韓國經營科學會誌
第25卷 第2號
2000年 6月

다단계 병렬 흐름생산시스템에서 이송크기가 2이상인 경우의 제품별 로트 투입순서 결정*

김중순** · F. Fred Choobineh***

Transfer Batch Scheduling for a Flexible Flowshop with Identical Parallel Machines at Each Stage*

Joong-Soon Kim** · F. Fred Choobineh***

Abstract

The problem of scheduling n independent jobs on serial stages with identical parallel machines at each stage is considered. Each job lot is allocated evenly to all machines at each stage for processing, and moved in transfer batches between stages. This scheduling strategy is called an identical production pattern. The objective is to find a permutation schedule that minimizes makespan. A branch and bound algorithm is suggested to find an optimal permutation schedule for a transformed problem. A numerical example is presented to illustrate the branch and bound algorithm. Computational results for 640 problems generated randomly show that within a reasonable time the suggested algorithm can be used for transfer batch scheduling in a flexible flowshop.

1. Introduction

A batch production system consisting of m stages and assembling n different products can be modeled as an n-job, m-stage flowshop problem. In such problems the processing sequence of jobs affects various performance measures of the production system. Johnson[10] presented the

* The authors would like to thank Keimynug University for the grant of sabbatical year and University of Nebraska-Lincoln for the facility support.

** Department of Industrial Engineering, Keimyung University
*** Industrial & Management Systems Engineering Department, University of Nebraska, Lincoln, USA.

well-known Johnson's rule, which minimizes makespan for an n-job, two-stage flowshop problem. Since then, considerable efforts have been made to develop a makespan minimizing procedure for an n-job, m-stage problem as well as its extensions. In this paper, two important extensions are considered: lot streaming and multiple identical parallel machines at one or more stages.

Lot streaming is the process of splitting a job's predetermined lot into smaller transfer batches so that its operations can be overlapped and its progress can be accelerated. Thus, under lot streaming a lot is divided into transfer batches that are transferred between stages sequentially. In general, lot streaming improves makespan compared to similar situations without lot streaming[18].

Previous research on lot streaming follows two paths. The first, examplified by Baker and Jia[4], Baker and Pyke[5], Potts and Baker[13], and Trietsch and Baker[18], assumes that the number of transfer batches is prespecified, and focuses on determining the size of transfer batches between stages in a single product situation. The second path, examplified by Baker[2], Cetinkaya and Kayaligil[7], Kim[11, 12], and Vickson and Alfredson[19], assumes that there is no limit on the number of transfer batches, and focuses on sequencing multiple jobs. Since makespan will be minimized when there is just one unit in each transfer batch[18, 19], the size of transfer batches is fixed to be one unit in their analysis. Furthermore, most of the literature in this path is limited to two-stage or special three-stage problems. The result of this path in its current form is not so useful as that of the first path because tracking a larger number of unit-sized transfer batches is often undesirable. The research presented in this paper places no direct restriction on the number of transfer batches and the number of stages, and essentially belongs to the second path. A summary of differences between attributes of our research and others in the second path is shown in <Table 1>.

〈Table 1〉 A summary of attribute differences between our research and others.

Attributes	Current Paper	Vickson & Alfredsson (1992)	Rajendran & Chaudhuri (1992 a, b)	Cetinkaya & Kayaligil (1992)	Kim (1993)	Baker (1995)	Guinet et al. (1996 a,b)	Kim, Kang, Lee (1997)
No. of Stages	multi-stage	two- and three-stage	multi-stage	two-stage	two- and multi-stage	two-stage	two- and multi-stage	two-stage
No. of Machines at Each Stage	multi-machine	1	multi-machine	1	multi-machine	1	multi-machine	multi-machine
Transfer Batch Size	k_i	1	-	1	1	1	-	k_i
Setup considered	Yes	no	no	yes	yes	yes	no	yes
Method	Branch and Bound	similar to Johnson's rule	Branch and Bound	similar to Johnson's rule	Branch & Bound, similar to Johnson's rule	similar to Johnson's rule	heuristic	similar to Johnson's rule
Performance Measure	makespan	makespan flow time	makespan flow time	makespan	makespan	makespan	makespan maximum tardiness	makespan

The impetus for having parallel identical machines at a stage is a practical consideration for balancing flexible flowlines and satisfying the required production rate. The important practical applications of this problem are encountered in industrial settings such as FMS[6], textile manufacturing, pharmaceutical production[9], and PCB assembly line[9, 20, 21]. Much research on flowshop scheduling with identical parallel machines at each stage has been done. Examples are the works of Guinet and Solomon[8], Guinet, Solomon, Kedia and Dussauchoy[9], Kim[11], Rajendran and Chaudhuri[14, 15], Sriskandarajah and Ladet[16], Sriskandarajah and Sethi[17], and Wittrock[20, 21].

