Using integer programming to solve the machine scheduling problem with a flexible maintenance activity

Jen-Shiang Chen

Department of Industrial Engineering and Management
Far East College
49 Junghua Road, Shinshr Shiang
Tainan 744
Taiwan
R.O.C.

Abstract
This work addresses the single machine and parallel machine scheduling problems, where machine is flexibly maintained and mean flow time is used as a performance measure. Machine $M_k$ should be stopped for maintenance for a constant time $w_k$ in the schedule. The maintenance period $[u_k, v_k]$ is assumed to be set in advance, and the maintenance time $w_k$ is assumed not to exceed the maintenance period (that is, $w_k \leq v_k - u_k$). The time $u_k$ ($v_k$) is the earliest (latest) time at which the machine $M_k$ starts (stops) its maintenance. Two cases, resumable and unresumable, are considered in the single machine and parallel machine problems, respectively. Moreover, four integer programming models are developed optimally to solve the problem.

Keywords: Scheduling, maintenance, integer programming, single machine, parallel machine.

Symbol definition

\[ J_i \]  job number $i$;
\[ M_k \]  machine number $k$.

Problem parameters

\[ M \]  a very large positive number;
\[ n \]  number of jobs for processing at time zero;
\[ m \]  number of machines in the shop;

*E-mail: jschenc@ms25.hinet.net

Journal of Statistics & Management Systems
Vol. 9 (2006), No. 1, pp. 87–104
© Taru Publications
Decision variables

\( s_i \) \hspace{1cm} the earliest start time of \( J_i \) (only used for the single machine problems);

\( f_i \) \hspace{1cm} the earliest finish time of \( J_i \) (only used for the single machine problems);

\( h_q \) \hspace{1cm} the earliest start time of the job in the sequence position \( q \) (only used for the single machine problems);

\( f_q' \) \hspace{1cm} the earliest finish time of the job in the sequence position \( q \) (only used for the single machine problems);

\( s_{ik} \) \hspace{1cm} the earliest start time of \( J_i \) on \( M_k \) (only used for the parallel machine problems);

\( f_{ik} \) \hspace{1cm} the earliest finish time of \( J_i \) on \( M_k \) (only used for the parallel machine problems);

\( h_{kq} \) \hspace{1cm} the earliest start time of the job is scheduled on \( M_k \) in the sequence position \( q \) (only used for the parallel machine problems);

\( f_{kq}' \) \hspace{1cm} the earliest finish time of the job is scheduled on \( M_k \) in the sequence position \( q \) (only used for the parallel machine problems);

\( z_{ij} \) \hspace{1cm} 1 if \( J_i \) precedes \( J_j \) (not necessarily immediately); 0 otherwise;

\( x_{iq}' \) \hspace{1cm} 1 if \( J_i \) is scheduled at position \( q \); 0 otherwise (only used for the single machine problems);

\( x_{ik} \) \hspace{1cm} 1 if \( J_i \) is processed on \( M_k \); 0 otherwise (only used for the parallel machine problems);

\( x_{ikq}' \) \hspace{1cm} 1 if \( J_i \) is scheduled on \( M_k \) at position \( q \); 0 otherwise (only used for the parallel machine problems).

1. Introduction

Classical scheduling problems assume that machines are always available for processing jobs. However, this assumption is inappropriate in several real world situations. Machines may be unavailable during
the scheduling horizon, due to breakdown (stochastic) or preventive maintenance (deterministic) [6]. The textbook by Pinedo [18, p. 393] emphasized the need for research in this area:

“Most theoretical models do not take machine availability constraints into account; usually it is assumed that machines are available at all times. In practice, machines are usually not continuously available”.

Research on scheduling, with machine availability constraints, focuses mainly on machine breakdowns, maintenance activities, and tool changes. Related literature falls into four categories.

(i) Each machine has a single unavailability interval and the starting time of this interval is constant (called Problem SC). See for example, Adiri et al. [1], Leon and Wu [14], Lee and Liman [11, 12], Mosheiov [16], Lee [5, 6], Lee et al. [8], and Tan and Le [24]. Lee et al. [9], Sanlaville and Schmidt [21], and Schmidt [23] provided surveys of this topic.

(ii) Each machine has a single unavailability interval and the starting time of this interval is a decision variable (called Problem SV). See Lee and Leon [10], Lee and Lin [13], and Yang et al. [25].

(iii) Each machine has multiple unavailability intervals and the starting time of each interval is constant (called Problem MC). See Schmidt [22].

(iv) Each machine has multiple unavailability intervals and the starting time of each interval is a decision variable (called Problem MV). See Qi et al. [19], Graves and Lee [4], Błazewicz et al. [3], and Lee and Chen [7].

