*Chung Yuan Journal,* Vol. 31, No. 3 / 2003, September / pp. 197-209

# Scheduling for a Three-Stage Flowshop with Batch and Discrete Processors

LING-HUEY SU AND CHUN-CHING LIAO

*Department of Industrial Engineering*
*Chung Yuan Christian University*
*Chung-Li, 32023, Taiwan, R.O.C.*

(Received: March 7, 2003; Accepted: July 18, 2003)

**ABSTRACT**

This paper is an extension of a three-stage flowshop with a batch processor in the second stage and discrete processors in the first and the third stage studied by Ahmadi et al. [3]. We consider two variants where the batch processor is located in the first and third stage respectively. The objective is to minimize the makespan. An efficient heuristic and a mathematical programming model for each case are presented. We prove some properties that identify a specific class of optimal schedule, and then use these properties in designing heuristics and the mathematical programming models. Computational experiences with the algorithms are also reported.

**Key words:** *Scheduling, Three-stage flowshop, Batch processor, Heuristic.*

## I. Introduction

There are many practical systems incorporating batch processor and discrete processor, for example, semiconductor manufacturing, computer integrated manufacturing, etc. Sung and Choung [1] stated that batch processors can be divided into two categories based on batch processing time pattern: (a) the processing time is dependent on the jobs assigned in each batch, (b) the processing time of each batch is fixed and independent of the jobs grouped together in the batch. An example of (a) is the burn-in oven process model wherein the processing time of each batch is the maximum processing time among those of jobs in the batch [2]. On the other hand, the wafer fabrication process is an example of category (b) wherein the independent processing time scheduling models are applicable to batch processing machines.

Ahmadi et al. [3] examined a class of problems defined by a two or three machine flowshop with respect to the sum of completion times and the makespan. Regardless of the number of jobs contained in a batch, the processing length of the batch is fixed and independent of the jobs in the batch. They analyzed the complexity of this class of problems, devised polynomial time algorithms for some special cases, and presented heuristic algorithms and their performance analysis. This problem had also been studied by Uzsoy et al. [4] and Webster and Baker [5] respectively. Recently, Cheng and Wang [6] considered a two-machine flowshop scheduling problem to minimize the makespan. Like the machine environment considered by Ahmadi et al., the two machines process the job either individually or in batches. They assumed that the processing time of a batch to be a constant for all jobs. They also show that these problems are NP-complete in the ordinary sense and proposed some polynomially solvable cases. Lin and Cheng [7] considered a scheduling problem where a set of jobs was simultaneously available for processing in a no-wait two-machine flowshop wherein all jobs were processed on both machines in batches. They also showed that several restricted versions of the problem were strongly NP-hard. Wang and Chern [8] considered a two-machine multi-family flowshop scheduling problem with non-identical capacity requirements on two batch processing machines. The objective is to find a sequence of families and sequence of jobs in each family such that the makespan is minimized. For the study of the batch processor in a three-stage flowshop, Ahmadi et al. [3] addressed the case in which the second stage is a batch processor. They showed that this problem was NP-complete in the strong sense and presented a simple heu-

ristic and established an upper bound on the worst case performance ratio of the heuristic. This paper extends Ahmadi et al.'s work to include other two variants wherein the batch processor is located in the first stage (case 1) and third stage (case 2) respectively. According to the notations devised by Ahmadi et al. [3], the considered flowhsop problems could be denoted as $\beta\rightarrow\delta\rightarrow\delta$ for case 1 and $\delta\rightarrow\delta\rightarrow\beta$ for case 2, where $\beta$ and $\delta$ denote a batch processor and a discrete processor, respectively. An efficient heuristic and a mathematical programming model for each case are presented. We developed several solution properties which is the base of our heuristics and the mathematical programming models. Computational experiences with the algorithms are also reported.

## II. Notations and formulation

As mentioned above, we consider a three-stage flowshop with a batch processor in one stage and a discrete processor in each of the other two stages. Two variants to Ahmadi's case are considered separately. The assumptions and notations are given as following:

1. The batch processing time for each job is a constant.
2. The discrete processor processes one job at a time and the processing time is known and deterministic.
3. The capacity of the batch is known and fixed.
4. Jobs are not preemptive.
5. All jobs are available simultaneously at time zero.

To describe the problem, we introduce the following notations:

**Known Variables:**

$N$: the number of jobs.
$U$: the capacity of the batch processor.
$t$: the time needed to process a batch of jobs, $t$ is constant.
$a_k$: the processing time of job $k$ on the discrete processor in stage 1; $k$=1,2,...,$N$.
$b_k$: the processing time of job $k$ on the discrete processor in stage 2; $k$=1,2,...,$N$.
$c_k$: the processing time of job $k$ on the discrete processor in stage 3; $k$=1,2,...,$N$.
$n$: the number of batches, $n=\lceil N/U \rceil$ where $\lceil x \rceil$ denotes the smallest integer not less than $x$.

**Decision Variables:**

$Z_{ijk}$: if job $k$ is scheduled at the $j$th position of the ith batch;
$Z_{ijk}$=1 : job $k$ is scheduled at the $j$th position of the $i$th batch;
$Z_{ijk}$=0 : otherwise.
$i$=1,2,3,...,$n$; $j$=1,2,3,..., $U$; $k$=1,2,3,...,$N$;
$x_{ij}$: the idle time of the $j$th job in the ith batch on the discrete processor in stage 2; $i$=1,2,...,$n$; $j$=1,2,..., $U$.
$y_{ij}$: the idle time of the $j$th job in the ith batch on the discrete processor in stage 3;
$i$=1,2,...,$n$; $j$=1,2,...,$U$.
$C_{max}$: the completion time of all jobs in the shop; i.e. the makespan.

**Auxiliary variables:**

$a_{ij}$: the processing time of the $j$th job in the $i$th batch on the discrete processor in stage 1.
$A_i$: the total processing time of the $i$th batch on the discrete processor in stage 1.
$b_{ij}$: the processing time of the $j$th job in the $i$th batch on the discrete processor in stage 2.
$B_i$: the total processing time of the $i$th batch on the discrete processor in stage 2.
$c_{ij}$: the processing time of the $j$th job in the $i$th batch on the discrete processor in stage 3.
$C_i$: the total processing time of the $i$th batch on the discrete processor in stage 3.
$S_i$: the start time of the ith batch on the batch processor.
$F_i$: the completion time of the $i$th batch on the batch processor.
$O_{ij}$: the start time of the $j$th job in the $i$th batch on the first discrete processor.
$V_{ij}$: the completion time of the $j$th job in the $i$th batch on the first discrete processor.
$R_{ij}$: the start time of the $j$th job in the $i$th batch on the second discrete processor.
$C_{ij}$: the completion time of the $j$th job in the $i$th batch on the second discrete processor.

## III. Case 1 of the $\beta\rightarrow\delta\rightarrow\delta$ system.

**Theorem 1.** *A full batch job schedule minimizes* $C_{\max}$.

**Proof.** The proof can be obtained by shifting-forward some jobs in late batches. We omit the details.

Theorem 1 implies that it is sufficient for the optimal solution to consider only the full batch schedules in stage 1.

**Definition 1.** LOE ($N$, $U$): Batching in which the first $n-1$ batches all are of full capacity and the last batch contains the remaining $[N-(n-1)U]$ jobs.

**Property 1.** *The jobs scheduled in the same batch in*

*stage 1 processed on the discrete processor in stage 2 consecutively (i.e. without inserted idle time) minimize the makespan.*

**Proof.** If the jobs scheduled in the same batch in stage 1 processed on the discrete processor in stage 2 consecutively, that is, the jobs are processed as early as possible and no idle time incurred within the batch, the makespan is therefore minimized.