The same problem for a two-stage flexible flowshop are considered by Kim, Kang, and Lee[12]. We are unaware of any previous research that addresses the general flowshop sequencing problem in which the two important extensions - lot streaming and parallel identical machines at three or more stages - exist simultaneously. Therefore, this paper presents an algorithm for determining a sequence minimizing makespan for an n-job, m-stage flowshop problem in the presence of lot streaming and parallel identical machines at three or more stages.

The paper is organized as follows. In section 2 we present the problem definition and scheduling strategies for solving the problem. Under the selected scheduling strategy, the original problem is transformed to a tractable flowshop sequencing problem. In section 3 we suggest a lower bound utilized for a branch and bound algorithm, and present the compact steps of the branch and bound algorithm. Section 4 gives a numerical example. Section 5 illustrates computational results of the algorithm. Section 6 concludes and discusses directions for further research.

2. Problem Definition

2.1 Problem definition and scheduling strategies

The following assumptions are made for the flexible flowshop scheduling problem:

- A job's setup time can not be separated from its processing.
- Setup times are sequence independent.
- Every job is processed on all stages and in the same order.
- A machine processes only one unit of a job at a time.
- One unit of a job is processed on one machine at a time.
- Units are not preemptable.
- Buffer storage between stages is unlimited.
- All jobs are available simultaneously at time zero.
- Unit processing times are deterministic and known.
- Interstage transfer times are negligible.

These ten assumptions are made generally in the flowshop scheduling. Our transfer batch flowshop scheduling problem considered here may be stated as follows:

Given n job lots in a batch production system whose transfers are performed in transfer batches between consecutive stages, it is desired to find a schedule that minimizes makespan.

The flowshop considered here has m stages where stage j has p_j identical parallel machines.

The production lot size of job i is Q_i, and its transfer batch size is k_i. We will refer to this flowshop as a flexible flowshop[9].

The flexible flowshop scheduling requires resolving the two problems of allocation and sequencing. The allocation problem means answering which job lot(or its transfer batch) is allocated to which machine. Scheduling techniques usually depend upon the selected scheduling strategy. There are three possible scheduling strategies for the flexible flowshop scheduling problem. Strategy I is to evenly divide a job lot between all machines at each stage. Under this strategy which is called an identical production pattern[12], all machines at a stage must be set up for each job - which makes this strategy inefficient when setup times are large. Strategy II is to set up only one machine at each stage for a job and allocate the job completely to that machine. This strategy requires fewer setups than the former strategy, but requires resolving the two problems of resource allocation and sequencing, which makes the problem of minimizing makespan more challenging[8, 9]. Strategy III is to resolve allocation decision and sequencing decision simultaneously. Under this strategy, the problem will be most complex to find a optimal schedule. Thus, in this paper we adopt the strategy I in order to make the flexible flowshop scheduling problem more tractable.

2.2 Problem transformation

Under the strategy I, all job lots are allocated evenly between all machines at each stage. Thus, we assume that the transfer batch size of job i is a multiple of the least common multiple of the number of parallel machines at all stages, i.e., k_i is proportional to $\prod_{j=1}^{m} p_j$. This assumption about transfer batch sizes is termed assumption (T). The practical feasibility of transfer batch sizes could be derived from two main considerations[18]. First, the number of transfer batches can chosen on the basis of scheduling convenience, so that a predetermined portion of the lot size constitute a scheduling unit. Second, the transfer batch size can be determined to suit the load-capacity of the transfer vehicle used for a particular job lot. For the purpose of scheduling convenience, assumption (T) is to be made in this research.

Furthermore, we assume that the production lot size of job i is a multiple of its transfer batch size, i.e., Q_i is proportional to k_i. This assumption about production lot sizes is termed assumption (P). This assumption can be relaxed to the more mild assumption - which will be discussed later. These assumptions will not impose severe restrictions on the problem, because in a batch production system the quantity of a job lot is large enough to evenly divide between all machines at each stage and make the sizes of all transfer batches at each stage equal. These assumptions allow us to allocate an equal amount of a job's workload to every machine at each stage. These assumptions may impose an additional constraint on the master production scheduling system. However, since master production scheduler has the flexibility to choose the lot size within the constraint, the master scheduling system will not be restricted seriously by such impositions. Moreover, it is well known that the total cost of the Economic Production Quantity model does not increase sharply within the neighborhood of the optimal lot size. Thus, without serious additional

costs, the master production scheduler has the flexibility to adjust the production lot size without severe restrictions. These impositions, however, will be rare, because generally the number of parallel machines at each stage might be less than six[14, 21]. The reason of this rareness is that if the number of machines at each stage is more than five, it might be better to install a more flexible, high automated machine at the stage because of high maintenance cost incurred in many machines.

