International Journal of Production Research

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tprs20

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

S.H. Chung [a] , Y.T. Tai [a] & W.L. Pearn [a]

[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

International Journal of Production Research
Vol. 47, No. 18, 15 September 2009, 5109–5128

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

*Corresponding author. Email: shchung@mail.nctu.edu.tw

ISSN 0020–7543 print/ISSN 1366–588X online
© 2009 Taylor & Francis
DOI: 10.1080/00207540802010807
http://www.informaworld.com

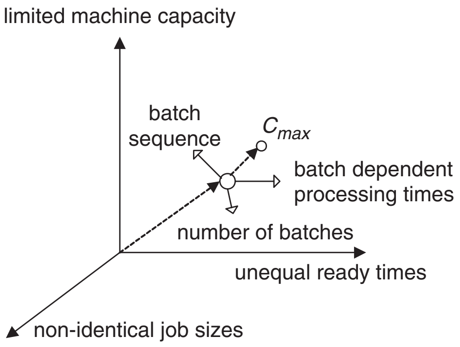

Figure 1. The multiple dimensions of the parallel batch processing machine scheduling problem.

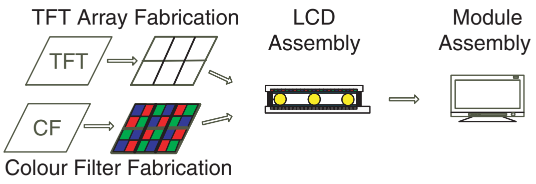

Figure 2. The TFT-LCD process flow.

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).

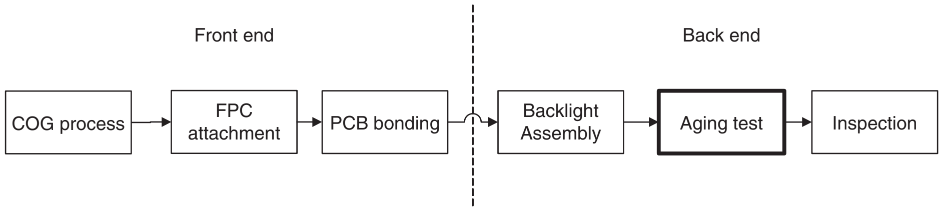

Figure 3. The six steps in the module assembly process.

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 in the aging test operation 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/batch, r_j, compatible/$C_{\max}$, 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 ($T_{\max}$), the number of tardy jobs (ΣU_i), and maximum lateness time ($L_{\max}$) on a single batch processing machine. They have also presented heuristics for the parallel batch processing machine scheduling problem

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

References	Shop type	Ready time	Job size	Performance criterion
Lee *et al*. (1992)	Single machine/parallel machines	Unequal/equal	Identical/identical	$T_{\max}$, ΣU_i, $L_{\max}/C_{\max}$, $L_{\max}$
Uzsoy (1994)	Single machine	Equal	Non-identical	$C_{\max}$, ΣC_i
Erramilli and Mason (2006)	Single machine	Equal	Non-identical	$\Sigma w_i T_i$
Damodaran and Srihari (2004)	Two machines in a flow shop	Equal	Non-identical	$C_{\max}$
Kashan *et al*. (2006)	Single machine	Equal	Non-identical	$C_{\max}$
Lee and Uzsoy (1999)	Single machine	Unequal	Identical	$C_{\max}$
Sung and Choung (2000)	Single machine	Unequal/equal	Identical	$C_{\max}$
Sung *et al*. (2002)	Single machine	Unequal	Identical	$C_{\max}$
Van Der Zee (2004)	Single machine	Dynamic arrival	Identical	$\bar{F}_i$
Chang *et al*. (2004)	Parallel machines	Equal	Non-identical	$C_{\max}$
This paper	Parallel machines	Unequal	Non-identical	$C_{\max}$

with the minimum makespan ($C_{\max}$) and maximum lateness time ($L_{\max}$) 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 (ΣC_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 ($\Sigma w_i T_i$) 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 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 $M = \{m_k \mid k = 1, 2, \ldots, K\}$, 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 bth 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	job index, $j = 1, 2, \ldots, N$,
b	batch index, $b = 1, 2, \ldots, B$,
k	machine index, $k = 1, 2, \ldots, K$.

