efficiency of Models 1 and 2 decreases with increasing R. For the proposed problem, Model 1 is the best optimization model in average CPU time, followed by Models 2, 3 and 4. The real data testing results for the models differ from the results of the theoretical analysis.

For the two heuristics, Table 2 shows that problems with smaller \( \tau \) values generate smaller average percentage errors and spend less computation time. Also, problems with larger \( T_L \) value and smaller \( T_C \) value were found to produce smaller average percentage error. Moreover, the average percentage error increases as the number of jobs increases. Heuristic H1 spends less computation time than Heuristic H2. Heuristic H2 is better than Heuristic H1 in terms of average percentage error.

### Problems with unknown optimal solutions

Since optimal solutions were unknown for these problems, the relative performance is denoted by \( T_{\text{max}} \text{ (H2)} / T_{\text{max}} \text{ (H1)} \), where \( T_{\text{max}} \text{ (H2)} \) and \( T_{\text{max}} \text{ (H1)} \) are the maximum tardiness values found by Heuristics H2 and H1, respectively. If Heuristic H2 produces a better solution than that Heuristic H1, then its relative performance is less than 1. Table 3 shows the average relative performance values and average percentage error.

Heuristic H2 outperforms Heuristic H1 in terms of average relative performance. The average relative performance \( T_{\text{max}} \text{ (H2)} / T_{\text{max}} \text{ (H1)} \) is 0.8495 with a maximum of 0.9215. Heuristic H1 spends less computation time than Heuristic H2. Additionally, computation time increases with increasing \( \tau \).

<table>
<thead>
<tr>
<th>n</th>
<th>T_L</th>
<th>T_C</th>
<th>( \tau )</th>
<th>R</th>
<th>( T_{\text{max}} \text{ (H2)} )</th>
<th>( T_{\text{max}} \text{ (H1)} )</th>
<th>R</th>
<th>( T_{\text{max}} \text{ (H2)} )</th>
<th>( T_{\text{max}} \text{ (H1)} )</th>
<th>R</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.8201</td>
<td>11.3467</td>
<td>0.6</td>
<td>0.8750</td>
<td>11.461</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9215</td>
<td>15.1252</td>
<td></td>
<td>0.8942</td>
<td>12.8541</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8592</td>
<td>11.0912</td>
<td></td>
<td>0.8427</td>
<td>10.5061</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8504</td>
<td>12.4752</td>
<td></td>
<td>0.8107</td>
<td>13.7892</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.8209</td>
<td>12.3245</td>
<td>0.6</td>
<td>0.9210</td>
<td>11.3782</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9230</td>
<td>12.3782</td>
<td></td>
<td>0.8590</td>
<td>12.9662</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8590</td>
<td>12.9662</td>
<td></td>
<td>0.8205</td>
<td>14.1322</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8815</td>
<td>10.4562</td>
<td></td>
<td>0.9010</td>
<td>11.5412</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.9200</td>
<td>11.3127</td>
<td></td>
<td>0.9001</td>
<td>12.1562</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.8350</td>
<td>41.2502</td>
<td>0.6</td>
<td>0.8590</td>
<td>38.8382</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8352</td>
<td>39.3547</td>
<td></td>
<td>0.8750</td>
<td>41.2502</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8204</td>
<td>43.8562</td>
<td></td>
<td>0.8207</td>
<td>43.7527</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8817</td>
<td>47.2514</td>
<td></td>
<td>0.9050</td>
<td>45.1245</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.2</td>
<td>0.8140</td>
<td>48.8386</td>
<td>0.6</td>
<td>0.8708</td>
<td>49.1508</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8212</td>
<td>52.2587</td>
<td></td>
<td>0.8051</td>
<td>50.9921</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8475</td>
<td>45.7252</td>
<td></td>
<td>0.8475</td>
<td>46.3892</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8258</td>
<td>47.1602</td>
<td></td>
<td>0.8258</td>
<td>47.1602</td>
<td>0.6</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8318</td>
<td>50.3760</td>
<td></td>
<td>0.8318</td>
<td>50.3760</td>
<td>0.6</td>
</tr>
</tbody>
</table>

### CONCLUSIONS

This investigation studied the single CNC machine scheduling problem of minimizing the maximum tardiness with taking account of tool changes. Four mixed BIP models to obtain the optimal solution. These four mixed BIP models are Models 1, 2, 3 and 4. Model 1 is the best optimization model in average CPU time, followed by Models 2, 3 and 4. Two efficient heuristics, Heuristics H1 and H2, were tested. The performance of these heuristics was compared with the optimal solutions. The results showed that Heuristic H2 outperformed Heuristic H1 in terms of average relative performance. The average relative performance \( T_{\text{max}} \text{ (H2)} / T_{\text{max}} \text{ (H1)} \) is 0.8495 with a maximum of 0.9215. Heuristic H1 spends less computation time than Heuristic H2. Additionally, computation time increases with increasing \( \tau \).
and H2, were proposed to provide the near-optimal solution for the problem. Performance of the heuristics was evaluated by comparing its solution with the optimal solution derived by the developed mixed BIP models. Several properties associated with the problem have also been investigated and implemented in the algorithm. Heuristic H1 spends less computation time than Heuristic H2. Heuristic H2 is better than Heuristic H1 in terms of average percentage error. The computational results show that the mixed BIP models are inefficient, especially Models 3 and 4. The two heuristic algorithms can improve the maximum tardiness of jobs in terms of percentage error. The heuristics were shown to spend much less computation time than the mixed BIP models. Therefore, the heuristic is applicable for large-sized problems, while the mixed BIP models can only be used for small-sized problems.

Future research should address problems with different shop environments, including flow-shop and job-shop. Problems with other performance measures, including mean tardiness and multi-criteria measures, should also be studied.

ACKNOWLEDGMENT

The author would like to thank the National Science Council of the Republic of China, Taiwan for financially supporting this research under Contract No. NSC93-2213-E-269-003.

REFERENCES