Under these assumptions Q_i/k_i is an integer equal to the number of transfer batches of job i. Furthermore, k_i/p_j is an integer equal to the number of units in one transfer batch assigned to one machine at stage j. Since all parallel and identical machines at each stage experience the equal loading and same job sequence, only one machine at each stage need to be considered for sequencing purposes. A machine at stage j processes Q_i/p_j units of job i. If t_{ij} is the processing time per unit of job i on a machine at stage j, then $t_{ij}(Q_i/p_j)$ is the total processing time of job i on a machine at stage j.

We designate a batch of k_i/p_j units as a scheduling unit for job i. At stage j this scheduling unit has a processing time of $z_{ij} = t_{ij}(k_i/p_j)$. If the assumption that the production lot size of job i is a multiple of its transfer batch size is relaxed to the mild assumption(that is, if only the assumption (P) holds without the assumption (T)), the formula of z_{ij} in equations (1) and (4) of section 3 only needs to be modified to $t_{ij}\mathrm{MOD}(Q_i/k_j)/p_j$, where $\mathrm{MOD}(Q_i/k_j)$ means the integer remainder when Q_i is divided by k_j. The number of scheduling units processed at each machine of stage j for job i is Q_i/k_i, which is independent of stage j. We treat a scheduling unit as a job in our transformed scheduling problem. There will be a total of $(\sum_{i=1}^{n} Q_i/k_i)$ jobs (scheduling units) of which n jobs are distinct. Thus, our n-job, m-stage transfer batch flowshop scheduling problem with identical parallel machines is converted to a $(\sum_{i=1}^{n} Q_i/k_i)$-job, m-stage flowshop scheduling problem with one machine at each stage, where identical jobs are processed consecutively. An optimal permutation schedule for the transformed problem is also an optimal permutation schedule for the original problem. A detailed example is given in [12] to delineate the effects of the assumption (T) and assumption (P) previously described under the strategy I. (See [12].)

Since all machines at each stage have the identical production pattern and the numbers of scheduling units processed at each machine are equal at all stages, it suffices to consider only one machine at each stage for sequencing purposes. If we find the optimal schedule for the transformed and simply duplicate the optimal machine schedule for all parallel machines at each stage, the original problem will be operating under the duplicated optimal schedule. Since an optimal permutation schedule is very close to the theoretical optimum solution for a flowshop scheduling problem[1, 3], we seek the optimal permutation schedule for the converted problem. In the following section, a branch and bound algorithm is presented to obtain the optimal permutation schedule for the transformed problem.

3. A Branch and Bound algorithm

A branch and bound (B&B) procedure for obtaining the optimal permutation schedule of the problem under consideration is proposed. An efficient B&B procedure is generally character-

ized by a branching rule, a lower bounding rule, and a search strategy. For branching purposes we designate nodes at level r of the search tree as partial sequences in which jobs in the first r positions have been fixed. In the search strategy jumptracking is used for selecting a partial sequence with the smallest lower bound, from which lower level sequences are generated. A trial complete sequence generated by a heuristic algorithm is utilized for fathoming the partial sequences as soon as possible. The heuristic algorithm is presented in subsection 3.2.

For describing the B&B algorithm, necessary notations are stated as follows:

i job index, $i = 1, 2, ..., n$.

j stage index, $j = 1, 2, ..., m$.

s_{ij} setup time of job i on stage j.

l_{ij} $s_{ij} + z_{ij}$, run-in delay time of job i on stage j.

w_{ij} $s_{ij} + (Q_i/p_j)t_{ij}$, total processing time (including setup time) of job i on one machine at stage j.

Ω the set of all jobs that are to be produced in a given time.

σ available partial schedule.

a a job being augmented to the partial sequence σ.

π the set of jobs not included in the partial schedule σ among the set Ω.

π' the set of jobs included in the partial schedule σ.

$R_j(\sigma)$ the time at which the machine at stage j is released after processing the partial sequence σ.

$S_j(\sigma a)$ the actual start time at which job a in the set π can be processed at stage j. Both job a and a machine of stage j need to be available for processing.

$S(\sigma, j)$ the earliest start time in terms of job availability, at which a job in the set π can be processed at stage j. This value is used for computing a lower bound on makespan of the partial sequence σ at stage j.

$L_j(\sigma)$ the lower bound for stage j on makespan of all schedules that begin with the partial schedule σ.

$LB(\sigma)$ the lower bound for all stages on makespan of all schedules that begin with the partial sequence σ.

3.1 A lower bound

To obtain a lower bound on a partial sequence σ for the scheduling problem for all stages, we find a lower bound for each stage, and we assign their maximum as the lower bound on the partial schedule σ.