**Property 2.** *Sequencing the jobs on the discrete processors in stage 2 and 3 using Johnson's algorithm will minimize makespan if the following condition hold,* $\sum_{i=1}^{k-1} B_i \geq (k-1)\times t$, *for* $k=2,3,..n$, *i.e., the discrete processor in stage 2 is a bottleneck processor.*

**Proof.** It follows from the condition $\sum_{i=1}^{k-1} B_i \geq (k-1)\times t$, for $k=2,3,..n$ that no idle time incurred in stage 2 and therefore one need consider only the idle times incurred in stage 3. This leads to the similar situation as typical two-machine static flowshop problem and Johnson's algorithm is thus used to obtain the optimal sequence.

### A. Mixed integer programming

In this section, a mixed integer programming model with $[(N+4)B+2]n$ variables and $(N+4n+6nB+B-2)$ constraints is formulated for benchmarking. The model is formulated as follows.
Objective Function:

Min: $C_{max}$

$$\sum_{j=1}^{n}\sum_{k=1}^{U} Z_{ijk} = 1 \qquad k=1.....N \qquad (1)$$

$$\sum_{j=1}^{U}\sum_{k=1}^{N} Z_{ijk} \leq U \qquad i=1.....n \qquad (2)$$

$$\sum_{k=1}^{N} Z_{ijk} \leq 1 \qquad i=1.....n \quad j=1.....U \qquad (3)$$

$$S_i \geq F_{i-1} \qquad i=2.....n \qquad (4)$$

$$F_i = S_i + t \qquad i=1.....n \qquad (5)$$

$$O_{i1} \leq V_{i-1,U} \qquad i=1.....n \qquad (6)$$

$$O_{i1} \geq V_{i-1,U} \qquad i=2.....n \qquad (7)$$

$$O_{i1} \geq V_{i,j-1} \qquad i=1.....n \quad j=2.....U \qquad (8)$$

$$V_{ij} = O_{ij} + \sum_{k=1}^{N} (b_k \times Z_{ijk}) \qquad i=1.....n \quad j=2.....U \qquad (9)$$

$$R_{ij} \geq V_{ij} \qquad i=1.....n \quad j=1.....U \qquad (10)$$

$$R_{i1} \geq C_{i-1,U} \qquad i=2.....n \qquad (11)$$

$$R_{ij} \geq C_{i,j-1} \qquad i=2.....n \qquad (12)$$

$$C_{ij} = R_{ij} + \sum_{k=1}^{N} (c_k \times Z_{ijk}) \qquad i=1.....n \quad j=1.....U \qquad (13)$$

$$C_{max} \geq C_{n,j} \qquad j=1.....U \qquad (14)$$

Constraint (1) ensures that each job must be scheduled exactly once. Constraint (2) guarantees that the number of jobs scheduled in one batch cannot exceed the capacity of the batch processor. Constraint (3) specifies that at most one job be scheduled at the given position. Constraint (4) ensures that each batch starts after its completion from the previous batch. Constraint (5) defines the completion time of each batch. Constraint (6) ensures that each job in the same batch on the discrete processor in stage 2 starts after its completion from the batch processing in stage 1. Constraint (7) and (8) indicate that each job in stage 2 starts after its completion time from the previous job. Constraint (9) defines the completion time of the jth job in the ith batch in stage 2. Constraint (10) ensures that each job in stage 3 starts after its completion time from the previous stage. Constraint (11) and (12) indicate that each job in stage 3 starts after its completion time from the previous job in the same stage. Constraint (13) defines the completion time of the jth ranked job in the ith batch in stage 3. Constraint (14) defines the maximum completion time.

### B. The heuristic algorithm

It is seen from the theorem and properties described above that in order to obtain the optimal solution, it is necessary to process the jobs by applying full batch policy in stage 1 and allocating the jobs consecutively within each batch to the discrete processor in stage 2.

**Initialization: Obtain an initial schedule**

Let $X_i$ be the idle time preceding batch $i$ in stage 2. The value of the $X_k$ can be given by the following recurrence relationships.

$$X_k = \max\left(nt - \sum_{i=1}^{k-1} X_i - \sum_{i=1}^{k-1} B_i, 0\right) \text{ and thus}$$

$$\sum_{i=1}^{n} X_i = \max\left(nt - \sum_{i=1}^{n-1} B_i, \sum_{i=1}^{n-1} X_i\right)$$

$$= \max\left[nt - \sum_{i=1}^{n-1} B_i, (n-1)t - \sum_{i=1}^{n-2} B_i, ....., 2t - B_1, t\right]$$

In the following discussions, we assume that the assignment of the job within any batch in stage 3 is shifted to the left and denote the idle time before batch $k$ as $Y_k$, as shown in Fig. 1. The following relation is existed.

$$\sum_{i=1}^{n} Y_k = \max\Big[\sum_{i=1}^{k} X_i + \sum_{i=1}^{k-1} B_i + b_{k1} - \Big(\sum_{i=1}^{k-1} C_i\Big),$$
$$\sum_{i=1}^{k-1} X_i + \sum_{i=1}^{k-2} B_i + b_{k-1,1} - \Big(\sum_{i=1}^{k-2} C_i\Big),$$
$$\ldots\ldots, X_1+X_2+B_1+b_{21}-C_1, X_1+b_{11}\Big]$$

It is clear that the term $\sum_{i=1}^{n} C_i$ is independent of the chosen sequence. Therefore minimizing $C_{\max}$ is equivalent to minimizing $\sum_{i=1}^{n} Y_i$.

An examination of the lower bound on $\sum_{i=1}^{n} Y_i$ instead of $\sum_{i=1}^{n} Y_i$ itself makes it theoretically easier to determine the sequence of the batches.

Denote the lower bound

$$L = \max\Big[\sum_{i=1}^{n} X_i + \sum_{i=1}^{n-1} B_i - \Big(\sum_{i=1}^{n-1} C_i\Big), \sum_{i=1}^{n-1} X_i$$
$$+ \sum_{i=1}^{n-2} B_i - \Big(\sum_{i=1}^{n-2} C_i\Big), \ldots\ldots, X_1 + X_2 + B_1 - C_1, X_1\Big] \le \sum_{i=1}^{n} Y_i$$

Let $H_v = \sum_{i=1}^{v-1} B_i - \sum_{i=1}^{v-1} C_i$, $v = 1,2,\ldots\ldots n$, and $K_u = u_t - \sum_{i=1}^{u-1} B_i$, $u = 1,2,\ldots\ldots,n$, then $L = \max_{1\le u\le v\le n} [\max K_u + H_v] = \max_{1\le u\le v\le n} [K_u + H_v]$ where $\max_{u\le v} K_u = K_v$

Consider a sequence S that contains a pair of adjacent batches, J and $J$+1. Also consider a new sequence, S' , in which batches $J$ and $J$+1 are interchanged. It is obvious that if batch $J$ precedes $J$+1 then the following relation holds. $\max(K_J+H_J, K_{J+1}+H_{J+1}) \le \max(K'_J+H'_J, K'_{J+1}+H'_{J+1})$ where

$$K_J + H_J = (J+1)t - \sum_{i=1}^{J-1} B_i + \sum_{i=1}^{J} C_i - \sum_{i=1}^{J-1} C_i - t - B_J$$

$$K_{J+1} + H_{J+1} = (J+1)t - \sum_{i=1}^{J-1} B_i + \sum_{i=1}^{J} B_i - \sum_{i=1}^{J-1} C_i - B_J - C_J$$

$$K'_J + H'_J = (J+1)t - \sum_{i=1}^{J-1} B_i + \sum_{i=1}^{J} B_i - \sum_{i=1}^{J-1} C_i - t_J - B_{J+1}$$

$$K'_{J+1} + H'_{J+1} = (J+1)t - \sum_{i=1}^{J-1} B_i + \sum_{i=1}^{J} B_i - \sum_{i=1}^{J-1} C_i - B_{J+1} - C_{J+1}$$

So we have implies that $(K_J + H_J, K_{J+1} + H_{J+1}) \le \max(K'_J+H'_J, K'_{J+1}+H'_{J+1})$. And then, the values of $t+B_J$, $B_J+C_J$, $t+B_{J+1}$ and $B_{J+1} + C_{J+1}$ could be used to determine the order of batch $J$ and batch $J$+1.

