Minimising makespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes

S.H. Chung\textsuperscript{a}, Y.T. Tai\textsuperscript{a} & W.L. Pearn\textsuperscript{a}

\textsuperscript{a} Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu, Taiwan, ROC

Published online: 30 Jun 2009.

To cite this article: S.H. Chung, Y.T. Tai & W.L. Pearn (2009) Minimising makespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes, International Journal of Production Research, 47:18, 5109-5128, DOI: 10.1080/00207540802010807

To link to this article: http://dx.doi.org/10.1080/00207540802010807

Taylor & Francis makes every effort to ensure the accuracy of all the information (the “Content”) contained in the publications on our platform. However, Taylor & Francis, our agents, and our licensors make no representations or warranties whatsoever as to the accuracy, completeness, or suitability for any purpose of the Content. Any opinions and views expressed in this publication are the opinions and views of the authors, and are not the views of or endorsed by Taylor & Francis. The accuracy of the Content should not be relied upon and should be independently verified with primary sources of information. Taylor and Francis shall not be liable for any losses, actions, claims, proceedings, demands, costs, expenses, damages, and other liabilities whatsoever or howsoever caused arising directly or indirectly in connection with, in relation to or arising out of the use of the Content.

This article may be used for research, teaching, and private study purposes. Any substantial or systematic reproduction, redistribution, reselling, loan, sub-licensing, systematic supply, or distribution in any form to anyone is expressly forbidden. Terms & Conditions of access and use can be found at http://www.tandfonline.com/page/terms-and-conditions
Minimising makespan on parallel batch processing machines with non-identical ready time and arbitrary job sizes

S.H. Chung*, Y.T. Tai and W.L. Pearn

Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu, Taiwan, ROC

(Received 31 July 2007; final version received 16 December 2007)

This paper considers the parallel batch processing machine scheduling problem which involves the constraints of unequal ready times, non-identical job sizes, and batch dependent processing times in order to sequence batches on identical parallel batch processing machines with capacity restrictions. This scheduling problem is a practical generalisation of the classical parallel batch processing machine scheduling problem, which has many real-world applications, particularly, in the aging test operation of the module assembly stage in the manufacture of thin film transistor liquid crystal displays (TFT-LCD). The objective of this paper is to seek a schedule with a minimum total completion time for the parallel batch processing machine scheduling problem. A mixed integer linear programming (MILP) model is proposed to optimise the scheduling problem. In addition, to solve the MILP model more efficiently, an effective compound algorithm is proposed to determine the number of batches and to apply this number as one parameter in the MILP model in order to reduce the complexity of the problem. Finally, three efficient heuristic algorithms for solving the large-scale parallel batch processing machine scheduling problem are also provided.

Keywords: parallel batch; scheduling; unequal ready time

1. Introduction

The existing and growing importance of parallel batch processing machines demands a solution to its scheduling problem in order to improve efficiency of production. In this paper, a parallel batch processing machine scheduling problem with a minimum makespan criterion is presented, which has many real-world applications, particularly in the aging test operation in the manufacture of thin film transistor liquid crystal displays (TFT-LCD). On the batch processing machines in aging test operation, multiple jobs composed of different product families can be simultaneously processed as a batch. The batch processing times and batch ready times are dependent on the longest processing time and the latest ready time of all the jobs in each batch, respectively. Notably, the parallel batch processing machine scheduling problem is a multi-dimensional problem, which involves the constraints of unequal ready times, non-identical job sizes, limited machine capacity, and batch dependent processing times; the relationships among these dimensions is depicted as Figure 1. The constraints of unequal ready times and non-identical job sizes affect the determination of the number of batches which is based on the limited machine
capacity and which then determines the batch processing times. Whenever batch formations are altered, the batch processing times are consequently varied and the batch sequence needs to be rescheduled in order to minimise the makespan. Since the parallel batch processing machine scheduling problem involves batch formation and scheduling with real-world constraints, it is an intractable problem for industrial planners and theoretical researchers. Therefore, the development of efficient algorithms to form appropriate batches and to arrange a suitable schedule for a parallel batch processing machine scheduling problem is difficult but critical.

The scheduling problem investigated in this paper has as its motivating cause the aging test operation in TFT-LCD manufacturing process. The TFT-LCD manufacturing process consists of four basic stages: TFT array fabrication, colour filter fabrication, LCD assembly, and module assembly, as shown in Figure 2. TFT array and colour filter fabrications are similar to the semiconductor wafer fabrication, and their process steps are also characterised by re-entrant flow. The LCD assembly simultaneously attaches the TFT and colour filter and fills the gap between them with liquid crystal. The final stage, module assembly, involves six steps in which the customer-specified components are assembled into the cells:

1. The COG (chip on glass) process.
2. The attachment of the flexible printed circuit board (FPC).
3. The bonding of the printed circuit board (PCB).
4. The assembly of the backlight.
5. The aging test.
6. The inspection (as shown in Figure 3).
At the end of the module assembly process, the aging test is undertaken by the only batch server in the whole process to put the assembled modules with different product families and unequal ready times into high temperature parallel batch processing machines. These machines then run aging tests for several hours to detect defects before the jobs are delivered; this batch production type is referred to as compatible product families. The processing times in the aging test operation are dependent on the longest processing time of all the jobs in the batch; they are generally longer than other serial processing operations, in particular for LCD TV products which usually take six hours to complete the aging test. Since the jobs with unequal ready times and long batch dependent processing times are processed at the end of the module assembly, the development of scheduling algorithms for the parallel batch processing machine scheduling problem is essential with a minimum makespan criterion.

This paper investigates the parallel batch processing machine scheduling problem, which involves constraints of unequal ready times, non-identical job sizes, limited machine capacity, and batch dependent processing times, is a variation of the classical parallel batch processing machine scheduling problem considered by Lee and Uzsoy (1999) and Chang et al. (2004). Due to fiercer competition in the global TFT-LCD industry, the manufacturing lead time in the throughout process for the module assembly stage has been reduced to only one or two days. This has resulted in a lower WIP (work in process) volume being maintained on the shop floor in order to accelerate the process. Due to the low level of WIP, batches need to be formed by collecting non-identical size jobs with unequal ready times at the aging test operation. The characteristic of unequal ready times makes the parallel batch processing machine scheduling problem more complicated because it is sometimes advantageous to process a non-full batch to avoid excessive delays in waiting for jobs with later ready times (Mönch et al. 2005). In other words, processing a full batch with late-ready-time jobs may cause a large makespan. However, taking the information of unequal ready times into consideration can lead to better decisions than those based only on the equal ready times of the system (Mönch et al. 2005). In this paper, it is assumed that the ready times of jobs are known before the determination of the parallel batch processing machines schedule.

