http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet

Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 7 (4): 580-589, 2008
ISSN 1812-5638

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

[1]Ming-Cheng Lo, [2]Jen-Shiang Chen and [3]Yi-Chien Chen
[1]Department of Business Administration, Ching Yun University, Taiwan, Republic of China
[2]Department of Business Administration, Far East University, Taiwan, Republic of China
[3]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 (Akturk *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 (while bounded by the tool life) 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. Akturk *et al*. (2003) studied the same problem for scheduling with tool changes to minimize total completion time. They provided a mixed Binary Integer Programming (BIP) model for the exact solution of the problem and proposed seven heuristics based on simple dispatch rules and generic search. Akturk *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

Corresponding Author: Ming-Cheng Lo, Department of Business Administration, Ching Yun University, 229, Chien-Hsin Road, Jung-Li, 320, Taiwan, Republic of China

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

J_i : Job number i
$J_{[k]}$: The job placed at the kth 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 J_i
d_i : The due date of J_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_j : The starting time of the job in the sequence position j (only for Model 1)
f_j : The finish time of the job in the sequence position j (only for Model 1)
s_i : The earliest start time of J_i (only for Model 2)
C_i : Completion time of J_i (only for Models 1 and 2)
b_j : 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_j : 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_j : 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 : Tardiness of J_i; where $T_i = \max\{0, C_i-d_i\}$
T_{max} : $\text{Max}_j\{T_j\}$
x_{ij} : 1 if J_i is scheduled at position j; 0 otherwise (only for Models 1, 3 and 4)
z_{ij} : 1 if J_i precedes J_j (not necessarily immediately); 0 otherwise (only for Model 2)

Assume that n independent jobs $J_1, J_2, \ldots, J_n$ 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_L and the tool change time is T_C. The relation $T_L \geq p_i$ (i = 1, 2, ..., n) is assumed since otherwise no feasible schedule is available.

Minimizing the maximum tardiness 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

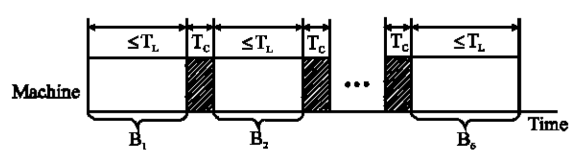

Fig. 1: Representation of a schedule as batch of jobs

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 (Rinnooy 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_L 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 $J_{n+1}, J_{n+2}, \ldots, J_{2n-1}$. Models 1 and 2 are based on the concept of (n-1) tool changes can occur.

The binary variable x_{ij} 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.

$$\text{Minimize} \quad T_{max} \tag{1}$$

$$\text{S.t} \quad \sum_{j=1}^{2n-1} x_{ij} = 1 \quad i = 1, 2, \ldots, n \tag{2}$$

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

$$\sum_{i=1}^{n} x_{i1} = 1 \tag{4}$$

$$\sum_{i=1}^{n+int(j/2)} x_{ij} = 1 \quad j = 2, 3, \ldots, 2n-1 \text{; where, int (y) denote the greatest integer less than or equal y} \tag{5}$$

$$x_{ij} = 0 \quad i = n+1, n+2, \ldots, 2n-1;\ j = 1, 2, \ldots, 2(i-n)-1 \tag{6}$$

$$x_{ij} = 0 \quad i = n+1, n+2, \ldots, 2n-2;\ j = i+1, i+2, \ldots, 2n-1 \tag{7}$$

$$h_1 + \sum_{i=1}^{n} p_i x_{i1} = f_1 \tag{8}$$

$$h_j + \sum_{i=1}^{n} p_i x_{ij} + T_c \sum_{i=n+1}^{2n-1} x_{ij} = f_j \quad j = 2, 3, \ldots, 2n-1 \tag{9}$$

$$f_j \le h_{j+1} \quad j = 1, 2, \ldots, 2n-2 \tag{10}$$

$$f_{j-1} \le T_L + M(1 - x_{n+1,j}) \quad j = 2, 3, \ldots, n+1 \tag{11}$$

$$f_{j-1} - f_{j'} \le T_L + M(2 - x_{ij} - x_{i-1,j'}) \quad i = n+2, n+3, \ldots, 2n-1;\ j = 2(i-n), 2(i-n)+1, \ldots, i;\ j' = 2(i-n-1), 2(i-n-1)+1, \ldots, i-1 \text{ and } j > j' \tag{12}$$