Consider any schedule beginning with a partial sequence σ. The lower bound for stage j ($j \neq m$) is determined by equation (1).

$$L_j(\sigma) = Max\,[R_j(\sigma),\ S(\sigma, j)] + \sum_{i \in \pi} w_{ij} + \underset{i \in \pi}{Min}\left[\sum_{v=j+1}^{m} z_{iv}\right]. \qquad (1)$$

In equation (1), the first term means the earliest start time in view of machine availability and job availability respectively, at which a job in the set π can be processed at stage j. The second term is the total remaining production time of jobs in the set π. The last term is the minimum run-out time of a job among jobs in the set π, which is sum of processing times of the last scheduling unit from stage $(j+1)$ to stage m. The proof that $L_j(\sigma)$ in equation (1) can be a lower bound for the partial schedule for stage j is as follows. Let $M_j(\sigma)$ be makespan of a schedule that begins with the partial schedule σ. And let c be the last

job among the set π. The schedule will be $\sigma \cdots c$. $M_j(\sigma) \geq$ max[machine available time, job available time] + remaining processing of the unscheduled jobs + run-out time of job c from stage $(j+1)$ to stage m. Thus,

$$\begin{aligned} M_j(\sigma) &\geq Max\,[R_j(\sigma),\ S(\sigma, j)] \\ &\quad + \sum_{i\in\pi} w_{ij} + \left[\sum_{v=j+1}^{m} z_{cv}\right] \\ &\geq Max\,[R_j(\sigma),\ S(\sigma, j)] \\ &\quad + \sum_{i\in\pi} w_{ij} + \underset{i\in\pi}{Min}\left[\sum_{v=j+1}^{m} z_{iv}\right]. \end{aligned}$$

Therefore, $L_j(\sigma)$ can be a lower bound for the partial schedule for stage j. For stage m, the third term is eliminated because there is no succeeding stage after stage m. Hence, $L_m(\sigma)$ may be determined by equation (2).

$$L_m(\sigma) = Max\,[R_m(\sigma),\ S(\sigma, m)] + \sum_{i\in\pi} w_{im}. \quad (2)$$

When job a in the set π is augmented to the partial sequence σ, $R_j(\sigma)$ and $S_j(\sigma)$ for $j \geq 2$ are updated according to equations (3) and (4), where $R_j(\varnothing) = 0$.

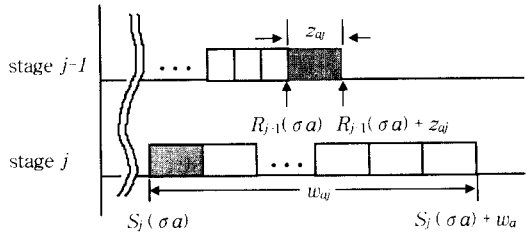
$$S_j(\sigma a) = Max\,[S_{j-1}(\sigma a) + l_{aj-1},\ R_j(\sigma)]. \quad (3)$$

$$R_j(\sigma a) = Max\,[S_j(\sigma a) + w_{aj},\ R_{j-1}(\sigma a) + z_{aj}]. \quad (4)$$

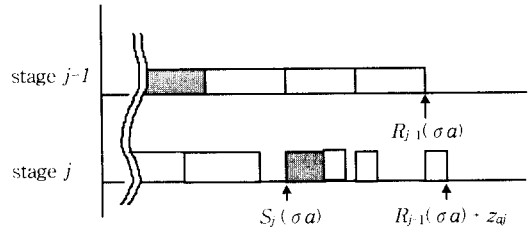
In equation (3), $[S_{j-1}(\sigma a) + l_{aj-1}]$ is selected when the machine at stage j is idle at the time of job's arrival from stage $(j-1)$. $R_j(\sigma)$ is selected when total processing time of the last job in the partial sequence σ is large at stage j. In equation (4), $[S_j(\sigma a) + w_{aj}]$ is selected when total production time of job a at stage j is large or start time of job a at stage j is delayed due to machine unavailability. This situation is displayed in [Figure 1]. $[R_{j-1}(\sigma a) + z_{aj}]$ is selected when total production time of job a at stage j is small as shown in [Figure 2]. Since there is no machine idle time at stage 1, $S_1(\sigma a)$ and $R_1(\sigma a)$ are computed according to equations (5) and (6), where $R_1(\sigma) = \sum_{i\in\sigma} w_{i1}$.

$$S_1(\sigma a) = R_1(\sigma) \quad (5)$$

$$R_1(\sigma a) = R_1(\sigma) + w_{a1} \quad (6)$$



[Figure 1] $S_j(\sigma a) + w_{aj} \geq R_{j-1}(\sigma a) + z_{aj}$



[Figure 2] $S_j(\sigma a) + w_{aj} < R_{j-1}(\sigma a) + z_{aj}$

In view of job availability, the earliest start time at which job a in the set π can be processed at stage j for $j \geq 2$, $S(\sigma, j)$, is computed as equation (7). Since all jobs are available for processing at time zero at stage 1, the earliest start time at which job a in the set π can be processed, $S(\sigma, 1)$, is zero.