To the best knowledge of the authors, the parallel batch processing machine scheduling problem with unequal ready times and non-identical job sizes has not been considered by the other researchers in this area. This problem can be represented by $P|\text{batch, } r_j, \text{ compatible}|C_{\text{max}}$, and involves batch formation and scheduling simultaneously. In this paper, the scheduling problem is formulated as a mixed integer linear programming (MILP) model to minimise the makespan. The programming model considers machine capacity restrictions, unequal ready times, non-identical job sizes, and batch-dependent processing times, in order to reflect real situations more accurately. A compound
MILP-based algorithm is developed to determine the number of batches and to apply this number as one parameter in running the proposed MILP model. Furthermore, three efficient heuristic algorithms are also proposed. The best solution selected from the result of the MILP model and that of the compound algorithm is used here as a convenient reference point to assess the accuracy of the heuristic solutions. To demonstrate the effectiveness and efficiency of all the proposed algorithms, a set of testing problems is explored and a real-world problem is taken from a module assembly shop floor in a TFT-LCD factory located in the Science-based Industrial Park in Hsinchu, Taiwan.

This paper is organised as follows. Section 2 provides the literature review, Section 3 presents the mathematical model for the problem and Section 4 presents the compound MILP-based algorithm. Three heuristic algorithms are presented in Section 5 and the computational comparisons are offered in Section 6. Finally, Section 7 provides the conclusions.

2. Literature review

In recent years, much research has focused on providing solutions to the batch processing machine (BPM) scheduling problems on a single or parallel batch processing machines. Two models of a single BPM scheduling problem with the characteristics of a common job processing time, unequal ready times and unequal due dates have been proposed by Morton and Pentico (1993). The first model concerns the one-class case that allows any two jobs with the same processing times to be simultaneously processed. Ikura and Gimple (1986) have provided efficient algorithms for this model to find a feasible schedule in order to minimise the final completion time under the assumption that ready times and due dates are agreeable, i.e. $r_i > r_j$ implies $d_i \geq d_j$. The second model considers the multi-class case that only allows jobs of the same product type to be simultaneously processed. This model is also referred to as incompatible product families. However, the parallel batch processing machine scheduling problem investigated in this paper considers the batch processing of compatible product families. It is assumed that jobs belonging to different product families may be simultaneously processed. In problems of this type, the batch processing time is computed by the longest job processing time in that batch.

The literature regarding the BPM scheduling problems on a single or parallel batch processing machines with compatible product families is shown in Table 1. However, their solution procedures have assumed that the ready times for the parallel batch processing machines are equal. This assumption prevents the developed procedure from being directly applied to the parallel batch processing machine scheduling problem investigated in this paper because the latter involves unequal ready times. Therefore, this paper arises from the need in industry to consider jobs with unequal ready times, non-identical job sizes and processing times, which are processed on identical parallel batch processing machines.

The first researchers to address the batch processing scheduling problem arising in a burn-in oven of the final test in the semi-conductor industry are Lee et al. (1992). They used dynamic programming-based algorithms and heuristics for a number of performance measures, such as maximum tardiness ($T_{\text{max}}$), the number of tardy jobs ($\Sigma U_i$), and maximum lateness time ($L_{\text{max}}$) on a single batch processing machine. They have also presented heuristics for the parallel batch processing machine scheduling problem.
with the minimum makespan \((C_{\text{max}})\) and maximum lateness time \((L_{\text{max}})\) criteria. They have explored the area of scheduling batch processing machines and offered a classification of complexity for the investigated problems.

For batch processing machine scheduling problems with compatible product family characteristics, the single batch processing machine problem (Uzsoy 1994, Erramilli and Mason 2006, Kashan et al. 2006) and the two batch processing machines in a flow shop (Damodaran and Srihari 2004) do not take unequal ready times into consideration. Uzsoy (1994) investigated the single batch processing machine scheduling problem with non-identical job sizes to minimise the total completion times \((\sum C_i)\) of the jobs and makespan. He has also provided bin-packing-based heuristics for minimising makespan and has used the branch and bound approach to minimise the total completion times. He also developed effective heuristics for the criteria of minimum makespan and minimum total completion time. Erramilli and Mason (2006) have investigated the multiple orders per job (MOJ) problem on a single batch processing machine. They grouped different customer orders into jobs and combined jobs into batches and scheduled them on a single batch processing machine to minimise the total weighted tardiness \((\sum w_i T_i)\) of orders. Damodaran and Srihari (2004) have proposed two mathematical models with the minimum makespan criterion to schedule batches of jobs on two machines in a flow shop. Kashan et al. (2006) has addressed the need to minimise makespan by employing two different genetic algorithms (GAs) for scheduling jobs with non-identical sizes on a single batch processing machine. Unfortunately, all the above models do not consider the unequal ready time that is a common phenomenon in module assembly factories.

Although Lee and Uzsoy (1999), Sung and Choung (2000), Sung et al. (2002), and Van Der Zee (2004) have considered the characteristic of unequal ready times, they limited their applications to a single batch processing machine and an identical job size. Lee and Uzsoy (1999) have provided efficient heuristics to solve the scheduling problem arising in the final test phase of semiconductor manufacturing. To minimise the maximum