Decision variables

$$x_{jbk} \quad \begin{cases} 1 & \text{if job } j \text{ is assigned to batch } b \text{ on machine } m_k, \\ 0 & \text{otherwise;} \end{cases}$$

$$y_{bb'k} \quad \begin{cases} 1 & \text{if batch } b' \text{ is scheduled following batch } b \text{ on machine } m_k, \\ 0 & \text{otherwise;} \end{cases}$$

$$z_{bk} \quad \begin{cases} 1 & \text{if batch } b \text{ is assigned to machine } m_k, \\ 0 & \text{otherwise;} \end{cases}$$

t_{bk}	the starting time of batch b to be processed on machine m_k,
$C_{\max}$	the maximum completion time (makespan).

Model P

$$\text{Minimise } C_{\max} + \mu \sum_{b=1}^{B} \sum_{k=1}^{K} z_{bk} \tag{1}$$

subject to

$$\sum_{b=1}^{B} \sum_{k=1}^{K} x_{jbk} = 1, \quad \text{for all } j, \tag{2}$$

$$\sum_{k=1}^{K} z_{bk} \le 1, \quad \text{for all } b, \tag{3}$$

$$\sum_{j=1}^{N} x_{jbk} \le Q_1 z_{bk}, \quad \text{for all } b, k, \tag{4}$$

$$\sum_{j=1}^{N} \sum_{k=1}^{K} s_j x_{jbk} \le S', \quad \text{for all } b, \tag{5}$$

$$pt_b \ge p_j \times x_{jbk}, \quad \text{for all } j, b, k, \tag{6}$$

$$C_{\max} \ge t_{bk} + pt_b, \quad \text{for all } b, k, \tag{7}$$

$$t_{bk} \ge r_j x_{jbk}, \quad \text{for all } j, b, k, \tag{8}$$

$$t_{bk} + pt_b - t_{b'k} + Q_2(y_{bb'k} - 1) \le 0, \quad \text{for all } b, k, b' \ne b, \tag{9}$$

$$(y_{bb'k} + y_{b'bk}) - Q_2(z_{bk} + z_{b'k} - 2) \ge 1, \quad \text{for all } b, k, b' \ne b, \tag{10}$$

$$(y_{bb'k} + y_{b'bk}) + Q_2(z_{bk} + z_{b'k} - 2) \le 1, \quad \text{for all } b, k, b' \ne b, \tag{11}$$

$$(y_{bb'k} + y_{b'bk}) - Q_2(z_{bk} + z_{b'k}) \leq 0, \quad \text{for all } b, k, b' \neq b, \tag{12}$$

$$(y_{bb'k} + y_{b'bk}) - Q_2(z_{b'k} - z_{bk} + 1) \leq 0, \quad \text{for all } b, k, b' \neq b, \tag{13}$$

$$(y_{bb'k} + y_{b'bk}) - Q_2(z_{bk} - z_{b'k} + 1) \leq 0, \quad \text{for all } b, k, b' \neq b, \tag{14}$$

$$x_{jbk} \in \{0,1\}, \quad \text{for all } j, b, k, \tag{15}$$

$$y_{bb'k} \in \{0,1\}, \quad \text{for all } b, k, b' \neq b, \tag{16}$$

$$z_{bk} \in \{0,1\}, \quad \text{for all } b, k, \tag{17}$$

$$C_{\max} \geq 0. \tag{18}$$

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 μ 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 $t_{bk} + pt_b \leq t_{b'k}$, 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 $y_{b'bk}$ should be set to zero ($y_{bb'k} + y_{b'bk} \leq 0$) if any two batches b and $b^{'}$ are not scheduled on the machine m_k($z_{bk} + z_{b'k} = 0$). 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

Table 2. Job sizes, ready times, and processing times of the seven independent jobs.