$$f_j \le C_i + M(1 - x_{ij}) \quad i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, 2n-1 \tag{13}$$

$$C_i - d_i \le T_{max} \quad i = 1, 2, \ldots, n \tag{14}$$

$$T_{max} \ge 0;\ h_j \ge 0,\ f_j \ge 0 \quad j = 1, 2, \ldots, 2n-1;\ C_i \ge 0 \quad i = 1, 2, \ldots, n;\ x_{ij} \text{ is binary} \quad i, j = 1, 2, \ldots, 2n-1 \tag{15}$$

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_{max}, h_j, f_j and C_i and x_{ij}, respectively, are specified in (15).

Model 2: This model uses the binary variable z_{ij} 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.

$$\text{Minimize} \quad T_{max} \tag{16}$$

$$\text{S.t.} \quad s_i + p_i = C_i \quad i = 1, 2, \ldots, n \tag{17}$$

$$s_i + T_C = C_i \quad i = n+1, n+2, \ldots, 2n-1 \tag{18}$$

$$C_i\text{-}d_i \le T_{max}\ i = 1, 2, \ldots, n \tag{19}$$

$$C_i \le s_j + M\,(1\text{-}z_{ij})\quad 1 \le i<j \le 2n\text{-}1 \tag{20}$$

$$C_j \le s_i + M\, z_{ij}\ 1 \le i<j \le 2n\text{-}1 \tag{21}$$

$$s_{n+1} \le T_L \tag{22}$$

$$s_i \le C_{i\text{-}1} + T_L\quad i = n + 2, n + 3, \ldots, 2n\text{-}1 \tag{23}$$

$$T_{max} \ge 0;\ s_i \ge 0,\ C_i \ge 0\quad i = 1, 2, \ldots, 2n\text{-}1;\quad z_{ij} \text{ is binary } 1 \le i<j \le 2n\text{-}1 \tag{24}$$

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 $C_i \le s_j$ or $C_j \le s_i$ will hold. Incorporating binary variable z_{ij} 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_{ij}.

Model 3: A new binary variable k_j is defined to be equal to 1 if the tool is replaced after position j and 0 otherwise. The variable r_j 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.

$$\text{Minimize } T_{max} \tag{25}$$

$$\text{S.t. } \sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, 2, \ldots, n \tag{26}$$

$$\sum_{i=1}^{n} x_{ij} = 1 \quad j = 1, 2, \ldots, n \tag{27}$$

$$r_1 + \sum_{i=1}^{n} p_i x_{i1} = T_L \tag{28}$$

$$r_j + \sum_{i=1}^{n} p_i x_{ij} \le r_{j\text{-}1} + M\, k_{j\text{-}1}\quad j = 2, 3, \ldots, n \tag{29}$$

$$r_j + \sum_{i=1}^{n} p_i x_{ij} \le T_L + M\,(1\text{-}k_{j\text{-}1})\quad j = 2, 3, \ldots, n \tag{30}$$

$$b_j \ge T_L\text{-}r_j\text{-}M\,(1\text{-}k_j)\quad j = 1, 2, \ldots, n\text{-}1 \tag{31}$$

$$f_j = \sum_{j'=1}^{j-1} b_{j'} + T_C \sum_{j'=1}^{j-1} k_{j'}\quad j = 1, 2, \ldots, n \tag{32}$$

$$f_j\text{-} \sum_{i=1}^{n} d_i x_{ij} \le T_{max}\quad j = 1, 2, \ldots, n \tag{33}$$

$$\begin{aligned} &T_{max} \ge 0;\ r_j \ge 0,\ f_j \ge 0 && j = 1, 2, \ldots, n; \\ &b_j \ge 0 && j = 1, 2, \ldots, n\text{-}1 \\ &x_{ij} \text{ is binary} && i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n \\ &k_j \text{ is binary} && j = 1, 2, \ldots, n\text{-}1 \end{aligned} \tag{34}$$

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_j, that is, either

$$r_j + \sum_{i=1}^{n} p_i x_{ij} \le r_{j\text{-}1} \text{ or } r_j + \sum_{i=1}^{n} p_i x_{ij} \le T_L$$

will hold. Incorporating binary variable $k_{j\text{-}1}$ 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_{ij} 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.