---

### Table 1. The literature related to the batch processing machine scheduling problem with compatible product families.

<table>
<thead>
<tr>
<th>References</th>
<th>Shop type</th>
<th>Ready time</th>
<th>Job size</th>
<th>Performance criterion</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lee et al. (1992)</td>
<td>Single machine/parallel</td>
<td>Unequal/equal</td>
<td>Identical/identical</td>
<td>(T_{\text{max}}, \sum U_i, L_{\text{max}}/C_{\text{max}}, L_{\text{max}})</td>
</tr>
<tr>
<td>Uzsoy (1994)</td>
<td>Single machine</td>
<td>Equal</td>
<td>Non-identical</td>
<td>(C_{\text{max}}, \sum C_i)</td>
</tr>
<tr>
<td>Erramilli and Mason (2006)</td>
<td>Single machine</td>
<td>Equal</td>
<td>Non-identical</td>
<td>(C_{\text{max}})</td>
</tr>
<tr>
<td>Damodaran and Srihari (2004)</td>
<td>Two machines in a flow shop</td>
<td>Equal</td>
<td>Non-identical</td>
<td>(C_{\text{max}})</td>
</tr>
<tr>
<td>Kashan et al. (2006)</td>
<td>Single machine</td>
<td>Equal</td>
<td>Non-identical</td>
<td>(C_{\text{max}})</td>
</tr>
<tr>
<td>Lee and Uzsoy (1999)</td>
<td>Single machine</td>
<td>Unequal</td>
<td>Identical</td>
<td>(C_{\text{max}})</td>
</tr>
<tr>
<td>Sung and Choung (2000)</td>
<td>Single machine</td>
<td>Unequal/equal</td>
<td>Identical</td>
<td>(C_{\text{max}})</td>
</tr>
<tr>
<td>Sung et al. (2002)</td>
<td>Single machine</td>
<td>Unequal</td>
<td>Identical</td>
<td>(C_{\text{max}})</td>
</tr>
<tr>
<td>Van Der Zee (2004)</td>
<td>Single machine</td>
<td>Dynamic arrival</td>
<td>Identical</td>
<td>(F_i)</td>
</tr>
<tr>
<td>Chang et al. (2004)</td>
<td>Parallel machines</td>
<td>Equal</td>
<td>Non-identical</td>
<td>(C_{\text{max}})</td>
</tr>
<tr>
<td>This paper</td>
<td>Parallel machines</td>
<td>Unequal</td>
<td>Non-identical</td>
<td>(C_{\text{max}})</td>
</tr>
</tbody>
</table>
completion time on a single batch processing machine with dynamic job arrivals, they
designed three algorithms (GRLPT, DELAY, and UPDATE) to find the approximate
solutions. Sung and Choung (2000) have presented a branch-and-bound algorithm and
several heuristics to solve the static and dynamic cases on a single batch processing
machine. Their objective was also to minimise the makespan of all jobs. Sung et al. (2002)
have considered a single batch processing machine with job families and dynamic job
arrivals. The performance measure used to evaluate a schedule is the minimum makespan.
Van Der Zee (2004) has also presented the dynamic control of a batch processing machine;
his objective was to find the minimum average flow time per product in the presence of
compatible product families.

More recently, the parallel batch processing machine scheduling problem with equal
ready time, non-identical job sizes, and the compatible product family characteristics is
considered by Chang et al. (2004). They have provided a mathematical model and
developed an algorithm based on simulated annealing (SA) approach to minimise
makespan. However, they have not included the unequal ready times in their model. At the
time this paper was being written, the authors were not aware of any other studies of the
parallel batch processing machine scheduling problem with unequal ready time,
non-identical job size, and compatible product family characteristics.

3. A mixed integer programming formulation

The mixed integer linear programming (MILP) model for the parallel batch processing
machine scheduling problem with unequal ready times and non-identical job sizes in order
to minimise the makespan is formulated in this section and is referred to as Model P. A set
of jobs is given to be processed in batches by identical parallel machines. Let the total
number of jobs be denoted by \( N \) and the number of batches be denoted by \( B \). In this paper,
however, an individual job cannot be split into different batches due to the inconvenience
to practical management that might result. Let machine group \( M = \{ m_k | k = 1, 2, \ldots, K \} \),
contain the \( K \) parallel batch processing machines. Due to the fact that the unequal ready
times and batch dependent processing times are considered, they are associated with job
\( j \) and have a processing time denoted by \( p_j \) and a ready time denoted by \( r_j \). The batch
processing time may vary, depending on the composite jobs. Term \( pt_b \) is the longest
processing time of all the jobs processed simultaneously in the \( b \)th batch, and it represents
the batch processing time. The batch ready time is also the latest ready times of those
composite jobs. Each job has a non-identical job size \( (s_j) \). A batch can be processed on a
machine on the condition that the accumulated size of those jobs in that batch does not
exceed the machine’s capacity (a maximum number of pieces can be processed
simultaneously on a machine) \( (S) \). Each job in its associated batch is a candidate that
is processed without pre-emption on one machine. The parallel batch processing machine
scheduling problem is to form batches appropriately as well as to find a schedule for those
batches that satisfies the ready time restrictions without violating the machine capacity
constraints, while also achieving the objective of minimising makespan. Initially, it is
assumed that each job will be contained in an individual batch \( (B = N) \). However, after the
mathematical model is solved, it may be that the better solution might require combining
more than one job into one batch. Hence, by using the proposed MILP model, the number
of batches required will be self-evident. Before the MILP model (Model P) is presented, the
notations used in the formulation are listed below.
Indices

\( j \) \quad \text{job index, } j = 1, 2, \ldots, N, \\
\( b \) \quad \text{batch index, } b = 1, 2, \ldots, B, \\
\( k \) \quad \text{machine index, } k = 1, 2, \ldots, K.

Decision variables

\[
\begin{align*}
x_{j bk} &= \begin{cases} 
1 & \text{if job } j \text{ is assigned to batch } b \text{ on machine } m_k, \\
0 & \text{otherwise};
\end{cases} \\
y_{bb'/k} &= \begin{cases} 
1 & \text{if batch } b' \text{ is scheduled following batch } b \text{ on machine } m_k, \\
0 & \text{otherwise};
\end{cases} \\
z_{bk} &= \begin{cases} 
1 & \text{if batch } b \text{ is assigned to machine } m_k, \\
0 & \text{otherwise};
\end{cases} \\
t_{bk} &= \text{the starting time of batch } b \text{ to be processed on machine } m_k,
\end{align*}
\]

\( C_{\text{max}} \) \quad \text{the maximum completion time (makespan).}

Model P

Minimise \( C_{\text{max}} + \mu \sum_{b=1}^{B} \sum_{k=1}^{K} z_{bk} \) \hspace{1cm} (1)