Job ID	Size (pieces)	Ready time	Processing time
1	50	6	160
2	200	40	120
3	240	8	90
4	180	10	190
5	400	80	290
6	300	30	160
7	150	80	200

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.

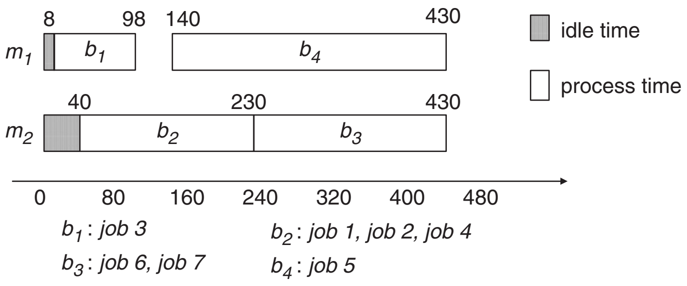

Figure 4. An optimal solution for the seven-job example with two parallel batch processing machines.

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 ($y_{bb'k}$) 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:

$$\text{Minimise} \sum_{b=1}^{B}\sum_{k=1}^{K} z_{bk} \tag{19}$$

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

$$C'_{\max} \geq \max\{p_j\}. \tag{20}$$

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: $BN = BN + 1$.

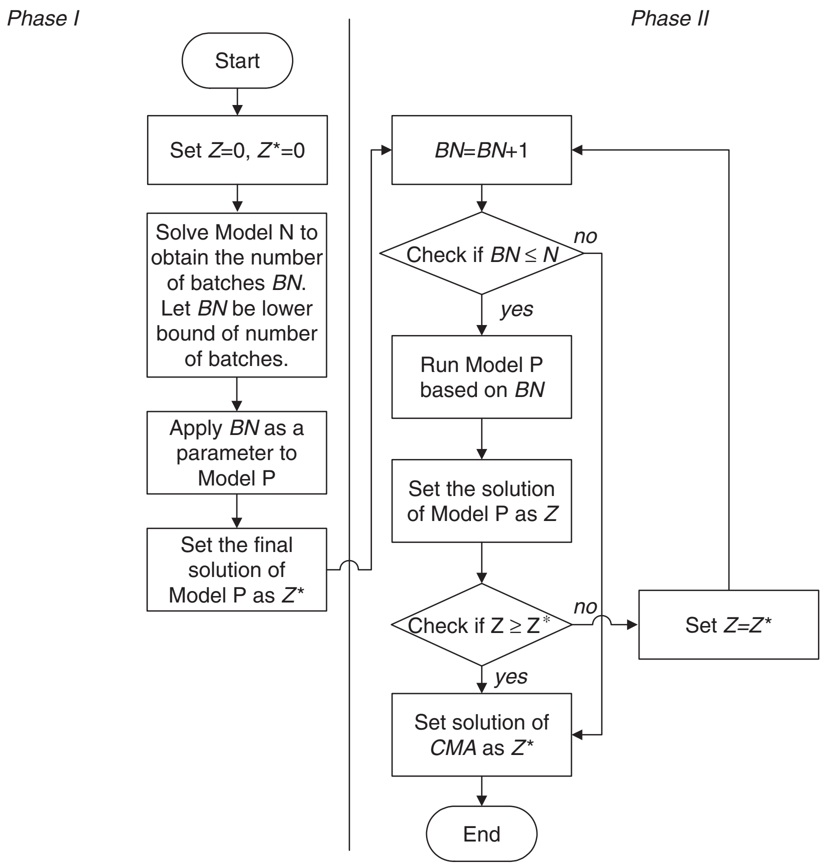
Phase I
Phase II
Start
Set Z=0, Z*=0
Solve Model N to obtain the number of batches BN. Let BN be lower bound of number of batches.
Apply BN as a parameter to Model P
Set the final solution of Model P as Z*
BN=BN+1
Check if BN ≤ N
no
yes
Run Model P based on BN
Set the solution of Model P as Z
Check if Z ≥ Z*
no
yes
Set Z=Z*
Set solution of CMA as Z*
End

Figure 5. The flow chart of the compound MILP-based algorithm (CMA).