**Improvement: perform job exchanges**

Since the lower bound on $\sum_{i=1}^{n} Y_i$ instead of $\sum_{i=1}^{n} Y_i$ itself is used, the sequence may be improved by the pairwise interchange for jobs between two batches.

Consider a sequence S that includes a pair of adjacent batches, $J$ and $J$+1, such that the total idle time incurred by these two consecutive batches in the third stage is $(Y_{J+1}, Y_{J+2})$. Now construct a new sequence, $S'$ in which a job in batch $J$ is exchanged with a job in batch $J$+1 and the corresponding idle time incurred is denoted as $(Y'_{J+1}+Y'_{J+2})$.

$$Y_{J+1} + Y_{J+2} = \max\left[\begin{array}{l} \sum_{i=1}^{J+1} X_i + \sum_{i=1}^{J+2} X_i + \sum_{i=1}^{J} B_i + \sum_{i=1}^{J+1} B_i - \\ \left(\sum_{i=1}^{J} Y_i + \sum_{i=1}^{J+1} Y_i + \sum_{i=1}^{J} C_i + \sum_{i=1}^{J+1} C_i\right), \\ \sum_{i=1}^{J+2} X_i + \sum_{i=1}^{J+1} B_i - \left(\sum_{i=1}^{J+1} Y_i + \sum_{i=1}^{J+1} C_i\right), \\ + \sum_{i=1}^{J+1} X_i \sum_{i=1}^{J} B_i - \left(\sum_{i=1}^{J} Y_i + \sum_{i=1}^{J} C_i\right), 0 \end{array}\right]$$

$$Y'_{J+1} + Y'_{J+2} = \max\left[\begin{array}{l} \sum_{i=1}^{J} X_i + X'_{J+1} + \sum_{i=1}^{J} X_i + X'_{J+1} + X'_{J+2} \\ + \sum_{i=1}^{J-1} B_i + B'_J + \sum_{i=1}^{J+1} B_i \\ - \left(\sum_{i=1}^{J} Y_i + \sum_{i=1}^{J} Y_{J+1} + \sum_{i=1}^{J-1} C_i + C'_J + \sum_{i=1}^{J+1} C_i\right), \\ \sum_{i=1}^{J} X_i + X'_{J+1} + X'_{J+2} + \sum_{i=1}^{J+1} B_i \\ - \left(\sum_{i=1}^{J} Y_i + Y'_{J+1} + \sum_{i=1}^{J+1} C_i\right), \\ \sum_{i=1}^{J} X_i + X'_{J+1} + \sum_{i=1}^{J-1} B_i + B'_J \\ - \left(\sum_{i=1}^{J} Y_i + \sum_{i=1}^{J-1} C_i + C'_i\right), 0 \end{array}\right]$$

where $X'_{J+1} = \max\left[(J+1)t - \sum_{i=1}^{J} X_i - \sum_{i=1}^{J-1} B_i - B'_J, 0\right]$ and

$$X'_{J+2} = \max\left[(J+2)t - \sum_{i=1}^{J} X_i - X'_{J+1} \sum_{i=1}^{J+1} B_i, 0\right]$$

Clearly, if $(Y_{J+1}+Y_{J+2})<(Y'_{J+1}+Y'_{J+2})$ and $(X_{J+1}+X_{J+2})<(X'_{J+1}+X'_{J+2})$ then no exchange of jobs in batch with is made.

This can be seen by examining the three possibilities.

Case 1. $Y_{J+1}+Y_{J+2}=\sum_{i=1}^{J+1}X_i+\sum_{i=1}^{J+2}X_i+\sum_{i=1}^{J}B_i+\sum_{i=1}^{J+1}B_i$

$$-\left(\sum_{i=1}^{J}Y_i+\sum_{i=1}^{J+1}Y_i+\sum_{i=1}^{J}C_i+\sum_{i=1}^{J+1}C_i\right)$$

$$Y'_{J+1}+Y'_{J+2}=\sum_{i=1}^{J}X_i+X'_{J+1}+\sum_{i=1}^{J}X_i+X'_{J+1}$$

$$+X'_{J+1}+\sum_{i=1}^{J-1}B_i+B'_J+\sum_{i=1}^{J+1}B_i$$

$$-\left(\sum_{i=1}^{J}Y_i+\sum_{i=1}^{J}Y_i+Y'_{J+1}+\sum_{i=1}^{J-1}C_i+C'_J+\sum_{i=1}^{J+1}C_i\right)$$

So, $X_{J+1}+X_{J+2}-\sum_{i=1}^{J}X_i-\sum_{i=1}^{J-1}B_i+\sum_{i=1}^{J}Y_i$

$$-\sum_{i=1}^{J-1}C_i\le X'_{J+1}+X'_{J+2}-\sum_{i=1}^{J}X_i-\sum_{i=1}^{J-1}B_i$$

$+\sum_{i=1}^{J}Y_i-\sum_{i=1}^{J-1}C_i$ which is equivalent to $X_{J+1}+X_{J+2}\le X'_{J+1}+X'_{J+2}$ and thus no exchange is made in this case.

Case 2. $Y_{J+1}+Y_{J+2}=\sum_{i=1}^{J+2}X_i+\sum_{i=1}^{J+1}B_i$

$$-\left(\sum_{i=1}^{J+1}Y_i+\sum_{i=1}^{J+1}C_i\right)$$

$$Y'_{J+1}+Y'_{J+2}=\sum_{i=1}^{J}X_i$$

$$+X'_{J+1}+X'_{J+2}+\sum_{i=1}^{J+1}B_i-\left(\sum_{i=1}^{J}Y_i+Y'_{J+1}+\sum_{i=1}^{J+1}C_i\right)$$

So $\sum_{i=1}^{J+2}X_i+\sum_{i=1}^{J+1}B_i-\left(\sum_{i=1}^{J+1}Y_i+\sum_{i=1}^{J+1}C_i\right)$

$$\le\sum_{i=1}^{J}X_i+X'_{J+1}+X'_{J+2}+\sum_{i=1}^{J+1}B_i$$

$$-\left(\sum_{i=1}^{J}Y_i+Y'_{J+1}+\sum_{i=1}^{J+1}C_i\right)$$, It follows that $X_{J+1}+X_{J+2}\le X'_{J+1}+X'_{J+2}$ which is equivalent to case 1 and thus no exchange is made.