$$S(\sigma, j) = \underset{h=1,\ldots,j-1}{Max}\ [R_h(\sigma) + \underset{i\in\pi}{Min}\left[\sum_{v=h}^{j-1} l_{iv}\right]],\ j \geq 2. \quad (7)$$

Therefore, for all stages, the lower bound on makespan of all schedules that begin with the partial sequence σ is determined as equation (8).

$$LB(\sigma) = \underset{j=1,2,\ldots,m}{Max}\ |\ L_j(\sigma)\ | \quad (8)$$

3.2 A heuristic algorithm

A heuristic algorithm is developed to find a trial complete sequence, which is utilized to fathom the partial sequences as soon as possible, whose lower bounds are greater than or equal to the makespan of the heuristic sequence. The heuristic algorithm selects the next job in the sequence that minimizes the idle time at stage m. In steps of the algorithm, π/a means set exclusion, i.e., the element a is eliminated from the set π. The algorithmic form of the heuristic is as follows:

- Step 1 (Determining the first job in the sequence)
  Let Ω be the set of all jobs.
  Find the set of jobs such that $w_{i1} \leq w_{i2} \leq \ldots \leq w_{im}$.
  Let the set be τ.
  if τ is not empty, *then* $\phi := \tau$
  else
  $$\phi := \Omega$$
  endif
  Choose job a such that
  $$R_m(a) - w_{am} = \underset{i \in \phi}{Min} [R_m(i) - w_{im}].$$
  Any tie can be resolved by choosing the job with the largest value of w_{im}.
- Step 2 (Initialization of the partial sequence σ and the set π)
  $$\sigma := a, \quad \pi := \Omega/a$$
- Step 3 (Determining the next job in the sequence)
  Choose job b such that
  $$R_m(\sigma b) - R_m(\sigma) - w_{bm} = \underset{i \in \pi}{Min} [R_m(\sigma i) - R_m(\sigma) - w_{im}]$$
  Any tie can be resolved by choosing the job with the largest value of w_{im}.
- Step 4 (Updating the partial sequence σ and the set π, and stopping rule)
  $$\sigma := \sigma b, \quad \pi := \pi/b$$
  if π is empty, *then* stop
  else goto Step 3
  endif

Final sequence σ is a trial complete sequence generated by the heuristic algorithm.

3.3 A branch and bound algorithm

As a B&B procedure is known to be most efficient method to find optimal permutation schedule for flowshop scheduling, its algorithm has been suggested for the problem. In the algorithm steps, the active list includes the set of partial sequences that have not been fathomed until that time. The B&B algorithm is stated as follows:

- Step 0 (Initialization)
  0-1 : Find a trial complete sequence using the heuristic and calculate its makespan.
  0-2 : Let MINLB be makespan of the trial complete sequence.
  0-3 : Place the trial complete sequence on the active list.
- Step 1 (Calculating a lower bound of the partial sequence that consists of one job)
  1-1 : Generate partial sequences that consist of one job.
  1-2 : Sort partial sequences in ascending order.
  1-3 : Calculate a lower bound of all sorted partial sequences.
- Step 2 (Selecting a branching sequence)
  2-1 : Choose the sequence with the smallest lower bound on the active list.

2-2 : Any tie can be resolved by selecting the sequence with the largest number of job included in the sequence

2-3 : Further ties can be resolved by selecting the first sequence in ascending order.

2-4 : Let the selected sequence be B.

- Step 3 (Stopping rule)

if the number of jobs included in B is $n-1$, then

goto Step 5

else

endif

- Step 4 (Branching the selected sequence and calculating the lower bound)

4-1 : $\pi := \Omega/B$

4-2 : For all $i \in \pi$

4-2-1 : Select the job in ascending order from the set π, which is augmented to the partial sequence B.

4-2-2 : Let a be the selected job.

4-2-3 : Calculate $R_j(Ba)$.

4-2-4 : $\sigma := Ba$

4-2-4 : Calculate $LB(\sigma)$.

4-2-5 :

if $LB(\sigma) \geq$ MINLB, *then* fathom this sequence σ.

else place this sequence σ on the active list.

endif

4-3 : goto Step 2

- Step 5 (Updating MINLB)

MINLB := *Min* [LB(B), MINLB]

- Step 6 (Fathoming)

For all sequences on the active list, fathom the sequences whose lower bounds are greater than or equal to MINLB and remove the fathomed sequence from the active list.