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.

Table 3. The composite jobs for each estimated batch.

Batch name	Job ID	Batch size
b_2	1, 3, 7	440
b_3	5	400
b_6	6	300
b_7	2, 4	380

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 $b = 1$.

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 $r_j \le t + \alpha pt_b$ and $p_j \ge \alpha pt_b$, where $0 \le \alpha \le 1$. 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 $\Sigma_{q \in b} p_q \le \beta \eta pt_b$, or $\Sigma_{q \in b} s_q > S'$ where $0 \le \beta \le 3$ and $\eta = \left\lceil \sum_{j=1}^{N} s_j / S' \right\rceil$, then they do not form a part of the candidate batch (b); therefore, let $i = i + 1$, decision time point (t) be r_i, 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 α to accommodate the postponement idea of the DELAY heuristic algorithm. Parameter α 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 αpt_b time units. In addition, its corresponding job processing time is greater than or equal to the αpt_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

Table 4. Experimental factors for small- and moderate-sized problems.

Factor	Value considered	Number of values
Number of jobs (N)	7, 15	2
Ready time variation	L, S	2
Processing time variation	L, S	2
Number of machines (K)	2, 3	2
Total problem configurations		16
Instances per configuration		5
Total problem instances		80

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 α and β (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 α, which are initially set to $0, 0.2, 0.4, 0.6, 0.8$, and 1 ($0 \leq \alpha \leq 1$) and β, which are initially set to $0, 0.2, 0.4, \ldots, 2.6, 2.8$, 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