$$\text{Minimize } T_{max} \tag{35}$$

$$\text{S.t. } \sum_{j=1}^{n} x_{ij} = 1 \quad i = 1, 2, \ldots, n \tag{36}$$

$$\sum_{i=1}^{n} x_{ij} = 1\ j = 1, 2, \ldots, n \tag{37}$$

$$e_1 = \sum_{i=1}^{n} p_i x_{i1} \tag{38}$$

$$e_{j-1} + \sum_{i=1}^{n} p_i x_{ij} \le e_j + M\, k_{j-1} \tag{39}$$
$j = 2, 3, ..., n$

$$\sum_{i=1}^{n} p_i x_{ij} \le e_j + M\,(1-k_{j-1}) \tag{40}$$
$j = 2, 3, ..., n$

$$e_j \le T_L \; j = 2, 3, \ldots, n \tag{41}$$

$$b_j \ge e_j - M\,(1-k_j)$$
$$j = 1, 2, \ldots, n-1 \tag{42}$$

$$f_j = \sum_{j'=1}^{j-1} b_{j'} + ej + T_C \sum_{j'=1}^{j-1} k_{j'} \tag{43}$$
$j = 1, 2, ..., n$

$$f_j - \sum_{i=1}^{n} d_i x_{ij} \le T_{max} \tag{44}$$
$j = 1, 2, ..., n$

$T_{max} \ge 0$; $e_j \ge 0$, $f_j \ge 0$, $j = 1, 2, \ldots, n$;
$b_j \ge 0$ $j = 1, 2, \ldots, n-1$;

x_{ij} is binary

$i = 1, 2, \ldots, n$; $j = 1, 2, \ldots, n$;
k_j is binary
$j = 1, 2, \ldots, n-1$; (45)

Constraint (35) describes the objective function, constraint sets (36) and (37) state in which each job is uniquely scheduled in a position for processing. Constraint (38) records the elapsed time of the tool which processes the job in the first position. Additionally, constraint sets (39) and (40) meet the requirement that the range of e_j, that is, either

$$e_{j-1} + \sum_{i=1}^{n} p_i x_{ij} \le e_j \text{ or } \sum_{i=1}^{n} p_i x_{ij} \le e_j$$

will hold. Incorporation binary variable k_{j-1} and a very large positive number M, Eq. 39 and 40 together ensure that one of these two constraints holds while the other is eliminated. Furthermore, constraint set (41) states the restriction of the tool life. Constraint sets (42) to (44) define the b_j, f_j and T_{max}, respectively. Finally, constraint set (45) specifies the non-negativity of T_{max}, e_j, f_j and b_j and establishes the binary restrictions for x_{ij} and k_j.

Table 1: The size of integer programming model

Model	No. of binary variables	No. of constraints	No. of continuous variables
1	$4n^2$-4n + 1	$(7/2)\, n^2 + (11/2)\, n + \sum_{i=3}^{n}\sum_{j=2}^{i} j - 2$	5n-1
2	$2n^2$-3n + 1	$4n^2$-2n + 1	4n-1
3	n^2 + n-1	7n-1	3n
4	n^2 + n-1	8n-2	3n

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 $(-1/2)\, n^2 + (15/2)\, n + \sum_{i=3}^{n}\sum_{j=2}^{i} j - 3$ 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_q denotes batch number q, where q = 1, 2, …, 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 J_k is in B_q. B_q is not the first batch in the schedule. Denote by J_j a job in the batch preceding B_q. Let BT_k and BT_j be the total processing time of the batch which contains J_k and J_j, respectively. If $BT_j-p_j + p_k \le T_L$ and $BT_k-p_k+p_j \le T_L$, then a feasible schedule can be obtained by interchanging J_k with J_j.

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 J_k is in B_q. B_q is not the first batch in the schedule. Denote by J_j a job in the batch preceding B_q. Suppose that J_i is not placed in B_q and between J_j and J_k. In a feasible schedule, if the following two conditions can be satisfied simultaneously, then a smaller T_{max} value can be obtained by interchanging J_k with J_j.