subject to

\[
\begin{align*}
\sum_{b=1}^{B} \sum_{k=1}^{K} x_{j bk} &= 1, \quad \text{for all } j, \\
\sum_{k=1}^{K} z_{bk} &\leq 1, \quad \text{for all } b, \\
\sum_{j=1}^{N} x_{j bk} &\leq Q_1 z_{bk}, \quad \text{for all } b, k, \\
\sum_{j=1}^{N} s_j x_{j bk} &\leq S', \quad \text{for all } b, \\
p t_b &\geq p_j \times x_{j bk}, \quad \text{for all } j, b, k, \\
C_{\text{max}} &\geq t_{bk} + p t_b, \quad \text{for all } b, k, \\
t_{bk} &\geq r_j x_{j bk}, \quad \text{for all } j, b, k, \\
t_{bk} + p t_b - t_{b'/k} + Q_2 (y_{bb'/k} - 1) &\leq 0, \quad \text{for all } b, k, b' \neq b, \\
(y_{bb'/k} + y_{b/bk}) - Q_2 (z_{bk} + z_{b'/k} - 2) &\geq 1, \quad \text{for all } b, k, b' \neq b, \\
(y_{bb'/k} + y_{b/bk}) + Q_2 (z_{bk} + z_{b'/k} - 2) &\leq 1, \quad \text{for all } b, k, b' \neq b,
\end{align*}
\]
The bi-functional objective of Equation (1) of Model P is to minimise the maximum completion time and the number of batches. The former is the main objective for the scheduling problem and the latter is the subsidiary one in order to reduce the complexity of the batch sequence. Therefore, the term \( \mu \) is a constant, which is chosen to be a sufficiently small value which cannot affect the makespan. Constraint (2) guarantees that each job is assigned to one batch and processed on exactly one machine. Constraint (3) ensures that each batch is either processed once by one machine or not at all. Constraint (4) is a contingent constraint. That is, if some jobs are assigned to batch \( b \) on machine \( m_k \) (\( x_{jbk} = 1 \)), then batch \( b \) should be assigned to machine \( m_k \) (\( z_{bk} = 1 \)). Term \( Q_1 \) is a constant and is greater than the total number of jobs (\( N \)). Constraint (5) is the batch size constraint, which requires that the sum of all the pieces of each job contained in each batch on one machine be simultaneously processed and within the maximum machine capacity. Constraint (6) ensures that the processing time of each batch is the longest processing time of all the jobs simultaneously processed in a batch. Constraint (7) is the maximum completion time (makespan) and is always greater than or equal to the sum of the starting and processing times for each batch. Constraint (8) indicates that the starting time of each batch is greater than or equal to the ready time of that batch. The ready time of a batch is the latest ready time of all the jobs clustered in a batch. Term \( Q_2 \) is the chosen constant as it is sufficiently large in value to satisfy \( y_{bb'k} = 0 \) or 1 which is required for constraints (9) to (14). Constraint (9) ensures the satisfaction of the inequality in \( t_{bk} + p_t_b \leq t_{bk} \) if batch \( b \) precedes batch \( b' (y_{bb'k} = 1) \). Constraints (10) to (14) are the precedence constraints provided by Pearn et al. (2002). Constraints (10) and (11) guarantee that one batch should precede another \( (y_{bb'k} + y_{bb'k} = 1) \) if two batches are scheduled on the same machine \( (z_{bk} + z_{b'k} - 2 = 0) \). It should be noted that the precedent relationships \( (y_{bb'k}) \) between batches \( b \) and \( b' \) on machine \( m_k \) may not be limited to direct ones. Constraint (12) ensures that the precedence variables \( y_{bb'k} \) and \( y_{b'b'k} \) should be set to zero \( (y_{bb'k} + y_{b'b'k} \leq 0) \) if any two batches \( b \) and \( b' \) are not scheduled on the machine \( m_k \). Constraint (13) indicates the situation in which batch \( b \) is scheduled on machine \( m_k \) and batch \( b' \) is scheduled on another machine \( (z_{b'k} - z_{bk} + 1 = 0) \) and constraint (14) indicates the situation in which batch \( b' \) is scheduled on machine \( m_k \) and batch \( b \) is scheduled on another machine \( (z_{bk} - z_{b'k} + 1 = 0) \). Constraints (15) to (17) indicate that \( x_{jbk}, y_{bb'k}, \) and \( z_{bk} \) are binary integer variables. Finally, constraint (18) indicates that the makespan is greater
than or equal to zero. The total number of variables is $NBK + B^2K + BK + 1$ and the total number of constraint equations is

$$(9/2)B^2K + 3NBK - (3/2)BK + N + 2B + 1,$$

where $B$ is the number of batches.

To demonstrate the applicability of Model P, an illustrative example is considered. The example involves two parallel batch processing machines ($m_1$ and $m_2$) and seven independent jobs with various sizes and processing times, which are ready for different starting times, as shown in Table 2. The seven jobs should be clustered into an appropriate number of batches. Those batches are scheduled on the two identical machines. The batch processing time is not affected by the machine processing it, but is dependent on the batch formation. The maximum number of pieces of one batch in a machine is set at 450 pieces in this example.

Model P is implemented using the software CPLEX OPL 3.5 to solve the seven-job example. For the example investigated, the model contains 225 variables and 736 equations. The MILP model is run on a Pentium IV 3.2GHz PC to obtain optimal solutions. The four batches, $b_1$, $b_2$, $b_3$, and $b_4$, are actually formed and their batch processing times are 90, 190, 200, and 290, respectively. Job 3 is grouped into $b_1$ and scheduled on Machine 1. Jobs 1, 2, and 4 are grouped into $b_2$ and scheduled on Machine 2. Moreover, jobs 6 and 7 are grouped into $b_3$ which is scheduled on Machine 2 and processed after $b_2$. Job 5 is the only one in $b_4$ and it is scheduled on Machine 1 and processed after $b_1$. The makespan of the example is 430 as shown in Figure 4 and the computational time is 1041.3 CPU seconds.

### 4. Compound MILP-based algorithm

A compound MILP-based algorithm (CMA) improves the efficiency of the MILP model proposed in Section 3. For the parallel batch processing machine scheduling problem investigated in this paper, the number of batches and the composite jobs for each batch determine the solution quality and efficiency. It is obvious that the probable numbers of batches fall into the range, $1 \leq B \leq N$, in an $N$-job scheduling problem. The lower and upper bound values of $B$ are obtained when all the $N$ jobs are combined with one batch and each job is contained in an individual batch, respectively. However, exploring all the possible numbers of batches would increase the run time to obtain the optimal solution.
Therefore, a relaxed MILP model (Model N) focuses on the batch formation and relaxes all relative precedence constraints to obtain the lower bound of the number of batches. Model N is then provided to reduce the search space for the parallel batch processing machine scheduling problem and is applied in the CMA. In Model N, the precedence variable $y_{bb^0k}$ and precedence constraints (constraints (9) to (14)) are removed. Model N uses $\sum_{b=1}^{B} \sum_{k=1}^{K} z_{bk}$ as its objective function. Model N starts from a makespan lower bound $C_{max}'$, which is greater than the longest processing time of all the jobs.