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

		MILP		*CMA*		*H1*		*H2*		*MixedH*	
	Problem configuration	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)
1	7LL2	459	2496.0	459	11.2	459	0.672	486	0.641	459	0.688
	7LL3	379	8742.0	379	71.9	379	0.625	379	0.625	379	0.656
	7LS2	395	501.2	395	13.7	395	0.609	401	0.609	395	0.672
	7LS3	345	1032.0	345	21.4	365	0.625	345	0.625	345	0.656
	7SL2	430	1041.3	430	11.2	430	0.656	480	0.625	430	0.676
	7SL3	370	648.0	370	1.7	370	0.625	370	0.688	370	0.719
	7SS2	346	1268.0	346	12.1	346	0.625	393	0.625	346	0.656
	7SS3	289	15,475.0	289	66.3	289	0.609	306	0.625	289	0.672
2	7LL2	607	463.0	607	5.5	607	0.613	631	0.625	607	0.656
	7LL3	540	15.8	540	6.2	568	0.609	540	0.609	540	0.656
	7LS2	488	482.0	488	6.4	488	0.625	502	0.672	488	0.712
	7LS3	418	2899.8	418	32.6	418	0.625	422	0.625	418	0.672
	7SL2	561	450.0	561	18.3	570	0.641	575	0.609	570	0.719
	7SL3	435	28,052.0	435	150.3	435	0.609	435	0.609	435	0.656
	7SS2	348	1376.0	348	25.2	348	0.641	367	0.703	348	0.756
	7SS3	267	20,714.0	267	115.9	267	0.625	267	0.609	267	0.688
3	7LL2	522	1.6	522	1.1	522	0.625	599	0.609	522	0.734
	7LL3	522	7.5	522	5.3	522	0.641	522	0.594	522	0.672
	7LS2	455	1.3	455	1.1	455	0.609	471	0.594	455	0.672
	7LS3	455	10.0	455	3.0	455	0.641	455	0.609	455	0.672
	7SL2	424	693.3	424	8.7	424	0.641	468	0.609	424	0.734
	7SL3	332	7389.2	332	4.4	358	0.625	332	0.672	332	0.692
	7SS2	330	461.8	330	11.0	345	0.625	343	0.594	343	0.656
	7SS3	303	23,341.0	303	85.0	318	0.625	303	0.609	303	0.813
4	7LL2	630	1606.0	630	13.1	656	0.609	656	0.609	656	0.672
	7LL3	543	41721.0	543	64.8	619	0.625	543	0.625	543	0.672
	7LS2	411	7.7	411	2.9	411	0.625	530	0.594	411	0.656
	7LS3	411	12.4	411	6.8	411	0.672	411	0.609	411	0.734
	7SL2	512	4209.9	512	7.7	512	0.625	534	0.672	512	0.756
	7SL3	425	31,346.8	425	117.8	465	0.625	425	0.703	425	0.741
	7SS2	313	462.8	313	9.6	317	0.625	336	0.609	317	0.641
	7SS3	258	22,398.0	258	61.6	261	0.625	258	0.609	258	0.641
5	7LL2	574	1306.8	574	6.4	592	0.641	590	0.641	590	0.656
	7LL3	555	7.6	555	5.3	555	0.625	555	0.609	555	0.655
	7LS2	453	1.6	453	0.7	454	0.625	453	0.703	453	0.741
	7LS3	453	10.9	453	4.8	453	0.609	453	0.719	453	0.741
	7SL2	498	3310.0	498	15.6	498	0.625	534	0.609	498	0.766
	7SL3	398	43,339.0	398	134.2	398	0.719	398	0.609	398	0.752
	7SS2	371	3748.0	371	23.1	393	0.609	373	0.672	373	0.741
	7SS3	294	22,360.0	294	96.1	298	0.625	294	0.609	294	0.719

Note: The underlined values represent the best solutions for each problem instance from among all of the algorithms.