This study concerns Problem SV. Yang et al. [25] studied a single machine scheduling problem with flexible maintenance, and considered a case in which the machine should be stopped for maintenance or resetting at a fixed time \( w_1 \) during the scheduling period. Yang et al. [25] assumed that the maintenance period \([u_1, v_1]\) was specified in advance, and that the time \( w_1 \) does not fall outside the maintenance period \([u_1, v_1]\) \((w_1 \leq v_1 - u_1)\). The time \( u_1 \) \((v_1)\) is the earliest (latest) time at which the machine maintenance starts (stops). The objective was to minimize the makespan. Yang et al. [25] proved that the proposed problem is NP-hard and provided a heuristic algorithm with complexity \(O(n \log n)\). To the current author’s knowledge, Yang et al. [25] were the first authors to incorporate flexible maintenance into job scheduling.
Based on the concepts of the flexible maintenance as discussed by Yang et al. [25], this study deals with scheduling jobs and maintenance activities with a single machine and parallel machine. The maintenance time of each machine is assumed to be known and fixed in advance. Accordingly, the start time of the period of unavailability is a decision variable. Most previous research on availability constraints has focused on the complexity proof and worst-case performance analysis. To my knowledge, mixed binary integer programming (BIP) has never been applied to solve the machine scheduling problem with a flexible maintenance activity. This study presents four mixed BIP formulations including two for a single machine and the two for parallel machines.

The remainder of the paper is organized as follows. Section 2 provides preliminary information. Sections 3 and 4 address a flexible maintenance activity on the single machine and parallel machine, respectively. Section 5 evaluates the proposed models. Section 6 briefly draws conclusions and suggests future topics for research.

2. Preliminaries

This study addresses the scheduling of \( n \) independent jobs on the single machine and parallel machine problems, while accounting for a single flexible maintenance period, as described by Yang et al. [25]. The \( n \) jobs processed are represented by \( J_1, J_2, \ldots, J_n \). All machines are assumed to be simultaneously available at time zero. Furthermore, the earliest start time of maintenance is assumed to be after the processing time of any operation on \( M_k \). Moreover, only a single maintenance activity can be performed on each machine in the planning horizon; let this maintenance activity be represented by \( J_{n+1} \). The group of jobs before or after this maintenance activity is considered to be a block; a schedule can thus be viewed as two blocks of jobs separated by a single maintenance activity. Figure 1 depicts a single machine schedule with a flexible maintenance activity, where \( s_{n+1} \) and \( f_{n+1} \) are the earliest start and finish times of the maintenance activity on a single machine, respectively. The decisions under consideration are (i) when to schedule the maintenance activity, and (ii) how to order jobs to minimize mean flow time. Hence, the problem has three characteristics: (i) a machine may not be always available, because of machine maintenance; (ii) the maintenance activity is performed during the maintenance period, and (iii) only a single maintenance activity is performed.
Two types of processing cases, resumable and unresumable, are considered in relation to the single machine and parallel machine, respectively. Suppose that machine maintenance interrupts the processing of a job. If the job continues after the machine becomes available again, the problem is “resumable”. However, the problem is called “nonresumable” if the job has to be restarted from the beginning when the machine becomes available [6, 13]. Notably, “resumable” and “nonresumable” have also been called “preempt-resume” and “preempt-repeat”, respectively, in the literature [18]. This paper considers problems representing both cases such objective function as mean flow time.

For concision, the notation of Pinedo [18] is extended here to include machine availability constraints. This notation represents three fields $\alpha|\beta|\gamma$, where $\alpha$ refers to the machine environment, $\beta$ refers to the processing characteristics and constraints and $\gamma$ specifies the objective to be minimized. In particular, $\alpha = 1$ and $Rm$ denote a single machine and unrelated machines in parallel, respectively. The second field $\beta$ can represent dynamic arrivals, special precedence constraints or special availability constraints, and the third field, $\gamma$, can represent mean flow time $\bar{F}$. This study uses $\gamma = nr - fa$ in the second field to denote a resumable case with a flexible maintenance activity, in which machine $M_k$ is unavailable for a constant time $w_k$, during the maintenance period $[u_k, v_k]$ for all $k$, and a job is resumable if it cannot be finished before $M_k$ is maintained. Similarly, $\beta = nr - fa$ denotes an unresumable job and a flexible maintenance activity. Hence, this paper considers the following four problems: $1|nr - fa|\bar{F}$, $1|r - fa|\bar{F}$, $Rm|nr - fa|\bar{F}$, and $Rm|r - fa|\bar{F}$.