In this section, the compound MILP-based algorithm (CMA) is developed. The algorithm essentially consists of two phases. Phase I applies Model N to obtain the lower bound of the number of batches, which can serve as a referenced batch number in Model P. As mentioned earlier, it is sometimes advantageous to assign one more batch than the referenced batch number obtained from Phase I of the CMA to avoid excessive delays in waiting for the next scheduled late ready time job. Therefore, in Phase II, the solution of the subsequent batch number is checked. The algorithm is stated as follows and the flow chart is depicted in Figure 5.

**Phase I: The number of batches is determined and applied to Model P**

**Step 1:** Solve the following relaxed MILP model (Model N) in order to allow the number of batches formed in the optimal solution to serve as lower bound batch numbers.

Model N:

\[
\text{Minimise } \sum_{b=1}^{B} \sum_{k=1}^{K} z_{bk},
\]

subject to constraints (2)–(8), and (15),

\[
C_{max}' \geq \max\{p_j\}.
\]

**Step 2:** Denote the number of lower-bound batches to be the candidate number and denote it as $BN$. Apply $BN$ as one parameter to Model P developed in Section 3. Denote the solution obtained within the limited computational time as $Z^*$.

**Phase II: The solution of subsequent batch number is checked**

**Step 1:** $BN = BN + 1$. 

![Figure 4. An optimal solution for the seven-job example with two parallel batch processing machines.](image-url)
Step 2: If $BN \leq N$, then the batch number $BN$ is applied as one parameter to Model P. Let the solution obtained from Model P be denoted as $Z$ and go to Step 3. If the $BN > N$, then let $Z^*$ be the final solution and stop the algorithm.

Step 3: If $Z \geq Z^*$, then let $Z^*$ be the final solution and stop the algorithm. If $Z < Z^*$, then set $Z$ as the new $Z^*$ and go back to Step 1.

It should be noted that the computational complexity of the original MILP model (Model P) developed in Section 3 is reduced when the number of lower-bound batches is obtained by using the algorithm of Phase I of the CMA. The greater the difference between the number of jobs and the number of batches, the greater is the reduction in complexity. For the example with seven jobs described in Section 3, the number of batches is four, and it is obtained by using the algorithm in Phase I of the CMA. The composite jobs for these batches are presented in Table 3.

Using four batches as the parameter for Model P, 97 variables and 316 constraint equations are obtained. The solution is a makespan of 430 and is obtained within 1.3 CPU seconds by using CPLEX OPL 3.5. Moreover, when the batch number is set to five using Step 1 of Phase II of the CMA, the makespan is also 430 but the computational time is 10 CPU seconds. Therefore, the final solution of the example with seven jobs using the CMA is 430.
5. Heuristic algorithms for large-scale problems

For small- and moderate-size parallel batch processing machine scheduling problems with unequal ready times and non-identical job sizes, the mixed integer linear programming model can provide optimal solutions within reasonable amounts of computational time. However, for large-size parallel batch processing machine scheduling problems, solving the mathematical model is computationally inefficient. Therefore, three heuristic algorithms are proposed to generate efficient solutions for large problems. The proposed algorithms incorporate the merits of the DELAY heuristic solution procedure proposed by Lee and Uzsoy (1999) with some modifications in order to accommodate the parallel batch processing machine environment. The DELAY heuristic algorithm allows for postponement in processing a batch in order to accommodate a job that is due to arrive soon and which might be combined with the delayed batch. This strategy can be used to avoid unacceptable delays in the completion of scheduled batches. At other times, however, if the expected job is due to arrive much later, it is not advantageous to delay processing as this would cause excessive delays to jobs already waiting to be processed.

The three new heuristic algorithms include the characteristics of unequal ready times, non-identical job sizes, and parallel batch processing machines; they are referred to as Heuristic Algorithm 1 (H1), Heuristic Algorithm 2 (H2), and Mixed-strategy Heuristic Algorithm (MixedH). The first two heuristic algorithms essentially consist of two phases. Phase I of each modifies the DELAY algorithm (proposed by Lee and Uzsoy 1999) to form appropriate batches by adding a machine capacity checking step in order to accommodate the constraint of non-identical job sizes to enable the processing of batches of varied numbers of jobs. Furthermore, in Phase II of the two algorithms, the original single-machine scheduling idea proposed by Lee and Uzsoy (1999) is also extended to accommodate a parallel batch processing machine environment. In this phase, the formed batches are then assigned to parallel batch processing machines and sequenced according to the batch ready times and batch processing times with a minimal makespan criterion. Finally, a mixed-strategy approach is also proposed. Algorithm details are presented as follows.

5.1 Heuristic algorithm 1 (H1)

Phase I: Batch formation (the modified DELAY algorithm; Lee and Uzsoy 1999)

Step 0: Let the available set be the set of jobs which are available to be selected as a batch. Sort all jobs associated with ready times in ascending order of magnitude as an unscheduled-job list. Index the ready times in the list as \( r_i \). Assign the first job on the unscheduled-job list into the available set. Set the decision time point \( t \) as the ready time of the first job \( r_1 \) in the available set and set \( i = 1 \) and \( b = 1 \).

<table>
<thead>
<tr>
<th>Batch name</th>
<th>Job ID</th>
<th>Batch size</th>
</tr>
</thead>
<tbody>
<tr>
<td>( b_2 )</td>
<td>1, 3, 7</td>
<td>440</td>
</tr>
<tr>
<td>( b_3 )</td>
<td>5</td>
<td>400</td>
</tr>
<tr>
<td>( b_6 )</td>
<td>6</td>
<td>300</td>
</tr>
<tr>
<td>( b_7 )</td>
<td>2, 4</td>
<td>380</td>
</tr>
</tbody>
</table>

Table 3. The composite jobs for each estimated batch.
Step 1: Arrange the jobs associated with the processing times in the available set in descending order of magnitude. Choose the required number of early jobs in the available set which satisfy the constraint of machine capacity and place them in the candidate batch \((b)\). The longest processing time of all the jobs in batch \(b\) serves as the corresponding batch processing time and is denoted as \(pt_b\).

Step 2: Check whether there is a job \((j)\) on the unscheduled-job list which satisfies constraints \(r_j \leq t + \alpha pt_b\) and \(p_j \geq \alpha pt_b\), where \(0 \leq \alpha \leq 1\). If a job \((j)\) satisfies the conditions, then put job \((j)\) into the candidate batch \((b)\) and go to Step 3. Otherwise go to Step 4.

Step 3: If \(\sum_{q \in b} p_q \leq \beta \eta pt_b\), or \(\sum_{q \in b} s_q > S'\) where \(0 \leq \beta \leq 3\) and \(\eta = \left[\sum_{j=1}^{N} s_j / S'\right]\), then they do not form a part of the candidate batch \((b)\); therefore, let \(i = i + 1\), decision time point \((t)\) be \(r_p\), and then go to Step 5. Otherwise, go to Step 4.