(i) $C_k \le d_j$, or $C_k>d_j$ and $d_k \le d_j$, or $C_j>d_j$ and $C_k \le C_j$,
(i) $C_i + p_k-p_j \le d_i$, or $C_i + p_k-p_j-d_i \le C_k-d_k$, or $C_i>d_i$ and $p_k \le p_j$

Proof: See Fig. 2, from Theorem 2 both jobs J_k and J_j are assumed to be feasibly interchangeable. Let schedule S = (PS_S, J_j, PS_1, J_i, PS_2, J_k, PS_F), where PS_S, PS_1, PS_2 and PS_F are partial schedules. Let S′ be obtained from S by interchanging J_j with J_k, that is, S′ = (PS_S, J_k, PS_3, J_i, PS_4, J_j, PS_F), where PS_3 and PS_4 are also partial schedules. Denote by T_r (S) and T_r (S′) the tardiness of J_r in schedule S and S′, respectively (Fig. 2). Then the tardiness of T_k (S), T_j (S), T_i (S), T_k (S′), T_j (S′) and T_i (S′) are

$$
\begin{aligned}
T_k (S) &= \max\{C_k-d_k, 0\}\\
T_j (S) &= \max\{C_j-d_j, 0\}\\
T_i (S) &= \max\{C_i-d_i, 0\}\\
T_k (S') &= \max\{C_j + p_k-p_j-d_k, 0\}\\
T_j (S') &= \max\{C_k -d_j, 0\}\\
T_i (S') &= \max\{C_i + p_k-p_j-d_i, 0\}
\end{aligned}
$$

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

Case 1: T_k (S′)≤T_k (S).

Obviously, T_k (S′)≤T_k (S) must hold.

Case 2: T_j (S′)≤T_k (S) or T_j (S′)≤T_j (S).

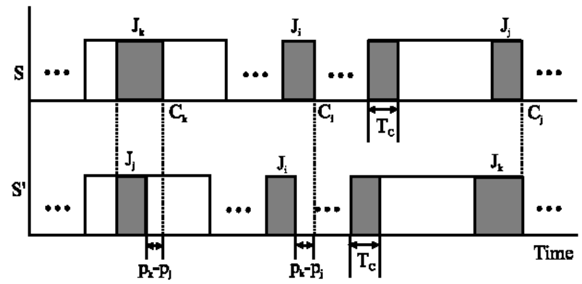

Fig. 2: Illustration of interchange with preceding job

To satisfy T_j (S′) = $\max\{C_k-d_j, 0\} \le T_k$ (S) = $\max\{C_k-d_k, 0\}$, the following two conditions must be satisfied.

$$\text{(i) } C_k \le d_j, \text{ or (ii) } C_k>d_j \text{ and } d_k \le d_j \qquad (46)$$

Then, to satisfy T_j (S′) = $\max\{C_k-d_j, 0\} \le T_j$ (S) = $\max\{C_j-d_j, 0\}$, the following two conditions must also be satisfied.

$$\text{(i) } C_k \le d_j, \text{ or (ii) } C_k>d_j,\ C_j>d_j, \text{ and } C_k \le C_j. \qquad (47)$$

Case 3: T_i (S′)≤T_k (S) or T_i (S′)≤T_i (S).

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

$$\text{(i) } C_i + p_k-p_j \le d_i, \text{ or (ii) } C_i + p_k-p_j>d_i \text{ and } C_i + p_k-p_j-d_i \le C_k-d_k \qquad (48)$$

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

$$\text{(i) } C_i + p_k-p_j \le d_i, \text{ or (ii) } C_i + p_k-p_j>d_i,\ C_i>d_i \text{ and } p_k \le p_j. \qquad (49)$$

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_k is in B_q and J_j placed the last position of the batch B_{q-1}. If the first condition (i) (that is, $C_k \le d_j$, or $C_k>d_j$ and $d_k \le d_j$, or $C_j>d_j$ and $C_k \le C_j$) can be satisfied, then a smaller T_{max} value can be obtained by interchanging J_k with J_j.

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 $b = 1$, $B_b = \varnothing$, $k = 1$, $TP = p_{[k]}$ and $U = \{J_{[1]}, J_{[2]}, \ldots, J_{[n]}\}$.

Step 2 = Delete $J_{[k]}$ from U and append $J_{[k]}$ to B_b.

Step 3 = If $k = n$, then stop.