Given advances in computer capacity and efficient integer programming (IP) software, mathematical programming-based scheduling
research is beginning to attract increasing interest from researchers [17]. Although it is not an efficient solution method, mathematical programming is a natural way to attack machine scheduling problems [20]. Why, then, do we study IP models at all? Morton and Pentico [15] offered the following two reasons. First, certain cases with special structures will always be solvable. If the general approaches are understood, such cases will be recognized. Second, various partially relaxed equations can be solved and may be useful.

A survey of the recent development of mathematical programming formulations of scheduling problems can be found in Błażewicz et al. [2]. Most IP problems in the scheduling field involve mixed binary IP (BIP); that is, some of the variables are binary and some are continuous. The new development of mixed BIP techniques, along with the substantial progress in computer capacity, strongly impacts IP scheduling models. This paper proposes four mixed BIP models. Two mixed BIP model, assuming a nonresumable job is first provided for a single machine with a flexible maintenance activity. For parallel machine with a flexible maintenance activity, two mixed BIP models, assuming a nonresumable job are presented.

3. Single machine problems

Single machine problems are fundamentally important. They can be considered as the building blocks of more complex problems. Formulations of such problems may be used to refer to bottleneck machines or an aggregated machine system [23]. It can be checked by a simple job exchange scheme for single machine problems; \(1|r - fa|\bar{F}\) can be solved optimally by the Shortest Processing Time (SPT) algorithm (sequencing jobs in the non-decreasing order of the processing time). For the problem \(1|nr - fa|\bar{F}\), if \((v_1 - u_1) = w_1\), then the problem \(1|nr - fa|\bar{F}\) is equivalent to problem \(1|nr - a|\bar{F}\), as defined by Lee [6]. Since the special case \(1|nr - a|\bar{F}\) is NP-hard, the problem \(1|nr - fa|\bar{F}\) is also NP-hard. This section presents two mixed BIP formulations, namely, Model SNR-1 and Model SNR-2, for solving single machine scheduling problems with nonresumable jobs and a flexible maintenance activity.

3.1 Model SNR-1 for problem \(1|nr - fa|\bar{F}\)

This model uses binary variable \(z_{ij}\) to express the ‘either-or’ relationship for the non-interference restrictions. The following Model SNR-1
applies the concept of non-interference to solve single machine problems.

Minimize \( \frac{1}{n} \sum_{i=1}^{n} f_i \)

subject to

\( s_i + p_i = f_i \quad i = 1, 2, \ldots, n \) (2)

\( s_{n+1} + w_1 = f_{n+1} \) (3)

\( f_i \leq s_j + M(1 - z_{ij}) \quad 1 \leq i < j \leq n + 1 \) (4)

\( f_j \leq s_i + Mz_{ij} \quad 1 \leq i < j \leq n + 1 \) (5)

\( s_{n+1} \geq u_1 \) (6)

\( f_{n+1} \leq v_1 \) (7)

\( s_i \geq 0 \quad f_i \geq 0, i = 1, 2, \ldots, n + 1; \)

\( z_{ij} \) is binary \( 1 \leq i < j \leq n + 1 \). (8)

Constraint (1) states the mean flow time. Constraint set (2) defines the flow time of jobs, while constraint (3) describes the flow time of the maintenance activity. Constraint sets (4) and (5) impose the requirement that only one job may be processed at any time. Either \( f_i \leq s_j \) or \( f_j \leq s_i \) will hold. By incorporating binary variable \( z_{ij} \) and a very large positive number \( M \), constraints (4) and (5) together guarantee that one of these two constraints must hold while the other does not apply. Constraints (6) and (7) state the maintenance interval. Constraint set (8) specifies the non-negativity of \( s_i \) and \( f_i \), and establishes the binary restrictions for \( z_{ij} \).

3.2 Model SNR-2 for problem \( 1|nr-fa|\bar{F} \)

The binary variable \( x'_{iq} \) that model SNR-2 uses is restricted, and specifies the order in which jobs are processed on the machine. The following model SNR-2 employs the concept of one-job-one-position to describe single machine problems.

Minimize \( \frac{1}{n} \sum_{i=1}^{n} f_i \)

Subject to

\( \sum_{q=1}^{n+1} x'_{iq} = 1 \quad i = 1, 2, \ldots, n + 1 \) (10)

\( \sum_{i=1}^{n+1} x'_{iq} = 1 \quad q = 1, 2, \ldots, n + 1 \) (11)
\[ h_q + \sum_{i=1}^{n} p_i x_{iq} + w_1 x_{n+1,q} = f_q' \quad q = 1, 2, \ldots, n + 1 \] (12)

\[ f_q' \leq h_{q+1} \quad q = 1, 2, \ldots, n + 1 \] (13)

\[ h_q \geq u_i - M(1 - x_{n+1,q}) \quad q = 1, 2, \ldots, n + 1 \] (14)

\[ f_q' \leq v_i - M(1 - x_{n+1,q}) \quad q = 1, 2, \ldots, n + 1 \] (15)

\[ f_q' \leq f_i + M(1 - x_{iq}') \quad i = 1, 2, \ldots, n; q = 1, 2, \ldots, n + 1; \] (16)

\[ h_q, f_q' \geq 0 \quad q = 1, 2, \ldots, n + 1; f_i \geq 0 \quad i = 1, 2, \ldots, n; \]

\[ x_{iq}' \text{ is binary} \quad i, q = 1, 2, \ldots, n + 1. \] (17)