- Step 7 (Stopping rule)

if there is only one sequence on the active list,

then

stop

else

goto Step 2

endif

4. A Numerical Illustration

Consider a three-stage flexible flowshop with three machines at stage 1, two machines at stage 2 and four machines at stage 3. Four jobs are to be scheduled. Their processing times at three stages, production lot sizes of jobs, and their transfer batch sizes are shown in <Table 2>. Setup times of jobs at three stages are given in <Table 3>. We have $m = 3$, $p_1 = 3$, $p_2 = 2$, and $p_3=4$. The objective is to find the optimal schedule that minimizes makespan.

<Table 2> Unit processing times(min/unit), transfer batch sizes, and production lot sizes(units).

Job	Stage 1	Stage 2	Stage 3	Q_i	k_i
Job 1	3	2	4	120	24
Job 2	1	2	6	72	12
Job 3	4	2	3	48	12
Job 4	2	3	4	108	36

<Table 3> Setup times(min/setup)

Job	Stage 1	Stage 2	Stage 3
Job 1	10	10	10
Job 2	10	15	20
Job 3	20	15	10
Job 4	15	20	15

4.1 A trial complete sequence generated by a heuristic algorithm

Using the heuristic algorithm described in subsection 3.2, a trial complete sequence is generated as shown in <Table 4>. Since the set ψ such

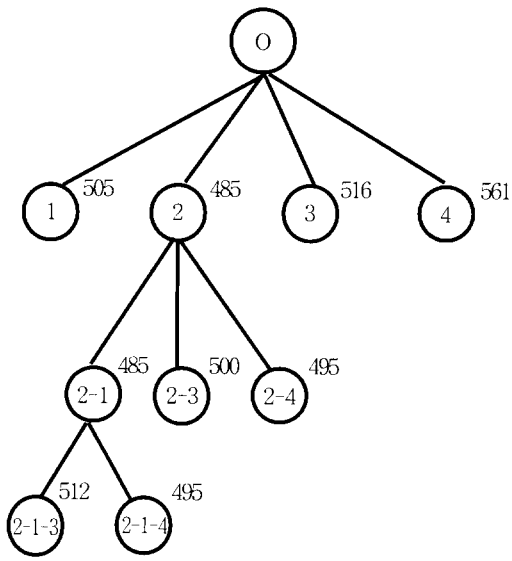
that $w_{i1} \leq w_{i2} \leq w_{i3}$ is {1,2} and $w_{13} > w_{23}$, the first position job in the sequence is job 1. Since $(R_3(1\text{-}2) - R_3(1) - w_{23})$ and $(R_3(1\text{-}3) - R_3(1) - w_{33})$ have the same smallest values but $w_{23} > w_{33}$, the second position job in the sequence is job 2. Since $(R_3(1\text{-}2\text{-}3) - R_3(1\text{-}2) - w_{33})$ is smaller among two jobs, the third position job in the sequence is job 3. Therefore a trial complete sequence is 1-2-3-4 and its makespan is 532.

<Table 4> An illustration of the heuristic algorithm.

Sequence	$R_m(\sigma i) - R_m(\sigma) - w_{im}$	w_{im}	Description
1-2	0	128	selected
1-3	0	46	
1 4	66	123	
1-2-3	0	46	selected
1-2-4	20	123	

4.2 A branch and bound algorithm

The B&B tree obtained in solving a given numerical example is shown in [Figure 3]. The nodes represent partial sequences. The numerical value right above the node represents a lower bound of the partial sequence. The list of partial sequences generated is shown in <Table 5>. The trial complete sequence generated by a heuristic algorithm, 1-2-3-4, is placed on the top of the list. As shown in <Table 5>, partial sequence 4 is fathomed by a trial sequence 1-2-3-4. Since the partial sequence 2 has the lowest lower bound, three partial sequences are generated from this sequence. In a similar way, two partial sequences are generated from partial sequence 2-1. Since the partial sequence 2-1-4 (that is, 2-1-4-3) has the lowest lower bound and is a complete sequence, all sequences including a trial complete sequence 2-3-1-4 are fathomed by the sequence 2-1-4 which is optimal sequence. If other optimal sequences are to be found, two partial sequences must be generated from partial sequence 2-4. Actually, partial sequence 4 is not included in the active list, because when generated they are fathomed by the trial complete sequence 2-1-3-4.