Case 3. $Y_{J+1}+Y_{J+2}=\sum_{i=1}^{J+1}X_i+\sum_{i=1}^{J}B_i$

$$-\left(\sum_{i=1}^{J}Y_i+\sum_{i=1}^{J}C_i\right)$$

$$Y'_{J+1}+Y'_{J+2}=\sum_{i=1}^{J}X_i+X'_{J+1}+\sum_{i=1}^{J-1}B_i+B'_J$$

$$-\left(\sum_{i=1}^{J}Y_i+\sum_{i=1}^{J-1}C_i+C'_J\right)$$, it follows that $X_{J+1}+B_J-C_J\le X'_{J+1}+B'_J-C'_J$ where

$$X_{J+1}=\max\left[(J+1)t-\sum_{i=1}^{J}X_i-\sum_{i=1}^{J}B_i,0\right]$$ and

$$X'_{J+1}=\max\left[(J+1)t-\sum_{i=1}^{J}X_i-\sum_{i=1}^{J-1}B_i-B'_J,0\right]$$

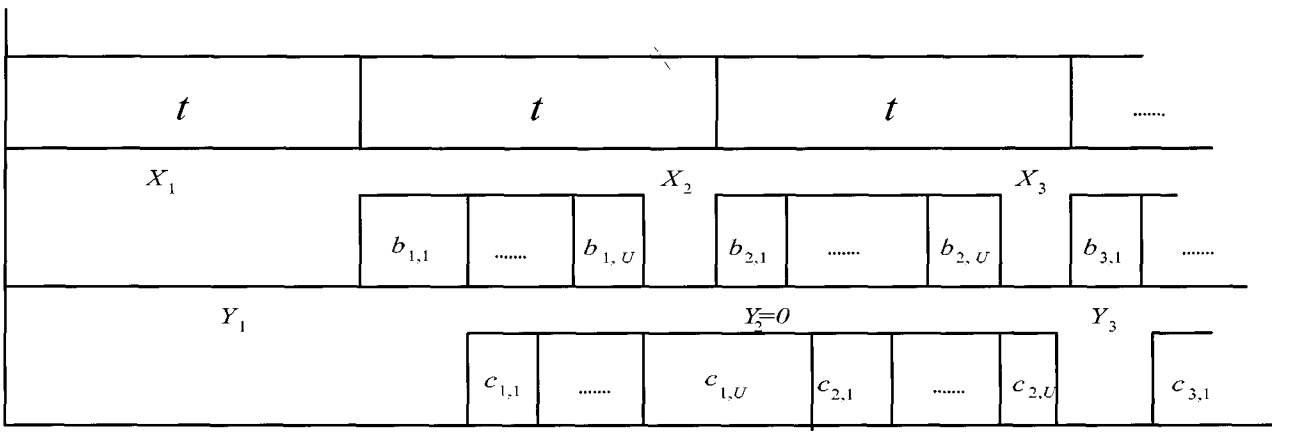


Fig. 1. The left-shifting on the discrete processor in stage 3.

By the above three cases, we know if the following relationship holds then no job exchange is made.

$$\max\left[(J+1)t - \sum_{i=1}^{J} X_i - \sum_{i=1}^{J} B_i + B_J - C_J, B_J - C_J\right]$$

$$\leq \max\left[(J+1)t - \sum_{i=1}^{J} X_i - \sum_{i=1}^{J} B_i + B'_J + B'_J - C'_J, B'_J - C'_J\right]$$

, which is equivalent to $C_J - B_J > C'_J - B'_J$

**The step of the heuristic algorithm.**

With the above results, the initial schedule can be determined by using the values of $t + B_J$, $B_J + C_J$, $t + B_{J+1}$ and $B_{J+1} + C_{J+1}$ that, for a batching sequence the rule $\max(t + B_{J+1}, B_{J+1} + C_{J+1}) \leq \max(t + B_J, B_J + C_J)$ satisfies the optimal sequence, besides, the rule $C_J - B_J > C'_J - B'_J$ implement the improvement of a given batching sequence. If we assume that the processing time of each batch is evenly distributed to jobs within each batch, the rules $\max(t + B_{J+1}, B_{J+1} + C_{J+1}) \leq \max(t + B_J, B_J + C_J)$ and $C_J - B_J > C'_J - B'_J$ can be transformed into $\max\left(\left\lceil \frac{t}{U} \right\rceil + b_{J+1,k}, b_{J+1,k}\right) \leq \max\left(\left\lceil \frac{t}{U} \right\rceil + b_{Jk}, b_{Jk} + c_{Jk}\right)$ for $J=1,2,..,n$, and $c_{ij} - b_{ij} > c_{i+1,j'} - b_{i+1,j'}$ respectively.

The detailed steps of the heuristic is as follows:

Phase 1. Initialization: Determine the initial schedule.

Step 1. Le $t^* = \left\lceil \frac{t}{U} \right\rceil$; $n = \left\lceil \frac{N}{U} \right\rceil$. $t_k^b = t^* + b_k$, $b_k^c = b_k + c_k$, $k = 1,2,...,N$.

Step 2. Set $m_k = \max\left(t_k^b, b_k^c\right)$, $k = 1,2,...,N$. Allocate all jobs to the batches by the ascending order of $m_k$ according to the LOE rule. Denote the resulting sequence as the current schedule and calculate the makespan.

Phase 2. Improvement: Perform the job exchange.

Step 1. Apply the pairwise interchange methods to examine whether it is possible to exchange the $j$th job in the ith batch with the the $j'$th job in the $i$+1th batch such that $c_{ij} - b_{ij} < c_{i+1,j'} - b_{i+1,j'}$ for $i = 1.....n$; $j = 1.....U$. If yes, calculate the corresponding makespan. If the makespan is smaller than the one in the current schedule, set the new schedule as the current schedule.

Step 2. Applying Johnson's rule to the jobs within each batch of the current schedule based on the values of $b_{ij}$ and $c_{ij}$, $i = 1.....n$; $j = 1.....U$.

Step 3. Calculate the makespan.

**Calculation of Lower Bound**

By the *LOE* batching and property 1 that $(N-(n-1)*U)$ jobs are contained in the last batch. Denote the set $L = \{(n-1)*U + 1, ..,N\}$ and order the jobs in ascending order of $b_i$, $i = 1.....N$, the lower bound $LB_1$ is calculated as follows:

$$LB_1 = [(nt + \sum_{j \in L} b_j + \min_j (c_j)]$$

Also, let $\alpha$ be the makespan obtained by applying Johnson's algorithm using the values of $b_i$ and $c_i$, for $i = 1,2,..N$. The lower bound $LB_2$ is calculated as follows:

$$LB_2 = (t + \alpha)$$

The lower bound $LB = \max(LB_1, LB_2)$.

## IV. Case 2 of $\delta \rightarrow \delta \rightarrow \beta$ system.

**Theorem 2.** *A full Batch job schedule minimizes* $C_{\max}$.

**Proof.** The proof can be obtained by shifting-forward some jobs in late batches.

**Definition 2.** FOE $(N, U)$ batching in which the first batch contains the first $[N-(n-1)U]$ jobs and the next $[N-(n-1)U]$ batches all are of full capacity.

**A. Mixed integer programming**

The mathematical programming is similar to case 1 except that batch processor at the third stage as opposed to the first stage.

**B. The heuristic algorithm**

Theorem 2 shows that it is sufficient for the optimal solution to consider only the full batch schedule for stage 3. For stages 1 and 2, the problem is treated as a two-machine flowshop subproblem and Johnson's algorithm is adopted. Again we are attempting to improve the schedule by job exchange. Define the idle time before the ith batch in stage 3 as $Y_i$. Also, let $A_i = \sum_{j=1}^{U} a_{ij}$, $B_i = \sum_{j=1}^{U} b_{ij}$, $X_i = \sum_{j=1}^{U} x_{ij}$, $i = 1,2,3..., n$, as that in Fig. 2.