Step 4: Form candidate batch \((b)\), let \(t = t + pt_b\), then remove those jobs which are formed in batch \(b\) from the unscheduled-job list, set \(i = 1\), \(b = b + 1\), and go to Step 6.

Step 5: Select the maximum time value between the current decision time point and the smallest ready time in the unscheduled-job list as the new decision time point. Choose the jobs which have ready times which are earlier than the new point in decision time \((t)\) to update the available set. Go back to Step 1 to reform the candidate batch.

Step 6: Repeat Step 5 until no more candidate jobs can be found on the unscheduled-job list.

Phase II: Batch scheduling

Step 1: Calculate the batch ready time and batch process time for each batch.

Step 2: Sort the batch ready times in ascending order of magnitude.

Step 3: If the ready times of some of the batches are equal, then sort the batch processing times in descending order of magnitude.

Step 4: Schedule the first batch into the first available machine and then remove the batch from the batch list.

Step 5: Repeat Step 4 of Phase II until all batches are scheduled.

In Step 2 of Phase I, Lee and Uzsoy (1999) used parameter \(\alpha\) to accommodate the postponement idea of the DELAY heuristic algorithm. Parameter \(\alpha\) can be used to examine whether there exists a job with a ready time which is less than or equal to the summation of the current decision time point and the \(\alpha pt_b\) time units. In addition, its corresponding job processing time is greater than or equal to the \(\alpha pt_b\). If such a job exists, the job might be combined with the delayed batch to avoid delaying that job.

5.2 Heuristic algorithm 2 (H2)

In Phase II of Heuristic Algorithm 1 (H1), the batches are mainly scheduled based on their batch ready times. If the number of batches to be processed is slightly greater than the number of machines available to process them and, furthermore, if the processing times of some batches are relatively short, then the result may be that H1 performs poorly because the individual short-process-time batches have been assigned to different machines.
resulting in a longer makespan. In attempting to improve the solution, an alternative approach which combines the ready and processing times is proposed. The algorithm is described in the following.

**Phase I: Batch formation**
Use the same process as for Phase I of Heuristic Algorithm 1 (H1).

**Phase II: Batch scheduling**

**Step 1:** Calculate the batch ready times \( r'_b \) and the batch process times \( pt_b \), which are determined by the latest ready time and the longest processing time of all the jobs in one batch, respectively.

**Step 2:** Calculate \( T_b = r'_b + pt_b \).

**Step 3:** Assign batches on \( K \) parallel batch processing machines using the longest processing time (LPT) rule (here it is assumed that the value of \( T_b \) represents the processing time for the LPT rule).

**Step 4:** Sequence batches on each machine in ascending order based on batch ready times.

### 5.3 Mixed-strategy heuristic algorithm (MixedH)

In attempting to obtain better solutions, a mixed-strategy approach is presented. The mixed-strategy approach first uses H1 and H2 algorithms to generate two complete parallel batch processing machine scheduling problem solutions, then selects the best one. This approach is referred to as the Mixed-strategy Heuristic Algorithm (MixedH). MixedH also involves two phases. Phase I, as in Phase I of H1, determines the batch formations. However, Phase II performs the solution procedures of Phase II of H1 and H2 simultaneously to determine the final sequence and to select the value with the minimum makespan from the two solutions available.

### 6. Computational results and comparisons

For the purposes of testing the proposed algorithms and comparing them with the MILP model, an experiment involving two computational tests was designed to generate a series of problem instances. One computational test performed with small to moderate size problems obtains the solution quality of the proposed heuristic algorithms by comparing their solutions with the optimal solutions generated by the MILP model (Model P). The other computational test involves the large size problem taken from a module assembly process at a TFT-LCD factory in Hsinchu Science-based Industrial Park, Taiwan. In the following, two computational results are provided.

#### 6.1 Analysis of results from small and moderate size problems

The experimental design involves two essential characteristics, ready time variation and processing time variation. These two variations are characterised by two magnitudes, large (L) and small (S). Accordingly, the ready times in a problem are generated from uniform distributions in \([0, 300]\), \([0, 100]\) for large and small variations, respectively. The processing times are generated from uniform distributions in \([90, 300]\), \([100, 200]\) for large and small
variations, respectively. The structure and data of the test problems are generated covering a wide variety of scheduling problems encountered in industrial practice. A number of machines and a number of jobs are alternated in order to get 16 problem configurations. For each problem configuration, five problem instances are randomly generated. Thus, 80 different problem instances are generated which are either small or moderate in size. The four different experimental factors are listed in Table 4. The aging oven can process a batch in which the total number of pieces from all the jobs in that batch does not exceed 450 pieces. Without loss of generality, we assume that the size of each job is less than the machine capacity (i.e. 450 pieces of panel). Once the batch processing begins, it is non-preemptive until the batch is completely processed. Processing and ready times are measured in minutes. All jobs should be formed as batches and be processed completely by the minimum makespan.

Table 5 presents the solutions generated by all the proposed algorithms on the eight small problem configurations with seven jobs in each. The values of the optimal solutions are obtained by solving the MILP model (Model P), which is formulated in Section 3. In Table 5, the problem configuration ‘7LL2’ represents the seven jobs with large ready time and large processing time variations, which are processed on two batch machines.

In this testing, the run times of Model P and the CMA may vary for problem instances with different configurations. However, the run times of the CMA are significantly faster than for the original MILP model (Model P). All the solutions of the CMA are equal to the values obtained from Model P; hence the solutions are optimal. In the three heuristic algorithms, their performances are sensitive to the values of the parameters $\alpha$ and $\beta$ (as also concluded by Lee and Uzsoy 1999). This paper provides the experiments involving the three heuristic algorithms to the testing problem instances using several values of $\alpha$, which are initially set to 0, 0.2, 0.4, 0.6, 0.8, and 1 ($0 \leq \alpha \leq 1$) and $\beta$, which are initially set to 0, 0.2, 0.4, … , 2.6, 2.8, and 3 ($0 \leq \beta \leq 1$). The best solution, obtained using one of the parameter combinations, is selected as the final solution to the heuristic algorithm. It is worthwhile to note that the MixedH obtains 34 (out of 40) optimal solutions within 0.8 CPU seconds for each problem instance.