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

		MILP		*CMA*		*H1*		*H2*		*MixedH*	
	Problem configuration	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)	$C_{\max}$	Time (sec)
1	15LL2	592	28,800.0	$\underline{579}$	786.8	610	0.844	612	0.797	610	0.947
	15LL3	$\underline{552}$	28,800.0	$\underline{552}$	15.3	$\underline{552}$	0.781	$\underline{552}$	0.781	$\underline{552}$	0.931
	15LS2	$\underline{467}$	250.6	$\underline{467}$	5.1	$\underline{467}$	0.781	$\underline{467}$	0.766	$\underline{467}$	0.916
	15LS3	$\underline{467}$	7471.3	$\underline{467}$	19.8	$\underline{467}$	0.797	$\underline{467}$	1.031	$\underline{467}$	1.041
	15SL2	$\underline{532}$	28,800.0	$\underline{532}$	1096.2	$\underline{532}$	0.813	575	0.766	$\underline{532}$	0.994
	15SL3	$\underline{382}$	28,800.0	$\underline{382}$	2789.5	$\underline{382}$	0.875	$\underline{382}$	0.875	$\underline{382}$	0.947
	15SS2	$\underline{372}$	28,800.0	$\underline{372}$	1252.5	409	0.828	392	0.766	392	0.931
	15SS3	276	28,800.0	$\underline{275}$	2225.0	303	0.938	275	0.766	275	0.994
2	15LL2	614	28,800.0	$\underline{605}$	3756.1	629	0.813	713	0.766	629	0.978
	15LL3	$\underline{565}$	642.0	$\underline{565}$	17.4	$\underline{565}$	0.797	$\underline{565}$	0.750	$\underline{565}$	0.947
	15LS2	$\underline{498}$	28,800.0	$\underline{498}$	3063.0	516	0.797	551	0.797	516	0.931
	15LS3	$\underline{478}$	28,800.0	$\underline{478}$	31.6	$\underline{478}$	0.813	$\underline{478}$	0.750	$\underline{478}$	0.931
	15SL2	500	28,800.0	$\underline{475}$	1085.9	$\underline{475}$	0.813	537	0.750	$\underline{475}$	0.963
	15SL3	$\underline{380}$	28,800.0	$\underline{380}$	150.7	$\underline{380}$	0.797	$\underline{380}$	0.750	$\underline{380}$	0.916
	15SS2	380	28,800.0	$\underline{374}$	8478.0	380	0.781	403	0.766	380	0.947
	15SS3	$\underline{293}$	28,800.0	$\underline{293}$	183.9	$\underline{293}$	0.797	$\underline{293}$	0.750	$\underline{293}$	0.931
3	15LL2	$\underline{505}$	28,800.0	$\underline{505}$	902.3	$\underline{505}$	0.797	596	0.750	$\underline{505}$	0.963
	15LL3	$\underline{442}$	28,800.0	$\underline{442}$	48.2	$\underline{442}$	0.781	$\underline{442}$	0.750	$\underline{442}$	0.916
	15LS2	$\underline{455}$	28,800.0	$\underline{455}$	872.4	467	0.781	498	0.750	467	0.931
	15LS3	$\underline{424}$	28,800.0	$\underline{424}$	18.8	426	0.875	426	0.813	426	0.963
	15SL2	$\underline{425}$	28,800.0	$\underline{425}$	999.0	428	0.797	466	0.750	428	0.931
	15SL3	323	28,800.0	$\underline{314}$	1089.0	323	0.797	$\underline{314}$	0.766	$\underline{314}$	0.931
	15SS2	$\underline{375}$	28,800.0	$\underline{375}$	1290.0	393	0.781	420	0.750	393	0.916
	15SS3	318	28,800.0	$\underline{306}$	3012.9	315	0.797	$\underline{306}$	0.750	$\underline{306}$	0.947
4	15LL2	$\underline{495}$	28,800.0	$\underline{495}$	1185.0	516	0.797	546	0.750	516	0.931
	15LL3	$\underline{445}$	28,800.0	$\underline{445}$	16.9	$\underline{445}$	0.859	469	0.813	$\underline{445}$	0.978
	15LS2	435	28,800.0	$\underline{422}$	997.7	435	0.781	456	0.828	435	0.947
	15LS3	$\underline{370}$	28,800.0	$\underline{370}$	1275.5	375	0.781	390	0.734	375	0.892
	15SL2	$\underline{459}$	28,800.0	$\underline{459}$	1323.0	473	0.797	473	0.750	473	0.910
	15SL3	$\underline{369}$	28,800.0	$\underline{369}$	219.0	$\underline{369}$	0.781	$\underline{369}$	0.766	$\underline{369}$	0.947
	15SS2	384	28,800.0	$\underline{383}$	1437.0	403	0.766	410	0.734	403	0.916
	15SS3	$\underline{327}$	28,800.0	$\underline{327}$	3684.0	331	0.781	331	0.750	331	0.916
5	15LL2	$\underline{553}$	28,800.0	$\underline{553}$	1014.7	586	0.828	590	0.766	586	0.963
	15LL3	$\underline{538}$	813.0	$\underline{538}$	13.1	$\underline{538}$	0.797	$\underline{538}$	0.781	$\underline{538}$	0.916
	15LS2	$\underline{454}$	28,800.0	$\underline{454}$	472.2	464	0.875	488	0.828	464	0.994
	15LS3	$\underline{451}$	28,800.0	$\underline{451}$	22.7	$\underline{451}$	0.859	$\underline{451}$	0.828	$\underline{451}$	0.993
	15SL2	517	28,800.0	$\underline{477}$	1483.9	505	0.797	517	0.750	505	0.916
	15SL3	363	28,800.0	$\underline{350}$	697.5	363	0.828	363	0.750	363	0.916
	15SS2	374	28,800.0	$\underline{368}$	3138.0	390	0.797	390	0.750	390	0.900
	15SS3	324	28,800.0	$\underline{321}$	4298.0	368	0.813	326	0.750	326	0.947

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 7. Comparisons of the five algorithms.