Step 4 = If $TP = T_L$, then set $b = b + 1$ and $TP = 0$.

Step 5 = Set $k = k + 1$ and $TP = TP + p_{[k]}$.

Step 6 = If $TP>T_L$, then set $b = b + 1$ and $TP = p_{[k]}$.

Step 7 = Go to Step 2.

Phase II

Step 8 = Denote by σ_π 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_k in B_q with T_{max} (if a tie exists, break it arbitrarily).

Step 10 = If B_q is the first batch, stop.

Step 11 = Denote by J_j a job in the batch preceding B_q. Apply Theorem 3 to determine whether any interchange of J_k and J_j is feasible. Let S_{FI} be the set of jobs which can feasibly interchange with J_k. If $S_{FI} = \varnothing$, stop.

Step 12 = Apply Theorem 4 and Corollary 1 to determine whether any interchange of J_k and each job in S_{FI} are advantageous. Let S_{adv} be the set of jobs which are advantageous to interchange with J_k. If $S_{adv} = \varnothing$, stop.

Step 13 = List the possible schedules according to S_{adv} 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 σ'_π be the schedule in which one job is the most advantageous to interchange with J_k (If a tie exists, break it arbitrarily). Replace σ_π with σ'_π and return to Step 9.

The steps are now elaborated in detail. Clearly that the smallest number of tool changes is obtained in Phase I. Step 8 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 (nlogn). The complexity of Steps 9 and 10 are O (n). Steps 11 and 12 require $O(n^2 \sum_{k=1}^{n} p_{[k]})$. Step 2.6 needs O (nlogn). Thus, the overall time complexity of the proposed heuristic is $O(n^2 \sum_{i=1}^{n} p_i)$.

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 $b = 1$, $B_b = \varnothing$ and $U = \{J_{[1]}, J_{[2]}, \ldots, J_{[n]}\}$.

Step 15 = Set $u = \min\{i \mid J_{[i]} \in U\}$, $v = \max\{i \mid J_{[i]} \in U\}$, $k = u$, $TP = p_{[k]}$.

Step 16 = Delete $J_{[k]}$ from U and append $J_{[k]}$ to B_b.

Step 17 = If $U = \varnothing$, then stop.

Step 18 = If $TP = T_L$, then set $b = b +1$ and go to Step 15.

Step 19 = Set $k = k + 1$ and $TP = TP + p_{[k]}$.

Step 20 = If $TP \le T_L$, then go to Step 16.

Step 21 = If $k \neq v$, 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^2logn). Thus, the time complexity of the Heuristic H2 is $O(n^2 \sum_{i=1}^{n} p_i)$.

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, τ. 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, τ was set at 0.2 and 0.6; R assumed the values of 0.2 and 0.6; T_L 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 τ, R, T_L 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