Table 6 displays the results for the problem instances with 15 jobs and different configurations and their performance comparisons in terms of the makespan obtained using mathematical and heuristic algorithmic solutions. In the MILP model (Model P) and Model N of the CMA, the depth-first search strategy (Wolsey 1998) is implemented by choosing the most recently created node. To avoid the CPLEX routine which requires
a tremendous amount of computation time, the maximum run time is set at 28,800 CPU seconds. Furthermore, the nodes created cannot be greater than 1E06 in Phase II of the CMA in order to check the subsequent batch numbers repeatedly. CPLEX could stop at the pre-determined time without guaranteeing optimality for problems with high

Table 5. Run times and makespan results for 7-job problem instances.

<table>
<thead>
<tr>
<th>Problem configuration</th>
<th>MILP</th>
<th>CMA</th>
<th>H1</th>
<th>H2</th>
<th>MixedH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C_{\max}$</td>
<td>Time (sec)</td>
<td>$C_{\max}$</td>
<td>Time (sec)</td>
<td>$C_{\max}$</td>
</tr>
<tr>
<td>1 7LL2</td>
<td>459</td>
<td>2496.0</td>
<td>459</td>
<td>11.2</td>
<td>459</td>
</tr>
<tr>
<td>7LL3</td>
<td>379</td>
<td>8742.0</td>
<td>379</td>
<td>71.9</td>
<td>379</td>
</tr>
<tr>
<td>7LS2</td>
<td>395</td>
<td>501.2</td>
<td>395</td>
<td>13.7</td>
<td>395</td>
</tr>
<tr>
<td>7LS3</td>
<td>345</td>
<td>1032.0</td>
<td>345</td>
<td>21.4</td>
<td>345</td>
</tr>
<tr>
<td>7SL2</td>
<td>430</td>
<td>1041.3</td>
<td>430</td>
<td>11.2</td>
<td>430</td>
</tr>
<tr>
<td>7SL3</td>
<td>370</td>
<td>648.0</td>
<td>370</td>
<td>1.7</td>
<td>370</td>
</tr>
<tr>
<td>7SS2</td>
<td>346</td>
<td>1268.0</td>
<td>346</td>
<td>12.1</td>
<td>346</td>
</tr>
<tr>
<td>7SS3</td>
<td>289</td>
<td>15,475.0</td>
<td>289</td>
<td>66.3</td>
<td>289</td>
</tr>
<tr>
<td>2 7LL2</td>
<td>607</td>
<td>463.0</td>
<td>607</td>
<td>5.5</td>
<td>607</td>
</tr>
<tr>
<td>7LS2</td>
<td>540</td>
<td>15.8</td>
<td>540</td>
<td>6.2</td>
<td>508</td>
</tr>
<tr>
<td>7LS3</td>
<td>488</td>
<td>1268.0</td>
<td>488</td>
<td>6.4</td>
<td>488</td>
</tr>
<tr>
<td>7SL2</td>
<td>518</td>
<td>2899.8</td>
<td>518</td>
<td>32.6</td>
<td>518</td>
</tr>
<tr>
<td>7SL3</td>
<td>418</td>
<td>648.0</td>
<td>418</td>
<td>1.7</td>
<td>418</td>
</tr>
<tr>
<td>7SS2</td>
<td>348</td>
<td>1376.0</td>
<td>348</td>
<td>25.2</td>
<td>348</td>
</tr>
<tr>
<td>7SS3</td>
<td>267</td>
<td>20,714.0</td>
<td>267</td>
<td>66.3</td>
<td>267</td>
</tr>
<tr>
<td>3 7LL2</td>
<td>522</td>
<td>1.6</td>
<td>522</td>
<td>1.1</td>
<td>522</td>
</tr>
<tr>
<td>7LS2</td>
<td>522</td>
<td>7.5</td>
<td>522</td>
<td>5.3</td>
<td>522</td>
</tr>
<tr>
<td>7LS3</td>
<td>455</td>
<td>1.0</td>
<td>455</td>
<td>1.1</td>
<td>455</td>
</tr>
<tr>
<td>7SL2</td>
<td>455</td>
<td>10.0</td>
<td>455</td>
<td>3.0</td>
<td>455</td>
</tr>
<tr>
<td>7SL3</td>
<td>424</td>
<td>693.3</td>
<td>424</td>
<td>8.7</td>
<td>424</td>
</tr>
<tr>
<td>7SS2</td>
<td>330</td>
<td>461.8</td>
<td>330</td>
<td>11.0</td>
<td>345</td>
</tr>
<tr>
<td>7SS3</td>
<td>303</td>
<td>23,341.0</td>
<td>303</td>
<td>85.0</td>
<td>303</td>
</tr>
<tr>
<td>4 7LL2</td>
<td>630</td>
<td>1606.0</td>
<td>630</td>
<td>13.1</td>
<td>630</td>
</tr>
<tr>
<td>7LS2</td>
<td>543</td>
<td>41721.0</td>
<td>543</td>
<td>64.8</td>
<td>543</td>
</tr>
<tr>
<td>7LS3</td>
<td>411</td>
<td>7.7</td>
<td>411</td>
<td>2.9</td>
<td>411</td>
</tr>
<tr>
<td>7SL2</td>
<td>411</td>
<td>12.4</td>
<td>411</td>
<td>6.8</td>
<td>411</td>
</tr>
<tr>
<td>7SL3</td>
<td>512</td>
<td>4290.9</td>
<td>512</td>
<td>7.7</td>
<td>512</td>
</tr>
<tr>
<td>7SL2</td>
<td>425</td>
<td>31,364.0</td>
<td>425</td>
<td>117.8</td>
<td>465</td>
</tr>
<tr>
<td>7SL3</td>
<td>313</td>
<td>462.8</td>
<td>313</td>
<td>9.6</td>
<td>317</td>
</tr>
<tr>
<td>7SS2</td>
<td>258</td>
<td>22,398.0</td>
<td>258</td>
<td>61.6</td>
<td>258</td>
</tr>
<tr>
<td>7SS3</td>
<td>258</td>
<td>22,398.0</td>
<td>258</td>
<td>61.6</td>
<td>258</td>
</tr>
<tr>
<td>5 7LL2</td>
<td>574</td>
<td>1306.8</td>
<td>574</td>
<td>6.4</td>
<td>592</td>
</tr>
<tr>
<td>7LS2</td>
<td>555</td>
<td>7.6</td>
<td>555</td>
<td>5.3</td>
<td>555</td>
</tr>
<tr>
<td>7LS3</td>
<td>453</td>
<td>10.9</td>
<td>453</td>
<td>4.8</td>
<td>453</td>
</tr>
<tr>
<td>7SS2</td>
<td>371</td>
<td>2348.0</td>
<td>371</td>
<td>23.1</td>
<td>393</td>
</tr>
<tr>
<td>7SS3</td>
<td>371</td>
<td>2348.0</td>
<td>371</td>
<td>23.1</td>
<td>393</td>
</tr>
</tbody>
</table>