Constraint (9) describes the objective function. Constraint set (10) ensures that each job and the maintenance activity must be placed in a unique position, while constraint set (11) satisfies the requirement that each position of the machine must perform a unique job or the maintenance activity. Constraint set (12) is essentially definitional, while constraint set (13) enforces the precedence relationships. Furthermore, constraint sets (14) and (15) state the earliest starting time and the latest finish time of the maintenance activity. Constraint set (16) defines the flow time of each job. Finally, constraint set (17) specifies the non-negativity of \( h_q, f_q' \), and \( f_i \), and sets up the binary restrictions for \( x_{iq}' \).

4. Parallel machine problems

The scheduling of parallel machine can be considered to be a two-step process. First, jobs must be allocated among machines; second, the sequence of jobs on an individual machine must be determined. The \( R_m|nr-fa|\bar{F} \) problem can be formulated as an integer program with a special structure that makes it possible to solve the problem in polynomial time [18]. Although the \( R_m||F \) problem is a special case of the \( R_m|r-fa|\bar{F} \) problem, but the \( R_m|r-fa|\bar{F} \) problem still can use the same solution technique to solve it. For the problem \( R_m|nr-fa|\bar{F} \), if \( vk - u_k = w_k \), then the problem \( R_m|nr-fa|\bar{F} \) is equivalent to problem \( R_m|nr-a|\bar{F} \), as defined by Lee [6]. Since the special case \( R_m|nr-a|\bar{F} \) is NP-hard, the problem \( R_m|nr-fa|\bar{F} \) is also NP-hard. This section presents two mixed BIP formulations, namely, Model PNR-1 and Model PNR-2, for solving parallel machine scheduling problems with nonresumable jobs and a flexible maintenance activity.
4.1 Model PNR-1 for problem \( Rm|nr - fa|F \)

This model uses a binary variable \( z_{ij} \) to express the ‘either-or’ relationship of the non-interference restrictions for individual machines. The binary variable \( x_{ik} \) is also introduced to express the ‘yes-no’ decision concerning whether a job is performed on a certain machine. Based on the concepts of one-job-one-machine and non-interference for machines, Model PNR-1 is as follows.

Minimize \[
\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{m} f_{ik}
\]  \( (18) \)

Subject to \[
x_{n+1,k} = 1 \quad i = 1, 2, \ldots, n \]
(19)

\[
x_{n+1,k} = 1 \quad k = 1, 2, \ldots, m
\]
(20)

\[
s_{ik} + p_{ik} \leq f_{ik} + M(1 - x_{ik}) \quad i = 1, 2, \ldots, n; \quad k = 1, 2, \ldots, m
\]
(21)

\[
s_{n+1,k} + w_{k} = f_{n+1,k} \quad k = 1, 2, \ldots, m
\]
(22)

\[
f_{ik} \leq s_{jk} + M(3 - x_{ik} - x_{jk} - z_{ij}) \quad 1 \leq i < j \leq n + 1; \quad k = 1, 2, \ldots, m
\]
(23)

\[
f_{jk} \leq s_{ik} + M(2 - x_{ik} - x_{jk} - z_{ij}) \quad 1 \leq i < j \leq n + 1; \quad k = 1, 2, \ldots, m
\]
(24)

\[
s_{n+1,k} \geq u_{k} \quad k = 1, 2, \ldots, m
\]
(25)

\[
f_{n+1,k} \leq v_{k} \quad k = 1, 2, \ldots, m
\]
(26)

\[
x_{ik} \text{ is binary} \quad i = 1, 2, \ldots, n + 1; \quad k = 1, 2, \ldots, m
\]

\[
z_{ij} \text{ is binary} \quad 1 \leq i < j \leq n + 1;
\]
(27)

Constraint set (19) required that each job can be processed on only one machine. Constraint set (20) sets the number of maintenance intervals on each machine to one. Constraint sets (21) and (22) define the flow time of jobs and the maintenance activity, while constraint (18) determines the mean flow time of jobs. Constraint sets (23) and (24) impose the requirement that only one job may be processed on each machine at any time. If both \( f_i \) and \( f_j \) are processed on \( M_k \), then either \( f_{ik} \leq s_{jk} \)
or $f_{jk} \leq s_{ik}$ will hold. Incorporating binary variables $z_{ij}$, $x_{ik}$, and $x_{jk}$, and a very large positive number $M$, constraints (23) and (24) together guarantee that one of these two constraints must hold while the other does not. Constraint sets (25) and (26) state the maintenance interval for each machine, Constraint set (27) specifies the non-negativity of $s_{ik}$ and $f_{ik}$, and sets up the binary restrictions for $x_{ik}$ and $z_{ij}$.