					Average CPU time of the model (sec)				Average percentage error of the heuristic (%)	
n	T_L	T_C	τ	R	Model 1	Model 2	Model 3	Model 4	H1	H2
6	10	2	0.2	0.2	1.0234	1.1796	12.3658	136.2811	0.0011	0.0010
				0.6	1.5498	1.5157	13.4110	43.1516	0.0017	0.0015
			0.6	0.2	0.8188	1.4485	23.4406	222.8265	0.0020	0.0015
				0.6	1.3843	1.8484	9.3111	50.0828	0.0014	0.0012
		4	0.2	0.2	0.8032	1.1983	19.4078	133.4220	0.0018	0.0010
				0.6	1.2966	2.0532	7.9081	23.1860	0.0015	0.0005
			0.6	0.2	0.7312	1.4205	33.4188	257.6388	0.0020	0.0005
				0.6	1.3859	1.8983	12.5392	28.5765	0.0024	0.0010
	18	2	0.2	0.2	1.0918	1.1518	6.3625	33.0673	0.0009	0.0000
				0.6	2.7062	1.6595	4.0984	5.7219	0.0011	0.0011
			0.6	0.2	0.5844	1.1593	5.0125	17.1984	0.0012	0.0010
				0.6	1.4748	1.9925	4.5985	8.4811	0.0010	0.0010
		4	0.2	0.2	1.1250	1.1062	6.6077	24.1360	0.0005	0.0000
				0.6	1.3155	1.3471	2.7689	12.3016	0.0010	0.0005
			0.6	0.2	0.4719	1.4343	6.6826	18.8783	0.0015	0.0005
				0.6	0.8624	1.4547	1.8375	5.6921	0.0013	0.0010
7	10	2	0.2	0.2	1.9889	1.7859	135.6031	3183.3469	0.0082	0.0025
				0.6	3.2720	3.1578	120.5032	772.2297	0.0055	0.0055
			0.6	0.2	1.3638	2.9002	315.6376	2885.0421	0.0130	0.0085
				0.6	3.4311	4.0142	54.1469	265.0688	0.0124	0.0024
		4	0.2	0.2	2.1861	2.1905	370.7093	3424.2705	0.0114	0.0095
				0.6	2.5283	3.3203	161.2016	1133.5561	0.0087	0.0082
			0.6	0.2	2.2469	3.7484	306.7455	1030.2704	0.0191	0.0080
				0.6	3.2859	5.2328	353.1705	2376.7952	0.0126	0.0025
	18	2	0.2	0.2	2.5626	2.1577	26.5389	200.0344	0.0080	0.0052
				0.6	2.2404	2.0326	18.6252	17.8829	0.0104	0.0084
			0.6	0.2	1.8654	2.7686	30.1080	150.6376	0.0095	0.0088
				0.6	3.8843	3.6968	17.1406	32.2907	0.0110	0.0022
		4	0.2	0.2	1.9298	1.8514	29.3999	245.0204	0.0121	0.0095
				0.6	3.1268	2.4123	26.3406	41.1563	0.0059	0.0020
			0.6	0.2	1.1764	2.8033	28.2592	181.4079	0.0106	0.0124
				0.6	2.0765	3.8077	16.6234	67.5002	0.0216	0.0025
12	10	2	0.2	0.2	191.0266	180.3391	N/A		0.8900	0.8002
				0.6	111.9875	278.7579			0.9736	0.8094
			0.6	0.2	113.8594	179.9609			1.0201	0.9202
				0.6	258.3265	298.7267			1.4994	0.9011
		4	0.2	0.2	149.7439	894.2564			0.8804	0.7910
				0.6	1423.7453	962.4126			1.0010	0.8539
			0.6	0.2	256.4610	543.7158			1.7508	0.9020
				0.6	222.2000	609.3404			0.8839	0.7811
	18	2	0.2	0.2	176.3810	126.0453			0.7800	0.6902
				0.6	245.6702	235.5955			0.9310	0.8539
			0.6	0.2	109.7013	212.9907			1.2607	0.8502
				0.6	207.9047	190.6234			0.8239	0.7320
		4	0.2	0.2	143.6263	244.8625			0.9310	0.6011
				0.6	133.1219	131.9765			0.9604	0.8239
			0.6	0.2	114.9377	162.2374			1.1532	0.9052
				0.6	162.6576	204.0516			0.9937	0.7939

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

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

where, T_{max} (heu) is the maximum tardiness obtained by the heuristic algorithm and T_{max} (opt) is the optimal T_{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 τ 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 τ 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_{max} (H2) /T_{max} (H1), where T_{max} (H2) and T_{max} (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_{max} (H2)/T_{max} (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 τ.

Table 3: Computational results of the proposed heuristics with Problems with unknown optimal solutions