The value of the $Y_n$ can be given by the following recurrence relationships.

$$Y_n = \max\left[\left(\sum_{i=1}^{n} X_i + \sum_{i=1}^{n} B_i\right) - \left(\sum_{i=1}^{n-1} Y_i + (n-1)t\right), 0\right], \text{ and}$$

$$\sum_{i=1}^{n} Y_i = \max\left[\begin{array}{l}\left(\sum_{i=1}^{n} X_i + \sum_{i=1}^{n} B_i\right) - (n-1)\,t,\ \left(\sum_{i=1}^{n-1} X_i + \sum_{i=1}^{n-1} B_i\right) \\ -(n-2)\,t,\ \ldots..,\ \left(\sum_{i=1}^{2} X_i + \sum_{i=1}^{2} B_i\right) - t, \\ X_1 + B_1\end{array}\right]$$

Consider a sequence S that contains a pair of adjacent batches $I$ and $I$+1 with $(Y_I+Y_{I+1})$ being the sum of idle time before batches $I$ and $I$+1. Also consider a new sequence $S$', in which the $j$th job in batch $I$ and the $j$'th job in batch $I$+1 are interchanged with $(Y_I+Y_{I+1})$ being the sum of idle time before batches $I$ and $I$+1. The situation can then be described as in Fig. 3.

By Fig. 3, The following relations are derived.

$$x_{Ij} = \max\lfloor l_1 + a_{Ij} - l_2, 0\rfloor$$

$$x_{I,j+1} = \max\lfloor l_1 + a_{Ij} + a_{I,j+1} - (l_2 + x_{Ij} + b_{Ij}), 0\rfloor$$

$$x_{I+1,j'} = \max\lfloor l_1 + a_{Ij} + \ldots + a_{I+1,j'} - (l_2 + x_{Ij} + \ldots + x_{I+1,j'-1} + b_{Ij} + \ldots + b_{I+1,j'-1}), 0\rfloor$$

$$x_{I+1,j'+1} = \max\lfloor l_1 + a_{Ij} + \ldots + a_{I+1,j'} - (l_2 + x_{Ij} + \ldots + x_{I+1,j'} + b_{Ij} + \ldots + b_{I+1,j'}), 0\rfloor$$

$$x'_{Ij} = \max\lfloor l_1 + a_{I+1,j'} - l_2, 0\rfloor$$

$$x'_{I,j+1} = \max\lfloor l_1 + a_{I+1,j'} + a_{I,j+1} - (l_2 + x'_{Ij} + b_{I+1,j'}), 0\rfloor$$

$$x_{I+1,j'} = \max\lfloor l_1 + a_{I+1,j'} + \ldots + a_{Ij} - (l_2 + x'_{Ij} + \ldots + x'_{I+1,j'-1} + b_{I+1,j'} + \ldots + b_{I+1,j'-1}), 0\rfloor$$

$$x_{I+1,j'+1} = \max\lfloor l_1 + a_{Ij} + \ldots + a_{I+1,j'+1} - (l_2 + x_{Ij}$$

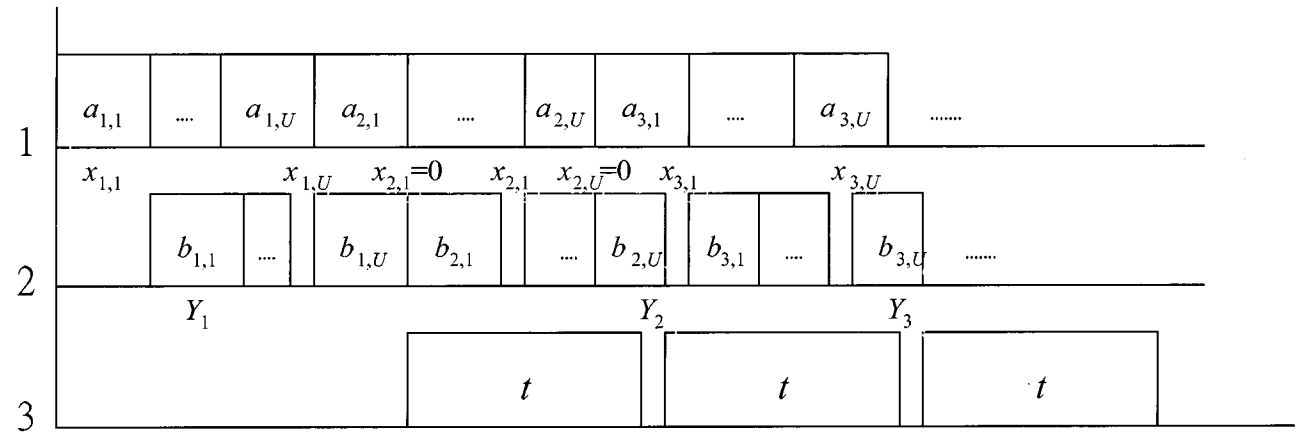


Fig. 2. Gantt chart for $\delta$–$\delta$–$\beta$ case.

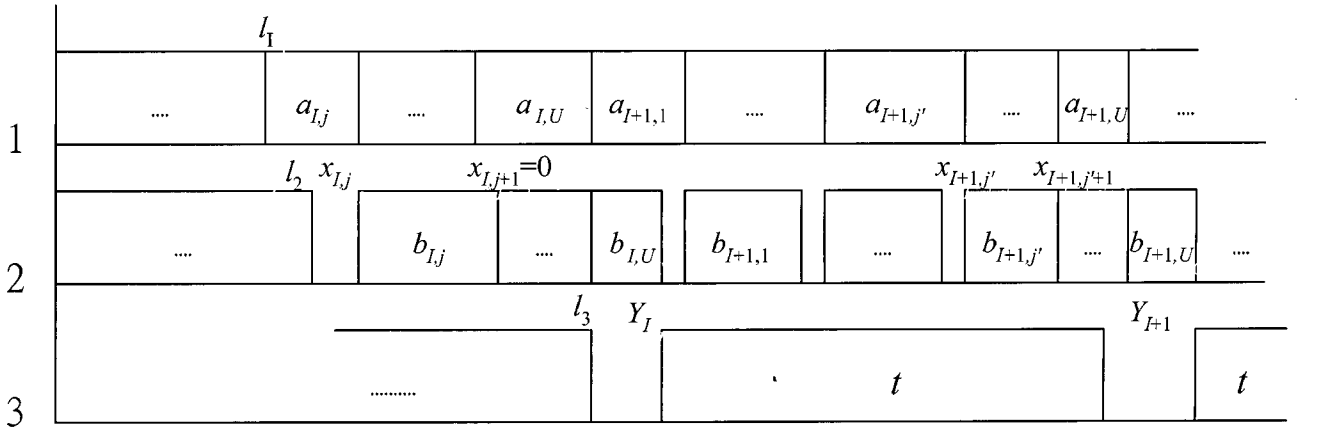


Fig. 3. The Gantt chart illustrate the relationship between batches $I$ and $I$+1.

$$+ \ldots + x_{I+1,j'} + \ldots + b_{I+1,j'}),0\rfloor$$

$$Y_I = \max\lfloor l_2 + b_{Ij} + \ldots + b_{IU} + x_{Ij} + \ldots + x_{IU} - (l_3),0\rfloor$$

$$Y_{I+1} = \max\lfloor l_2 + b_{Ij} + \ldots + b_{I+1,U} + x_{Ij} + \ldots + x_{I+1,U} - (l_3 + Y_I + t),0\rfloor$$

$$Y'_{I+1} = \max\lfloor l_2 + b_{I+1,j'} + \ldots + b_{I+1,U} + x'_{Ij} + \ldots + x'_{I+1,U} - (l_3 + Y'_I + t),0\rfloor$$

$$Y'_I = \max\lfloor l_2 + b_{I+1,j'} + \ldots + b_{IU} + x'_{Ij} + \ldots + x'_{IU} - (l_3),0\rfloor$$

Accordingly,

$$Y_I + Y_{I+1} = \max\begin{bmatrix} l_2 + b_{Ij} + \cdots + b_{I+1,U} + x_{Ij} + \cdots + x_{I+1,U} - l_3 - t, \\ l_2 + b_{Ij} + \cdots + b_{IU} + x_{Ij} + \cdots + x_{IU} - l_3, 0 \end{bmatrix}$$

$$Y'_I + Y'_{I+1} = \max\begin{bmatrix} l_2 + b_{I+1,j'} + \cdots + b_{I+1,U} + x'_{Ij} + \cdots + x'_{I+1,U} - l_3 - t, \\ l_2 + b_{I+1,j'} + \cdots + b_{IU} + x'_{Ij} + \cdots + x'_{IU} - l_3, 0 \end{bmatrix}$$

Thus it will be shown by the following two cases corresponding to the situation of whether the $j$th job being exchanged with the $j'$th job.