4.2 Model PNR-2 for problem $Rm|nr-fa|\bar{F}$

The binary variable $x'_{ikq}$ that model PNR-2 uses is restricted, and specifies the order in which jobs are processed on the machine. The following model PNR-2 employs the concept of one-job-one-position to describe parallel machine problems.

Minimize $\frac{1}{n} \sum_{i=1}^{n} \sum_{k=1}^{m} f_{ik}$

Subject to

$$\sum_{k=1}^{m} \sum_{q=1}^{n+1} x''_{ikq} = 1 \quad i = 1, 2, \ldots, n$$

(29)

$$\sum_{i=1}^{n+1} x''_{ikq} \leq 1 \quad k = 1, 2, \ldots, m; \quad q = 1, 2, \ldots, n + 1$$

(30)

$$\sum_{q=1}^{n+1} x''_{ikq} \leq 1 \quad i = 1, 2, \ldots, n; \quad k = 1, 2, \ldots, m$$

(31)

$$\sum_{q=1}^{n+1} x''_{n+1,k,q} \leq 1 \quad k = 1, 2, \ldots, m$$

(32)

$$\sum_{i=1}^{n+1} x'_{i,k,q+1} \leq \sum_{i=1}^{n+1} x''_{ikq} \quad k = 1, 2, \ldots, m; \quad q = 1, 2, \ldots, n$$

(33)

$$h_{kq} + \sum_{i=1}^{n} p_{ik} x''_{ikq} + w_{k} x''_{n+1,k,q} \leq f'_{kq} + M \left(1 - \sum_{i=1}^{n+1} x''_{ikq}\right) \quad k = 1, 2, \ldots, m; \quad q = 1, 2, \ldots, n + 1$$

(34)

$$f'_{kq} \leq h_{k,q+1} \quad k = 1, 2, \ldots, m; \quad q = 1, 2, \ldots, n$$

(35)

$$h_{kq} \geq u_{k} - M(1 - x''_{n+1,k,q}) \quad k = 1, 2, \ldots, m; \quad q = 1, 2, \ldots, n + 1$$

(36)
\[ f'_{k,q} \leq v_k - M(1 - x''_{n+1,k,q}) \quad k = 1,2,\ldots,m; \]
\[ q = 1,2,\ldots,n+1 \]  
(37)

\[ f'_{k,q} \leq f_{ik} + M(1 - x'_{ikq}) \quad i = 1,2,\ldots,n; \quad k = 1,2,\ldots,m; \]
\[ q = 1,2,\ldots,n+1 \]  
(38)

\[ h_{k,q} \geq 0, \quad f'_{k,q} \geq 0 \quad k = 1,2,\ldots,n; \quad q = 1,2,\ldots,n+1; \]
\[ f_{ik} \geq 0 \quad i = 1,2,\ldots,n; \quad k = 1,2,\ldots,m; \]
\[ x'_{ikq} \text{ is binary} \quad i = 1,2,\ldots,n+1; \quad k = 1,2,\ldots,m; \]
\[ q = 1,2,\ldots,n+1. \]  
(39)

Constraint (28) describes the objective function. Constraint set (29) ensures that each job must be placed in a unique position on all machines. Constraint set (30) restricts the number of jobs on \( M_k \) at position \( q \), while constraint set (31) restricts the number of positions of \( J_i \) on \( M_k \). Constraint set (32) assumes that each machine have just one maintenance activity. Constraint set (33) restricts the position order of each machine, that is, if no job is assigned the position \( q \) on \( M_k \), then the position \( q + 1 \) on \( M_k \) must not be placed any job. Constraint set (34) is essentially definitional, while constraint set (35) enforces the precedence relationships. Furthermore, constraint sets (36) and (37) state the earliest starting time and the latest finish time of the maintenance activity on each machine. Constraint set (38) defines the flow time of each job. Finally, constraint set (39) specifies the non-negativity \( h_{k,q}, f'_{k,q}, \) and \( f_{ik} \), and sets up the binary restrictions for \( x'_{ikq} \).

5. Evaluation of the models

The four different models are evaluated in two different ways. First, the problem size complexities of the models are presented and compared. Second, each model is used to solve the real-data problems, and the computer solution times required by each model are compared.