Note: The underlined values represent the best solutions for each problem instance from among all of the algorithms.
computational complexity. However, the depth-first search strategy can incorporate the strong branching rule (Wolsey 1998) causing the variable selection based on partially solving a number of sub-problems with tentative branches in order to find the most promising branch. Table 6 shows that the performances of the CMA are reasonably good;

<table>
<thead>
<tr>
<th>Problem configuration</th>
<th>MILP</th>
<th>CMA</th>
<th>H1</th>
<th>H2</th>
<th>MixedH</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$C_{max}$</td>
<td>Time (sec)</td>
<td>$C_{max}$</td>
<td>Time (sec)</td>
<td>$C_{max}$</td>
</tr>
<tr>
<td>1 LL2</td>
<td>592</td>
<td>28,800.0</td>
<td>579</td>
<td>786.8</td>
<td>610</td>
</tr>
<tr>
<td>1 LL3</td>
<td>552</td>
<td>28,800.0</td>
<td>552</td>
<td>15.3</td>
<td>552</td>
</tr>
<tr>
<td>1 LS2</td>
<td>467</td>
<td>250.6</td>
<td>467</td>
<td>5.1</td>
<td>467</td>
</tr>
<tr>
<td>1 LS3</td>
<td>467</td>
<td>7471.3</td>
<td>467</td>
<td>19.8</td>
<td>467</td>
</tr>
<tr>
<td>1 LL2</td>
<td>532</td>
<td>28,800.0</td>
<td>532</td>
<td>1096.2</td>
<td>532</td>
</tr>
<tr>
<td>1 LS3</td>
<td>382</td>
<td>382</td>
<td>2789.5</td>
<td>382</td>
<td>0.875</td>
</tr>
<tr>
<td>1 SL2</td>
<td>372</td>
<td>28,800.0</td>
<td>372</td>
<td>1252.5</td>
<td>409</td>
</tr>
<tr>
<td>1 SS2</td>
<td>376</td>
<td>28,800.0</td>
<td>375</td>
<td>2225.0</td>
<td>375</td>
</tr>
</tbody>
</table>

Table 6. Run times and makespan results for 15-job problem instances.

Note: The underlined values represent the best solutions for each problem instance from among all of the algorithms.
that is, with all the problem instances it achieved better solutions than the original MILP model (Model P). Furthermore, the three heuristic algorithms perform well and efficiently. As seen in Table 6, due to its solution strategy, the MixedH provides the best solutions of all three heuristic algorithms for all problem configurations.

Table 7 displays the performance comparisons among the five algorithms in terms of (1) average rank, (2) average run times, (3) number of problem instances receiving the best solutions, and (4) number of optimal solutions. The results indicate that the CMA can obtain 40 (out of 40) optimal solutions for the seven-job test problem instances and it can perform remarkably well for the 15-job problem instances. The CMA significantly speeds up the original MILP model in test problem instances. It should also be noted that all three heuristic algorithms run very fast. With the 80 problem instances tested, it was found that two of them required more than 1 CPU seconds on a Pentium IV 3.2 GHz PC. To access the accuracy of the heuristic solutions, the best solution selected from the CMA and the original MILP model (Model P) is used for each problem instance as a convenient reference point. The average percentage deviations between the MixedH and the selected best solutions are 0.36% and 1.8% for seven-job and 15-job problem instances, respectively. The percentage deviation is defined as \( \frac{V_{\text{MixedH}} - V}{V} \), where \( V_{\text{MixedH}} \) and \( V \) are the values for each problem instance which is obtained using the MixedH and the two mathematically based algorithms, respectively. The CMA may perform inefficiently as the number of jobs increases to the large scale usual in real-world factories. Thus, if the computational time is a primary concern, MixedH can solve real-world problems well.

### 6.2 Analysis of the result based on the large scaled problem

To demonstrate the applicability of the MixedH heuristic algorithm in real world problems, the following problem taken from a module assembly process of a TFT-LCD factory in Hsinchu Science-based Industrial Park, Taiwan, is considered. For the parallel batch processing machine scheduling problem in the aging test operation, there are 100 jobs (which can be categorised into 35 product families) to be processed on six identical aging test machines arranged in parallel. The maximum batch size in one machine is 450 pieces of panel. It is assumed that the job testing recipes require the same testing temperature, which is normally set to a high temperature (55°C).
In order to solve the real-world aging test problem with unequal ready times and non-identical job sizes using the MixedH algorithm, the program codes of the three heuristic algorithms are written in Visual Basic 6.0. As a result, the MixedH algorithm runs quite efficiently. In fact, the MixedH algorithm takes 6.8 CPU seconds to obtain makespan 1330 on six aging test machines.

7. Conclusions
This paper tackled the parallel batch processing machine scheduling problem with unequal ready times, non-identical job sizes, limited machine capacity, and batch dependent processing times. This problem is a variation of the classical parallel batch processing machine scheduling problem, which has many real-world applications. To the best knowledge of the authors, the parallel batch processing machine scheduling problem has not been considered before. In this paper, the scheduling problem is formulated as a mixed integer linear programming model considering batch formation and batch scheduling simultaneously. To reduce the complexity associated with a search of all probable numbers of batches, an effective compound MILP-based algorithm (CMA) is proposed to pre-determine the number of batches and then to apply the number as a parameter of the MILP model. From the computational tests conducted in the paper, the CMA significantly outperforms the original MILP model (Model P) within the limited computational time. Furthermore, if the computational time is a primary concern, three efficient heuristic algorithms are also developed for solving large-scale problems. The performances of the three heuristic algorithms are quite satisfactory. In particular, the MixedH shows its superiority with respect to run time and solution quality, which are essential for real-world factories to schedule their batch processing machines. A real-world problem taken from a module assembly shop floor at a TFT-LCD factory is solved by using MixedH to obtain the near optimal solution within a few CPU seconds. From this investigation, two possible concerns might be useful in further research. The first involves solving a batch processing machine scheduling problem which involves continuous multiple-batch operations. The second is the need to minimise the maximum completion time involved in the consideration of setup times for the parallel batch processing machine scheduling problem with unequal ready times and incompatible product families.

References