Problem		*MILP*	*CMA*	*H1*	*H2*	*MixedH*
7-job	Average rank among the five algorithms[a]	1	1	2.23	2.9	1.3
	Average run times (in CPU seconds)	7335.3	31.5	0.629	0.630	0.696
	Number of problem instances receiving the best solutions	40	40	25	19	34
	Number of optimal solutions	40	40	25	19	34
15-job	Average rank among the five algorithms[a]	1.6	1	2.3	3.1	1.95
	Average run times (in CPU seconds)	26149	1362	0.810	0.776	0.943
	Number of problem instances receiving the best solutions	26	40	15	15	19
	Number of optimal solutions	4	4	4	4	4

Note: [a]The smallest rank value indicates the best solutions among all 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 $(V_{MixedH}-V)/V$, where V_{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

Chang, P.Y., Damodaran, P., and Melouk, S., 2004. Minimising makespan on parallel batch processing machines. *International Journal of Production Research*, 42, 4211–4220.

Damodaran, P. and Srihari, K., 2004. Mixed integer formulation to minimise makespan in a flow shop with batch processing machines. *Mathematical and Computer Modelling*, 40, 1465–1472.

Erramilli, V. and Mason, S.J., 2006. Multiple orders per job compatible batch scheduling. *IEEE Transactions on Electronics Packaging Manufacturing*, 29 (4), 285–296.

Ikura, Y. and Gimple, M., 1986. Efficient scheduling algorithms for a single batch processing machine. *Operations Research Letters*, 5, 61–65.

Kashan, A.H., Karimi, B., and Jolai, F., 2006. Effective hybrid genetic algorithm for minimising makespan on a single-batch-processing machine with non-identical job sizes. *International Journal of Production Research*, 44, 2337–2360.

Lee, C.Y. and Uzsoy, R., 1999. Minimising makespan on a single batch processing machine with dynamic job arrivals. *International Journal of Production Research*, 37 (1), 219–236.

Lee, C.Y., Uzsoy, R., and Martin-Vega, L.A., 1992. Efficient algorithms for scheduling semiconductor burn-in operations. *Operations Research*, 40 (4), 764–775.

Mönch, L., *et al*., 2005. Heuristic scheduling of jobs on parallel batch machines with incompatible job families and unequal ready times. *Computer & Operations Research*, 32, 2731–2750.

Morton, T.E. and Pentico, D.W., 1993. *Heuristic scheduling systems: with applications to production systems and project management*. New York: John Wiley & Sons.

Pearn, W.L., Chung, S.H., and Yang, M.H., 2002. Minimising the total machine workload for the wafer probing scheduling problem. *IIE Transactions*, 34, 211–220.

Sung, C.S. and Choung, Y.I., 2000. Minimising makespan on a single burn-in oven in semiconductor manufacturing. *European Journal of Operational Research*, 120, 559–574.

Sung, C.S., *et al*., 2002. Minimising makespan on a single burn-in oven with job families and dynamic job arrivals. *Computer & Operations Research*, 29, 995–1007.

Uzsoy, R., 1994. Scheduling of a single batch processing machines with non-identical job sizes. *International Journal of Production Research*, 32, 1615–1635.

Van Der Zee, D.J., 2004. Dynamic scheduling of batch servers with compatible product families. *International Journal of Production Research*, 42, 4803–4826.

Wolsey, L.A., 1998. *Integer programming*. 1st ed. New York: John Wiley & Sons.