Case 1. Suppose $Y_I + Y_{I+1} = l_2 + b_{Ij} + \ldots + b_{I+1,U} + x_{Ij} + \ldots + x_{I+1,U} - l_3, - t$ and $Y'_I + Y'_{I+1} = (l_2 + b_{I+1,j} + \ldots + b_{I+1,U} + x'_{Ij} + \ldots + x'_{I+1,U} - l_3 - t)$. It is then sufficient to show that if the relation $x_{Ij} + \ldots + x_{I+1,U} < x'_{Ij} + \ldots + x'_{I+1,U}$ holds, then the exchange is unnecessary. Since this relation involves only the idle time in stage 2, therefore Johnson's rule is applied to obtain the optimal solution. That is, if $(a_{Ij}, b_{I+1,j'}) < \min(a_{I+1,j'}, b_{Ij})$, then no exchange for the $j$th job in batch $I$ with the $j'$th job in batch $I+1$ is made.

Case 2. Suppose $Y_I + Y_{I+1} = (l_2 + b_{Ij} + \ldots + b_{IU} + x_{Ij} + \ldots + x_{IU} - l_3)$ and $Y'_I + Y'_{I+1} = l_2 + b_{I+1,j'} + \ldots + b_{IU} + x'_{Ij} + \ldots + x'_{IU} - l_3$, it is then sufficient to show that the exchange is unnecessary if the relation.

$$l_2 + b_{Ij} + \ldots + b_{IU} + x_{Ij} + \ldots + x_{IU} - l_3 < l_2 + b_{I+1,j'} + \ldots + b_{IU} + x'_{Ij} + \ldots + x'_{IU} - l_3 \text{ holds.}$$

This relation follows that

$$b_{Ij} + \max\lfloor l_1 + a_{Ij} + \ldots + a_{IU} - (l_2 + b_{Ij} + \ldots + b_{I,U-1}),\ldots,l_1 + a_{Ij} + a_{I,j+1} - (l_2 + b_{Ij}), l_1 + a_{Ij} - l_2\rfloor$$

$$< b_{I+1,j'} + \max\begin{bmatrix} l_1 + a_{I+1,j'} + \cdots + a_{IU} - (l_2 + b_{I+1,j'} \\ + \cdots + b_{I,U-1}),\ldots, l_1 + a_{I+1,j'} + a_{I,j+1} \\ - (l_2 + b_{I+1,j'}), l_1 + a_{I+1,j'} - l_2 \end{bmatrix}$$

which is equivalent to

$$\max\lfloor l_1 + a_{Ij} + \ldots + a_{IU} - (l_2 + b_{I,j+1} + \ldots + b_{I,U-1}), \ldots, l_1 + a_{I,j+1} - l_2, l_1 + a_{Ij} + b_{Ij} - l_2\rfloor$$

$$< \max\begin{bmatrix} l_1 + a_{I+1,j'} + \ldots + a_{IU} - (l_2 + b_{I,j+1} + \ldots + b_{I,U-1}), \\ \ldots, l_1 + a_{I+1,j'} + a_{I,j+1} - l_2, \\ l_1 + a_{I+1,j'} + b_{I+1,j'} - l_2 \end{bmatrix}$$

that is, if $a_{Ij} < a_{I+1,j'}$ and $a_{Ij} + b_{Ij} < a_{I+1,j'} + b_{I+1,j'}$ then $(Y_I + Y_{I+1}) < (Y'_I + Y'_{I+1})$.

With the above results, we see that if $(a_{Ij}, b_{I+1,j'}) < \min(a_{I+1,j'}, b_{Ij})$ and $a_{Ij} + b_{Ij} < a_{I+1,j} + b_{I+1,j'}$ hold then the exchange is not applied.

**Property 3.** Sequencing the jobs on the discrete processors in stage 1 and 2 using Johnson's algorithm and in stage 3 applying the FOE (first only empty) rule will minimize makespan if the following condition hold. $\sum_{i=1}^{k-1} B_i \geq (k-1) \times t,\ k = 2,3,.n.$

**Proof.** Since $\sum_{i=1}^{k-1} B_i \geq (k-1) \times t,\ k = 2,3,.n$, therefore the makespan is determined by $\sum_{i=1}^{n} (B_i + X_i) + t$, where t is a constant. This implies that the makespan is determined by the processing of stage 1 and stage 2 and the Johnson's algorithm is thus applied. Moreover, as Ikura and Gimple [9] showed that FOE batching minimized the makespan for a single batch processor with dynamic job arrival times. This completes the proof.

**The steps of the heuristic algorithm.**

Step 1: **Determine the initial sequence.** Apply Johnson's algorithm using the values of $a_i$ and $b_i$, $i = 1,2,..,N$.

Step 2. **Determine the batch allocation.** Let $n = \left\lceil \frac{N}{U} \right\rceil$. Allocate each job sequentially according to the initial sequence and the FOE rule. Denote $H_i$, $i = 1,2,..., n$ as the job set for each batch.

Step 3. **Improve the schedule.** Apply the adjacent pairwise interchange method to examine whether it is possible to exchange the $j$th job in $H_i$ with the $j'$th job in $H_{i+1}$. Let $a_{ij}$ and $b_{ij}$ be the

$j$th job in $H_i$, $a_{i+1,j'}$ and $b_{i+1,j'}$ be the $j'$th job in $H_{i+1}$. Exchange job $j$ with $j'$ such that the following conditions hold (i) $\min(a_{ij}, b_{i+1,j}) < \min(a_{i+1,j'}, b_{ij})$ (ii) $a_{ij} + b_{ij} < a_{i+1,j'} + b_{i+1,j'}$ and (iii) the makespan after exchange is better than the original one.

Step 4. Continue step 3 until all jobs have been checked.

**Calculation of Lower Bound**

Let $\alpha$ be the makespan by applying Johnson's algorithm using the values of $a_i$ and $b_i$, $i = 1,2,..,N$. The lower bound calculated is $LB = (t + \alpha)$.

## V. Computational results

The objective of the computational experiments described in this section is to evaluate the performance of the two heuristics presented in the previous sections. Experimental results are divided into two parts. For $N \leq 35$, the heuristic solutions were compared to the optimal solution or the lower bound value obtained by the mixed integer programming. For moderate or large problem, the heuristic solutions are evaluated with the lower bound calculated in this paper. All experimental tests are run on a personal computer with Pentium IV 1.4G MHz CPU. The integer programming models solve the problems using LINGO Extended 5.0 software package. Whenever the integer programming did not give the solution within the upper limit time of 7200 CPU seconds, the lower bound value was recorded. The lower bound value instead of the optimal solution will be used to evaluate the performance of the heuristic algorithm. Therefore, the actual values of solution quality of the proposed heuristic algorithms are slightly higher than those shown in Table 1 and 2.

In the design of the test problem, we considered various factors: number of jobs $N$, batch capacity $U$, batch processing time $t$; and processing times of each job on two discrete processor $b_k$ and $c_k$ for case 1 and $a_k$ and $b_k$ for case 2. The details of the factors in the computational experiments are listed below:

$U$, the capacity of the batch processor, was equal to 4,5, or 6.
$t$, the batch processing time, was equal to 20, or 30.
$a_k$, $b_k$ and $c_k$, the discrete processing time of job $k$, was uniformly distributed over the discrete interval [1,10].