5.1 Size complexity of the models

Size complexity indicates how large a problem is in terms of binary variables, constraints, and continuous (real) variables as a function of \( m \) and \( n \), the number of machines and jobs, respectively, in the problem. The size complexity of each of the four mixed BIP models for the single and parallel machine is presented in Table 1.
As Table 1 shows, Model SNR-1 has \((1/2)n^2 + (3/2)n + 1\) fewer binary variables, \(5n + 2\) fewer constraints, and \(n\) fewer continuous variables than that of Model SNR-2 for single machine problems. Thus, Model SNR-2 is theoretically better than the Model SNR-1. Similarly, Model PNR-2 is superior to Model PNR-1 for parallel machine problems.

<table>
<thead>
<tr>
<th>Model</th>
<th>Number of binary variables</th>
<th>Number of constraints</th>
<th>Number of continuous variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>SNR-1</td>
<td>((1/2)n^2 + (1/2)n)</td>
<td>(n^2 + 2n + 4)</td>
<td>(2n + 2)</td>
</tr>
<tr>
<td>SNR-2</td>
<td>(n^2 + 2n + 1)</td>
<td>(n^2 + 7n + 6)</td>
<td>(3n + 2)</td>
</tr>
<tr>
<td>PNR-1</td>
<td>(mn + m + (1/2)n^2)</td>
<td>(mn^2 + 2mn + 4m + n + 1)</td>
<td>(2mn + 2m)</td>
</tr>
<tr>
<td>PNR-2</td>
<td>(mn^2 + 2mn + m)</td>
<td>(mn^2 + 8mn + 5m + n + 1)</td>
<td>(3mn + 2m)</td>
</tr>
</tbody>
</table>

5.2 Computer solution time

A program coded in ILOG OPL 3.5 language is used to generate these four models and is solved with ILOG CPLEX 7.5 on a PC PIV/2.67GHz with 1G DRAM. The numerical values of problem parameters are generated according to the following scheme. Processing times are randomly generated from a discrete uniform distribution over \([5, 15]\). The test problems were divided into two sets, one consisting of problems for which optimal solutions were known by solving Models SNR-1 and SNR-2 in a reasonable time, and the other containing problems for which optimal solutions were known by solving Models PNR-1 and PNR-2. Twenty-four problem sizes were designed for the first set and 10 test problems were generated for each problem size. Thirty-six problem sizes were designed for the second set and 10 test problems were generated for each problem size. Thus, a total of 600 \((24 \times 10 + 36 \times 10)\) were randomly generated and tested.

5.2.1 Single machine problems

In order to evaluate the computational efficiency of Models SNR-1 and SNR-2, we generate several groups of random problems as follows:

1. \(n\) is equal to 6, 7, 8, or 9.
2. \(p_i\) is selected from a discrete uniform distribution (DU) over \([5, 15]\).
(3) \( w_1 \) is selected from another DU over \([1, 15]\) or \([16, 30]\).

(4) \( u_1 \) is equal to the integer part of \( 0.25 \times \sum_{i=1}^{n} p_i, 0.5 \times \sum_{i=1}^{n} p_i, \) or \( 0.75 \times \sum_{i=1}^{n} p_i, \) with restriction \( u_1 \geq \max_{1 \leq i \leq n} p_i. \)

(5) \( v_1 \) is equal to \( u_1 + 30. \)

Table 2 summarizes the computational results for the single machine problems. Notably, the efficiency of mixed BIP models is reported based on the average CPU time (in seconds) and number of times in 10 problems Model SNR-1 used less time than Model SNR-2. Table 2 yields the following observations.

1. Model SNR-1 is superior to Model SNR-2 in the average CPU time and the number of times in 10 problems Model SNR-1 used less time than Model SNR-2.

2. The average CPU time of Models SNR-1 and SNR-2 decreases with increasing \( w_1 \).

3. The efficiency of Model SNR-2 increases with decreasing \( u_1 \). Model SNR-1 is efficiency if \( u_1 \) is high and \( w_1 \) is low. But Model SNR-1 is inefficiency if \( u_1 \) and \( w_1 \) are high.