[Figure 3] The Branch and Bound Tree

<Table 5> List of partial sequences

Partial sequence, σ	$L_1(\sigma)$	$L_2(\sigma)$	$L_3(\sigma)$	$LB(\sigma)$	Fathoming & Branching
1-2 3-4	-	-	-	532	fathomed by 2-1 4
0	-		-	-	branched from
1	356	505	495	505	fathomed by 2 1 4
2	356	485	468	485	branched from
3	365	516	507	516	fathomed by 2 1-4
4	356	510	561	561	fathomed by 1-2-3 4
2-1	356	485	468	485	branched from
2-3	383	500	468	500	fathomed by 2-1-4
2-4	356	485	495	495	fathomed by 2-1-4
2 1 3	425	512	491	512	fathomed by 2-1-4
2-1-4	356	475	495	495	optimal sequence

The computation of lower bounds for partial sequences on the optimal path is as follows:

partial sequence 2

$\sigma = 2, \ \pi = \{1, 3, 4\}$

$R_1(2) = 34, \ R_2(2) = 101, \ R_3(2) = 169$

$S(2, 1) = 0$

$L_1(2) = Max\,[R_1(2),\ S(2,\ 1)]$

$$+ \sum_{i\in\pi} w_{i1} + \underset{i\in\pi}{Min}\left[\ \sum_{v=2}^{3} z_{iv}\right] = 356$$

$S(2,\ 2) = Max\,[R_1(2) + \underset{i\in\pi}{Min}\ \lfloor\ l_{i,1}\rfloor\ ] = 68$

$L_2(2) = Max\,[R_2(2),\ S(2,\ 2)]$

$$+ \sum_{i\in\pi} w_{i2} + \underset{i\in\pi}{Min}\left[\ \sum_{v=3}^{3} z_{iv}\right] = 485$$

$S(2,\ 3) = Max\,[R_1(2) + \underset{i\in\pi}{Min}\ \lfloor\ l_{i,1} + l_{i,2}\ \rfloor\ ,$

$R_2(2) + \underset{i\in\pi}{Min}\ \lfloor\ l_{i,2}\ \rfloor\ ] = 128$

$L_3(2) = Max\ [R_3(2),\ S(2,\ 3)] + \sum_{i\in\pi} w_{i3} = 468$

$LB(2) = \underset{j=1,2,.3}{Max}\ \lfloor\ L_j(2)\ \rfloor = 485$

partial sequence 2-1

$\sigma = 2\text{-}1,\ \pi = \{3,\ 4\}$

$R_1(2\text{-}1) = 164,\ R_2(2\text{-}1) = 231,\ R_3(2\text{-}1) = 299$

$S(2\text{-}1,\ 1) = 0$

$L_1(2\text{-}1) = max\,[R_1(2\text{-}1),\ S(2\text{-}1,\ 1)]$

$$+ \sum_{i\in\pi} w_{i1} + \underset{i\in\pi}{Min}\left[\ \sum_{v=2}^{3} z_{iv}\right] = 356$$

$S(2\text{-}1,\ 2) = max\,[R_1(2\text{-}1) + \underset{i\in\pi}{Min}\ \lfloor\ l_{i,1}\ \rfloor\ ] = 200$

$L_2(2\text{-}1) = max\,[R_2(2\text{-}1),\ S(2\text{-}1,\ 2)]$

$$+ \sum_{i\in\pi} w_{i2} + \underset{i\in\pi}{Min}\left[\ \sum_{v=3}^{3} z_{iv}\right] = 485$$

$S(2\text{-}1,\ 3) = max\,[R_1(2\text{-}1) + \underset{i\in\pi}{Min}\ \lfloor\ l_{i,1} + l_{i,2}\ \rfloor\ ,$

$R_2(2\text{-}1) + \underset{i\in\pi}{Min}\ \lfloor\ l_{i,2}\ \rfloor\ ] = 258$

$L_3(2\text{-}1) = max\,[R_3(2\text{-}1),\ S(2\text{-}1,\ 3)] + \sum_{i\in\pi} w_{i3} = 468$

$LB(2\text{-}1) = \underset{j=1.2,.3}{Max}\ \lfloor\ L_j(2-1)\ \rfloor = 485$

partial sequence 2-1-4

$\sigma = 2\text{-}1\text{-}4,\ \pi = \{3\}$

$R_1(2\text{-}1\text{-}4) = 251,\ R_2(2\text{-}1\text{-}4) = 403,\ R_3(2\text{-}1\text{-}4) = 449$

$S(2\text{-}1\text{-}4,\ 1) = 0$

$L_1(2\text{-}1\text{-}4) = 356$

$S(2\text{-}1\text{-}4,\ 2) = 287$

$L_2(2\text{-}1\text{-}4) = 475$

$S(2\text{-}1\text{-}4,\ 3) = 430$

$L_3(2\text{-}1\text{-}4) = 495$

$LB(2\text{-}1\text{-}4) = 495$

5. Computational Results

The effectiveness of the proposed Branch and Bound algorithm has been investigated by solving 640 problems. The algorithm was programmed in C, and a SUN Enterprise 4000 was used to investigate the performance of the algorithm. For each problem, the number of stages ranged from 3 to 6 inclusive, and the number of jobs is from 4 to 11 inclusive. The number of problem cases is equal to 32(8×4). Problem sets were randomly generated as follows:

1) The number of machines at stage j, p_j, were randomly generated from integer numbers 2 to 5 inclusive.
2) The transfer batch size of job i, k_i, were generated from (the least common multiple of p_js) × random numbers from integers 2 to 5 inclusive.
3) The production lot size Q_i were generated from k_i × random numbers from integer 2 to 5 inclusive.
4) The processing times, t_{ij} were randomly generated from integers 1 to 10 inclusive.
5) The setup times, s_{ij} were randomly generated from integers 5 to 20 inclusive.