Ten test problems were generated and the average performance measure of every 10 test problems were calculated. To compare the performances of the heuristic for each case, the following formula is applied to determine the solution quality of the heuristic scheduling algorithm.

Solution quality =[(2 x optimal or lower bound)–heuristic] / [(optimal or lower bound )] x 100%

For small problem, $N$ was set from $2U$ to 32, or 35. Table 1 and 2 summarize the computational results of the heuristics for case 1 and 2 respectively. The parameters of this experiment were selected generate 'difficult' problem. Define the expected value of the discrete processing time as $\mu$ , $\mu = 5$ in our experiment. In all the experiments, the ratio $\frac{t}{U \times \mu}$ had a significant effect on the relative performance of the heuristic versus the optimal solution or lower bound: performances increase as the ratio $\leq 0.5$ or $\geq 1.5$ since less deliberate idle time contained in these schedules.

As both tables show, the average computing time of the integer programming model drastically increases as the number of jobs increases in these two cases. When the number of jobs is greater than 20, some of the testing examples cannot be solved optimally within the allowable pivoting limit. Therefore, the average execution time of the integer programming models is somewhat underestimated and the actual execution time is higher than that shown in both tables. The average execution time of the heuristic scheduling algorithms will slowly increase when the number of jobs increases. For case 1, when the number of jobs increases to 32, the integer programming model requires 6500 seconds on average to produce a solution. The average execution time of the heuristic algorithm, however, is within 0. 003 seconds. For case 2, when the number of jobs increases to 35, the integer programming model requires 6900 seconds on average to produce a solution. The average execution time of the heuristic algorithm, however, is less than 0.011 seconds.

The average solution quality is above 95.00% and 95.89% for 330 testing problems of case 1 and 340 testing problems of case 2 respectively. The corresponding average solution qualities are shown in Fig. 4 and 5 respectively. This is a conservative estimate of the solution quality since the lower bound values instead of the optimal values are used to evaluate the solution qualities.

For moderate and large problem, the number of jobs, $N$, is set to be 50, 100, 500 and 1000, since $N = 1000$ is enough to what occurs in industry. Other parameters including $U$, $t$, $a_k$, $b_k$ and $c_k$ remained the same as those in small-sized test problem since these values generate 'difficult' problem in which deliberate idle time contained

Table 1. Execution time comparison and solution quality of the heuristic algorithm for $\beta$–$\delta$–$\delta$ case.

| Processing time of the batch processor. (t) | Capacity of the batch processor (U) | Number of job (N) | Average execution time of Integer programming. (sec.) | Average execution time of heuristic scheduling algorithm. (sec.) | Average solution quality. (%) |
|---|---|---|---|---|---|
| t=20 | 4 | 8 | 202.5 | 0.0015 | 98.87114 |
| | | 10 | 502.8 | 0.0017 | 96.71878 |
| | | 16 | 1431.5 | 0.0018 | 97.12836 |
| | | 18 | 2193.5 | 0.0018 | 97.34228 |
| | | 20 | 2654.9 | 0.0021 | 95.11605 |
| | | 22 | 3242.1 | 0.0021 | 95.16492 |
| | | 25 | 3827.6 | 0.0020 | 97.21026 |
| | | 27 | 4491.5 | 0.0020 | 94.81365 |
| | | 32 | 6445.7 | 0.0021 | 94.96481 |
| | | 35 | 7121.6 | 0.0030 | 94.54146 |
| | 5 | 10 | 603.2 | 0.0017 | 99.08394 |
| | | 16 | 1431.5 | 0.0019 | 95.20034 |
| | | 18 | 2262.5 | 0.0019 | 96.80851 |
| | | 20 | 2762.4 | 0.0021 | 98.54699 |
| | | 22 | 3426.2 | 0.0021 | 96.53823 |
| | | 25 | 4210.3 | 0.0022 | 95.87561 |
| | | 27 | 4961.8 | 0.0021 | 94.24476 |
| | | 32 | 6518.7 | 0.0028 | 94.67822 |
| | | 35 | 6847.6 | 0.0031 | 93.82346 |
| t=30 | 4 | 8 | 267.4 | 0.0015 | 95.23105 |
| | | 10 | 487.5 | 0.0015 | 96.36929 |
| | | 16 | 1517.2 | 0.0019 | 95.20656 |
| | | 18 | 2263.5 | 0.0018 | 92.15222 |
| | | 20 | 2542.9 | 0.0019 | 93.94391 |
| | | 22 | 3147.1 | 0.0019 | 91.92822 |
| | | 27 | 4391.5 | 0.0023 | 91.09435 |
| | | 32 | 6345.7 | 0.0024 | 91.42023 |
| | 6 | 16 | 1837.4 | 0.0020 | 94.55525 |
| | | 18 | 2533.5 | 0.0019 | 93.71325 |
| | | 20 | 2830.8 | 0.0018 | 93.05086 |
| | | 22 | 3437.3 | 0.0021 | 92.57516 |
| | | 27 | 4221.4 | 0.0021 | 94.51936 |
| | | 32 | 6635.6 | 0.0023 | 92.56374 |

in two discrete processors. The experiments show that both heuristic find solutions for each of these instances in no more than 2 of a second. Table 3 summarizes the average solution quality for both heuristics. As seen in the table, the heuristic solution get slightly worse as the capacity of the batch gets larger. The reason is that is may incur more job combination within each batch. The performances appear in descending trend as the value of $N$ increase. This implies that the proposed heuristics do not work well for the problem instances with the

Table 2. Execution time comparison and solution quality of the heuristic for $\delta$–$\delta$–$\beta$ case.

| Processing time of the batch processor. (t) | Capacity of the batch processor (U) | Number of job (N) | Average execution time of Integer programming. (sec.) | Average execution time of heuristic scheduling algorithm. (sec.) | Average solution quality. (%) |
|---|---|---|---|---|---|
| t=20 | 4 | 8 | 202.5 | 0.0044 | 99.11326 |
| | | 10 | 483.8 | 0.0047 | 97.82994 |
| | | 16 | 1431.5 | 0.0063 | 95.59362 |
| | | 18 | 2241.6 | 0.0058 | 94.15464 |
| | | 20 | 2773.4 | 0.0061 | 96.00602 |
| | | 22 | 3213.8 | 0.0072 | 96.85147 |
| | | 25 | 3726.6 | 0.0070 | 96.00779 |
| | | 27 | 4591.5 | 0.0081 | 92.76741 |
| | | 32 | 6445.7 | 0.0083 | 92.48758 |
| | | 35 | 7081.6 | 0.0085 | 93.42699 |
| | 5 | 10 | 603.2 | 0.0044 | 98.58747 |
| | | 16 | 1431.5 | 0.0058 | 97.81804 |
| | | 18 | 2262.5 | 0.0055 | 96.10165 |
| | | 20 | 2762.4 | 0.0063 | 98.21363 |
| | | 22 | 3414.5 | 0.0071 | 98.94675 |
| | | 25 | 3910.3 | 0.0075 | 96.24081 |
| | | 27 | 4469.2 | 0.0080 | 96.26341 |
| | | 32 | 6292.5 | 0.0113 | 96.08116 |
| t=30 | 4 | 8 | 247.2 | 0.0035 | 93.75128 |
| | | 10 | 539.5 | 0.0055 | 96.36929 |
| | | 16 | 1772.2 | 0.0069 | 95.20656 |
| | | 18 | 2453.6 | 0.0071 | 95.08833 |
| | | 20 | 2846.9 | 0.0079 | 96.43516 |
| | | 22 | 3263.1 | 0.0076 | 96.78393 |
| | | 27 | 4679.3 | 0.0078 | 94.96971 |
| | | 32 | 6625.7 | 0.0079 | 94.22799 |
| | | 35 | 6947.1 | 0.0075 | 96.90426 |
| | 6 | 16 | 1731.4 | 0.0063 | 94.55525 |
| | | 18 | 2365.5 | 0.0063 | 93.35717 |
| | | 20 | 2830.8 | 0.0068 | 93.05086 |
| | | 22 | 3277.1 | 0.0071 | 99.31947 |
| | | 27 | 4518.6 | 0.0072 | 95.48007 |
| | | 32 | 6397.2 | 0.0080 | 95.13986 |
| | | 35 | 6841.5 | 0.0081 | 97.31481 |

number of jobs being larger than 500. However, It can be seen that for job size equal to 1000, the average solution quality is still having 84.10% and 86.38% $\beta$–$\delta$–$\delta$ and $\delta$–$\delta$–$\beta$ case respectively. It illustrates that the proposed heuristics can solve large size problems sub-optimally fairly quickly.