<table>
<thead>
<tr>
<th>( n )</th>
<th>( u_1 )</th>
<th>( w_1 )</th>
<th>Average CPU time of the model (seconds)</th>
<th>SNR-1</th>
<th>SNR-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>0.25T</td>
<td>[1, 15]</td>
<td>0.3751</td>
<td>1.9655</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>0.1843</td>
<td>1.2233</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.5T</td>
<td>[1, 15]</td>
<td>0.2905</td>
<td>2.6313</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>0.2594</td>
<td>1.6843</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.75T</td>
<td>[1, 15]</td>
<td>0.1813</td>
<td>2.8343</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>0.2468</td>
<td>1.6485</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>0.25T</td>
<td>[1, 15]</td>
<td>13.6626</td>
<td>90.5921</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>3.9673</td>
<td>42.8452</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.5T</td>
<td>[1, 15]</td>
<td>10.2673</td>
<td>102.9109</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>5.0624</td>
<td>54.0688</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>0.75T</td>
<td>[1, 15]</td>
<td>6.1172</td>
<td>110.4828</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>7.3499</td>
<td>77.6767</td>
<td>10</td>
</tr>
</tbody>
</table>

(Contd. Table 2)
Table 3 summarizes the computational results for the parallel machine problems. The efficiency of mixed BIP models is reported based on the average CPU time (in seconds) and number of times in 10 problems Model PNR-1 used less time than Model PNR-2. Table 3 yields the following observations.

<table>
<thead>
<tr>
<th>n</th>
<th>υ₁</th>
<th>w₁</th>
<th>Average CPU time of the model (seconds)</th>
<th>SNR-1 &lt; SNR-2†</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.25T</td>
<td>[1,15]</td>
<td>24.3858</td>
<td>97.6516</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16,30]</td>
<td>5.4471</td>
<td>48.8137</td>
</tr>
<tr>
<td>0.5T</td>
<td>[1,15]</td>
<td>13.9438</td>
<td>111.7937</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16,30]</td>
<td>6.2500</td>
<td>60.7500</td>
<td>10</td>
</tr>
<tr>
<td>0.75T</td>
<td>[1,15]</td>
<td>10.4127</td>
<td>149.7326</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16,30]</td>
<td>9.5815</td>
<td>93.0106</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>0.25T</td>
<td>[1,15]</td>
<td>124.4328</td>
<td>862.3204</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16,30]</td>
<td>65.6687</td>
<td>532.1671</td>
</tr>
<tr>
<td>0.5T</td>
<td>[1,15]</td>
<td>116.7703</td>
<td>1307.1578</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16,30]</td>
<td>67.2596</td>
<td>610.1606</td>
<td>10</td>
</tr>
<tr>
<td>0.75T</td>
<td>[1,15]</td>
<td>78.7047</td>
<td>1541.2298</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16,30]</td>
<td>107.4406</td>
<td>983.0798</td>
<td>10</td>
</tr>
</tbody>
</table>

† T = \sum_{i=1}^{n} p_i

‡ Number of times in 10 problems Model SNR-1 used less time than Model SNR-2

5.2.2 Parallel machine problems

In order to evaluate the computational efficiency of Models PNR-1 and PNR-2, we generate several groups of random problems as follows:

(1) \((n, m)\) is equal to \((5, 2), (5, 5), (6, 2), (6, 4), (6, 6)\), or \((7, 7)\).

(2) \(p_{ik}\) is selected from a discrete uniform distribution (DU) over \([5, 15]\).

(3) \(w_k\) is selected from another DU over \([1, 15]\) or \([16, 30]\).

(4) \(u_k\) is selected from a DU over \(\left[ u_{kn} \times \left( \sum_{i=1}^{n} \min_{1 \leq i \leq m} p_{ik}/m \right), u_{kn} \times \left( \sum_{i=1}^{n} \min_{1 \leq i \leq m} p_{ik}/m \right) \right] \), with restriction \(u_k \geq \max_{1 \leq i \leq n} p_{ik}\), where \(u_{kn}\) equals \(0.25, 0.5, \) or \(0.75\).

(5) \(v_k\) is equal to \(u_k + 30\).
(1) Model PNR-1 is superior to Model PNR-2 in the average CPU time and the number of times in 10 problems Model PNR-1 used less time than Model PNR-2.

(2) The average CPU time of Models PNR-1 and PNR-2 decreases with increasing $w_k$.

(3) The efficiency of Models PNR-1 and PNR-2 decreases with increasing $m$.

(4) The relation of the $u_k$ value and efficiency of Model PNR-1 and PNR-2 is not apparent.