n	T_L	T_C	τ	R	Average relative performance T_{max} (H2)/ T_{max} (H1)	Average CPU time of the heuristic (sec) H1	H2
50	10	2	0.2	0.2	0.8201	11.3467	9.1102
				0.6	0.8750	11.1461	10.8542
			0.6	0.2	0.9215	15.1252	14.0571
				0.6	0.8942	12.8541	10.1242
		4	0.2	0.2	0.8662	11.0012	9.8761
				0.6	0.8427	10.5601	9.4587
			0.6	0.2	0.8504	12.4752	11.4558
				0.6	0.8307	13.7892	10.5411
	18	2	0.2	0.2	0.8209	12.3245	8.1857
				0.6	0.9210	11.3782	10.7824
			0.6	0.2	0.8500	12.9662	11.3891
				0.6	0.8205	14.1522	10.7854
		4	0.2	0.2	0.8815	10.4562	8.4511
				0.6	0.9010	11.5412	9.9682
			0.6	0.2	0.9200	11.3127	10.1254
				0.6	0.9001	12.1562	10.7854
100	10	2	0.2	0.2	0.8500	41.2562	35.1220
				0.6	0.8350	38.3782	34.1452
			0.6	0.2	0.8352	39.3547	33.44521
				0.6	0.8204	43.8562	40.7854
		4	0.2	0.2	0.8570	45.7527	43.2254
				0.6	0.8207	51.1564	40.1524
			0.6	0.2	0.8817	47.2514	40.7851
				0.6	0.9050	53.8542	45.1245
	18	2	0.2	0.2	0.8140	48.3866	40.3655
				0.6	0.8708	49.1568	43.4482
			0.6	0.2	0.8242	52.2582	41.8572
				0.6	0.8051	50.9921	40.1262
		4	0.2	0.2	0.8475	45.7521	42.3472
				0.6	0.8247	46.3892	40.2461
			0.6	0.2	0.8258	47.1062	41.3872
				0.6	0.8318	50.3760	45.2467
200	10	2	0.2	0.2	0.8152	180.7542	175.0452
				0.6	0.8280	190.8596	182.0555
			0.6	0.2	0.8785	200.4527	162.7472
				0.6	0.8307	186.8652	149.0521
		4	0.2	0.2	0.8108	171.5262	158.7828
				0.6	0.8709	160.2522	150.0212
			0.6	0.2	0.8210	180.4521	163.8542
				0.6	0.8208	168.8965	161.0911
	18	2	0.2	0.2	0.8109	165.3524	148.1482
				0.6	0.8240	178.0578	140.7528
			0.6	0.2	0.8801	176.0856	160.2142
				0.6	0.8752	175.2035	168.1852
		4	0.2	0.2	0.8170	160.3021	140.9542
				0.6	0.8285	169.2062	161.9042
			0.6	0.2	0.8657	178.2928	154.8721
				0.6	0.8371	183.1922	165.0876

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 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

Akturk, M.S., J.B. Ghosh and E.D. Gunes, 2003. Scheduling with tool changes to minimize total completion time: A study of heuristics and their performance. Nav. Res. Logistics, 50 (1): 15-30.

Akturk, M.S., J.B. Ghosh and E.D. Gunes, 2004. Scheduling with tool changes to minimize total completion time: Basic results and SPT performance. Eur. J. Oper. Res., 157 (3): 784-790.

Blazewicz, J., M. Drozdowski, P. Formanowicz, W. Kubiak and G. Schmidt, 2000. Scheduling preemptable tasks on parallel processors with limited availability. Parallel Comput., 26: 1195-1211.

Cassady, C.R. and E. Kutanoglu, 2003. Minimizing job tardiness using integrated preventive maintenance planning and production scheduling. IIE. Trans., 35 (6): 503-513.

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

Graves, G.H. and C.Y. Lee, 1999. Scheduling maintenance and semiresumable jobs on a single machine. Nav. Res. Logistics, 46 (7): 845-863.

Lee, C.Y., 1996. Machine scheduling with an availability constraint. J. Global Optim., 9 (3-4): 395-416.

Liao, C.L. and C.T. You, 1992. Improved formulation for the job-shop scheduling problem. J. Oper. Res. Soc., 43 (11): 1047-1054.

Liao, C.L. and W.J. Chen, 2003. Single-machine scheduling with periodic maintenance and nonresumable jobs. Comput. Oper. Res., 30 (9): 1335-1347.

Qi, X., T. Chen and F. Tu, 1999. Scheduling the maintenance on a single machine. J. Oper. Res. Soc., 50 (10): 1071-1078.

Rinnooy Kan, A.G., 1976. Machine Scheduling Problems: Classification, Complexity and Computations. Martinus Nijhoff, The Hague, Holland.

Sanlaville, E. and G. Schmidt, 1998. Machine sche duling with availability constraints. Acta Inform., 35 (9): 795-811.

Schmidt, G., 1988. Scheduling independent tasks with deadlines on semi-identical processors. J. Oper. Res. Soc., 39 (3): 271-277.

Schmidt, G., 2000. Scheduling with limited machine availability. Eur. J. Oper. Res., 121 (1): 1-15.

Wilson, J.M., 1989. Alternative formulations of a flow-shop scheduling problem. J. Oper. Res. Soc., 40 (4): 395-399.