## VI. Conclusion

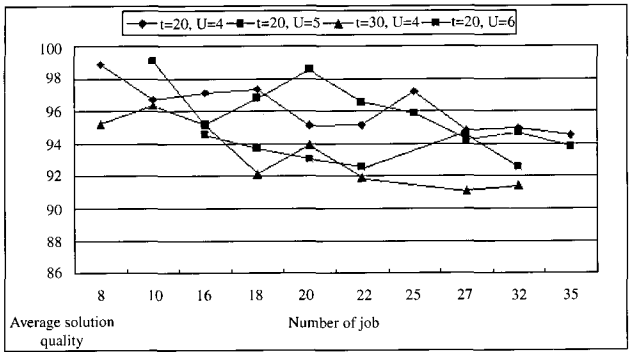


Fig. 4. The average solution quality for $\beta$–$\delta$–$\delta$ case.

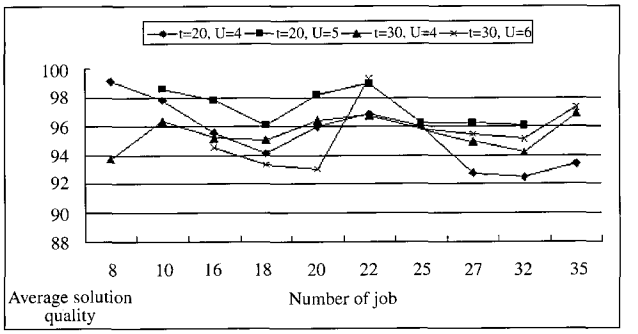


Fig. 5. The average solution quality for $\delta$–$\delta$–$\beta$ case.

Table 3. The solution quality of the heuristic algorithms for $\beta$–$\delta$–$\delta$ and $\delta$–$\delta$–$\beta$ cases.

| Processing time of the batch processor. (t) | Capacity of the batch processor (U) | Number of job (N) | Average solution quality. (%) for $\beta$–$\delta$–$\delta$ | Average solution quality. (%) for $\delta$–$\delta$–$\beta$ |
|---|---|---|---|---|
| t=20 | 4 | 50 | 91.23915 | 92.27294 |
| | | 100 | 90.05168 | 92.13415 |
| | | 300 | 90.92387 | 91.32850 |
| | | 500 | 88.37615 | 88.27631 |
| | | 700 | 86.05293 | 87.47211 |
| | | 1000 | 85.13726 | 86.37703 |
| | 5 | 50 | 90.51031 | 95.23362 |
| | | 100 | 90.51094 | 95.16570 |
| | | 300 | 88.17263 | 93.62703 |
| | | 500 | 87.59462 | 90.57601 |
| | | 700 | 85.29843 | 91.18093 |
| | | 1000 | 86.14802 | 92.33829 |
| t=30 | 4 | 50 | 90.45620 | 95.13215 |
| | | 100 | 88.23002 | 92.87623 |
| | | 300 | 86.90039 | 93.38972 |
| | | 500 | 84.54763 | 91.47652 |
| | | 700 | 84.57398 | 92.33773 |
| | | 1000 | 84.10234 | 91.03973 |
| | 6 | 50 | 92.01436 | 914.24902 |
| | | 100 | 91.37625 | 95.21043 |
| | | 300 | 88.26562 | 92.89731 |
| | | 500 | 87/41372 | 93.62438 |
| | | 700 | 85.41032 | 91.46782 |
| | | 1000 | 85.07052 | 90,27830 |

This paper has considered the three-stage flowshoop with a batch processor at the first and third stage respectively. We extend Ahmadi's three-stage flowshop shop problem in which the second stage is a batch processor. An integer programming and a heuristic for both problems are provided. Our results show that the heuristics are capable of obtaining high-quality solutions with very short computational time.

## References


1. Sung, C.S. and Choung, Y.I. "Minimizing Makespan on a Single Burn-in oven in Semiconductor Manufacturing", *Europ. J. Ops Res.*, Vol. 12, 559-574 (2000).
2. Lee, C.Y., Uzsoy, R. and Martin-Vega, L.A., "Efficient Algorithm for Scheduling Semiconductor Burn-in Operations", *Ops Res.*, Vol. 40, 764-775 (1992).
3. Ahmadi, J.H., Ahmadi, R.H., Dasu, S. and Tang, C.S., "Batching and Scheduling Jobs on Batch and Discrete Processors", *Ops Res.*, Vol. 39, No. 4, 750-763 (1992).
4. Uzsoy, R., Lee C.Y. and Martin-Vega, L.A., "A Review of Production Planning and Scheduling Models in the Semiconductor Industry. Part II: Shop Floor Control", *IIE Trans*, Vol. 26, No. 5, 44-55 (1994).
5. Webster, W. and Baker, K.R., "Scheduling Groups of Jobs on a Single Machine", *Ops Res.*, Vol. 43, No. 4, 692-703 (1995).
6. Cheng, T.C.E. and Wang, G.., "Batching and Scheduling to Minimize the Makespan in the Two-Machine Flowshop", *IIE Trans*, Vol. 30, 447-453 (1998).
7. Lin, B.M.T. and Cheng, T.C.E., "Batch Scheduling in the No-Wait Two-Machine Flowshop to Minimize the Makespan", *Computer Ops Res.* Vol. 28, 613-624 (2001).
8. Wang, J.T. and Chern, M. S., "A Two-Machine Multi-Family Flowshop Scheduling Problem with Two Batch Processors", *J. Chin. Inst. Ind. Eng.*, Vol.18, No.3, 77-85 (2001).
9. Ikura, Y. and Gimple, M., "Efficient Scheduling Algorithm for a Single Batch Processing Machine", *Ops Res. Letter*, Vol. 5, 61-65 (1986).


## 包含單機與批次機台三階段流程型工廠排程問題


蘇 玲 慧　　廖 俊 景

中原大學工業工程系

中壢市普仁 22 號


### 摘 要


本論文探討包含單機與批次機台之三階段流程型工廠排程問題，係 Ahmadi 於 1992 年所提問題的延伸。Ahmadi所探討的是第二階段爲批次機台而第一及第三階段爲單機機台之三階段排程問題。本論文探討其他兩種不同之類型，即批次機台分別在第一及第三階段之三階段排程問題，目標爲最小化總完成時間。兩者皆爲NP-艱難問題，故針對兩種情況各提出一啓發式排程演算法與數學規劃模式。文中亦針對問題提出一些特性與定理，並將之用於啓發式排程演算法與數學規劃模式中。實驗結果顯示啓發式排程演算法的求解品質極佳。



關鍵詞：排程，三階段流程型工廠，批次處理，啓發式方法。