<table>
<thead>
<tr>
<th>$n$</th>
<th>$m$</th>
<th>$u_k$</th>
<th>$w_k$</th>
<th>Average CPU time of the model (seconds)</th>
<th>PNR-1</th>
<th>PNR-2</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>2</td>
<td>0.25</td>
<td>[1, 15]</td>
<td>0.3733</td>
<td>2.7532</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>0.3904</td>
<td>2.3000</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td>[1, 15]</td>
<td>0.3717</td>
<td>2.8157</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>0.3219</td>
<td>2.3045</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.75</td>
<td>[1, 15]</td>
<td>0.2499</td>
<td>2.9501</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>0.2516</td>
<td>2.1843</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>0.25</td>
<td>[1, 15]</td>
<td>0.1998</td>
<td>1.9375</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>0.2141</td>
<td>1.6796</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td>[1, 15]</td>
<td>0.2921</td>
<td>1.9172</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>0.1562</td>
<td>1.7922</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.75</td>
<td>[1, 15]</td>
<td>0.2608</td>
<td>1.5171</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>0.2826</td>
<td>1.4768</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2</td>
<td>0.25</td>
<td>[1, 15]</td>
<td>2.4640</td>
<td>23.5859</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>1.9250</td>
<td>20.6267</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.5</td>
<td>[1, 15]</td>
<td>2.4593</td>
<td>23.9674</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>1.9470</td>
<td>23.2578</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0.75</td>
<td>[1, 15]</td>
<td>1.9702</td>
<td>23.6969</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>2.2735</td>
<td>21.7578</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0.25</td>
<td>[1, 15]</td>
<td>0.6639</td>
<td>234.5860</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[16, 30]</td>
<td>0.6062</td>
<td>176.2846</td>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

(Contd. Table 3)
### Table

<table>
<thead>
<tr>
<th>$n$</th>
<th>$m$</th>
<th>$u_k$</th>
<th>$w_k$</th>
<th>Average CPU time of the model (seconds)</th>
<th>PNR-1 &lt; PNR-2‡</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PNR-1</td>
<td>PNR-2</td>
</tr>
<tr>
<td>0.5</td>
<td>[1, 15]</td>
<td></td>
<td>0.8330</td>
<td>287.5030</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16, 30]</td>
<td></td>
<td>0.4813</td>
<td>140.4422</td>
<td>10</td>
</tr>
<tr>
<td>0.75</td>
<td>[1, 15]</td>
<td></td>
<td>0.7405</td>
<td>278.3532</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16, 30]</td>
<td></td>
<td>0.4938</td>
<td>143.2499</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>0.25</td>
<td>[1, 15]</td>
<td>1.7184</td>
<td>537.9627</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16, 30]</td>
<td></td>
<td>1.1780</td>
<td>432.5298</td>
<td>10</td>
</tr>
<tr>
<td>0.5</td>
<td>[1, 15]</td>
<td></td>
<td>1.3846</td>
<td>609.9420</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16, 30]</td>
<td></td>
<td>1.0797</td>
<td>372.1608</td>
<td>10</td>
</tr>
<tr>
<td>0.75</td>
<td>[1, 15]</td>
<td></td>
<td>1.4002</td>
<td>476.5796</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16, 30]</td>
<td></td>
<td>0.9764</td>
<td>434.1638</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>0.25</td>
<td>[1, 15]</td>
<td>12.9388</td>
<td>9957.2485</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16, 30]</td>
<td></td>
<td>10.2610</td>
<td>6220.0642</td>
<td>10</td>
</tr>
<tr>
<td>0.5</td>
<td>[1, 15]</td>
<td></td>
<td>18.0564</td>
<td>10606.3532</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16, 30]</td>
<td></td>
<td>9.2684</td>
<td>7138.0737</td>
<td>10</td>
</tr>
<tr>
<td>0.75</td>
<td>[1, 15]</td>
<td></td>
<td>12.2891</td>
<td>8504.1999</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>[16, 30]</td>
<td></td>
<td>7.7751</td>
<td>6672.5719</td>
<td>10</td>
</tr>
</tbody>
</table>

† $u_k$ is selected from a DU over 
$$
0.25 \times \left( \sum_{i=1}^{n} \min_{1 \leq k \leq m} p_{ik} / m \right) ,
0.25 \times \left( \sum_{i=1}^{n} \max_{1 \leq k \leq m} p_{ik} / m \right)
$$

‡ Number of times in 10 problems Model PNR-1 used less time than Model PNR-2

### 6. Conclusion

This work proposes four mixed BIP models for the single machine and parallel machine scheduling problems, with flexible maintenance on each machine. These four models include two for the single machine and the other two for the parallel machine. Models SNR-1 and SNR-2 assume that jobs are nonresumable on the single machine. Models PNR-1 and PNR-2 assume that jobs are nonresumable on the parallel machine. This study uses ILOG CPLEX 7.5 and OPL 3.5 software to verify the accuracy of the above four models. The computational results show that Models SNR-1 and PNR-1 outperform Model SNR-2 and PNR-2, respectively.

Future research should address problems with multiple unavailability intervals and the different shop environments, including flow-
shop and job-shop. Problems with other performance measures, such as minimum makespan, mean tardiness, and multi-criteria measures, should also be studied. Meta-heuristics should be used to solve such problems.

Acknowledgements. This research was partially supported by the National Science Council of Taiwan, Republic of China, under contract NSC 93-2213-E-269-003.

References


Received October, 2004