Twenty problems were created in each case, and average values over 20 problems in each case were shown in each cell of <Table 6>, <Table 7>, and <Table 8>. Therefore, the total number of problems solved is equal to 640. In the problem sets of eleven jobs, the algorithm was stopped

if computing time was more than thirty minutes or error message 'core dumped' occurred due to memory shortage. This occurred very frequently, e.g., in 70 per cent of eleven job problems.

〈Table 6〉 The number of nodes generated by the Branch and Bound algorithm.

No. of stages	No. of jobs in problem set							
	4	5	6	7	8	9	10	11
3	10.40	19.25	54.90	126.35	503.70	150.15	3,376.50	69.55
4	12.80	25.55	27.20	145.50	293.60	1,422.65	12,180.75	113.20
5	11.95	31.85	67.10	195.75	2,583.15	2,923.85	2404.95	371.30
6	12.85	25.35	65.15	153.30	2,170.05	1,806.65	753.30	5,292.95
Average	12.00	25.50	53.59	155.23	1,387.63	1,575.83	4,678.88	1,461.75
Min. nodes possible	9	14	20	27	35	44	54	65
Max. nodes possible	24	120	720	5,040	40,320	362,880	3,628,800	39,916,800

<Table 6> shows average number of nodes generated by the algorithm, minimum number of nodes possible, and maximum number of nodes possible. The average values were rounded off the fractions to two decimal places. From <Table 6>, it is evident that as the number of jobs increases, the maximum possible number of nodes generated increases faster than the average number of nodes created by the branch and bound algorithm.

<Table 7> shows the computing time which does not include problem generation time. The average values in all cells were rounded off the fractions to four decimal places. From <Table 7>, it is found that as the number of jobs increases, computing time increases sharply. It is also evident that as the number of stages increases, computing time in most of the problem cases increases. The reason why this occurs is that in order to save memory storage, the release time, $R_j(\sigma)$, of a parent node are not stored. Instead, the release time of a parent node is calculated each time the release time of the node is calculated.

〈Table 7〉 Computing times(sec)

No. of stages	No. of jobs in problem set							
	4	5	6	7	8	9	10	11
3	0.0020	0.0050	0.0125	0.0215	0.2215	0.0130	13.5945	0.0245
4	0.0050	0.0080	0.0125	0.0810	0.0970	2.4120	73.9880	0.1865
5	0.0040	0.0175	0.0475	0.1985	10.9045	16.8870	25.8925	3.7725
6	0.0080	0.0240	0.1205	0.7015	24.0670	29.6695	16.1820	242.0765
Average	0.0048	0.0136	0.0483	0.2506	8.8225	12.2454	32.4143	61.5150

<Table 8> shows average heuristic performance ratio, in which average values were rounded off the fractions to two decimal places. The value of heuristic performance ratio means the percent ratio of the difference between the makespan of the heuristic and optimal makespan to optimal makespan. <Table 8> shows that average heuristic performance ratio over 32 cases ranges from 4.13 to 12.82. The total average over 640 problems is 8.93. The computing time of the heuristic ranges from 0.0001 second(3 stages, 4 jobs) to 1.4500 seconds(11 stages, 6 jobs). Hence the suggested heuristic may be used for a large job problem.

〈Table 8〉 Heuristic performance ratio(%)

No. of stages	No. of jobs in problem set							
	4	5	6	7	8	9	10	11
3	6.61	7.13	6.20	7.46	4.13	5.35	5.78	4.33
4	8.93	8.19	7.30	6.71	12.01	9.56	8.66	4.97
5	7.42	12.62	11.82	8.44	11.22	11.51	10.93	5.77
6	12.82	12.25	11.98	10.50	10.07	11.41	11.22	12.55
Average	8.94	10.05	9.33	8.28	9.36	9.46	9.15	6.91

6. Summary and Discussion

Flexible flowshops with parallel machines at each stage are frequently encountered in manufacturing environments. Since the parallel machines at the stage usually have similar functions, they may be assumed to be identical. Therefore our methodology can be applied to some manu-

facturing environments.

We presented a branch and bound algorithm to find the optimal permutation schedule which minimizes makespan in a flexible flowshop where transfers are performed in transfer batches. By making mild assumptions about the size of transfer batches and production lot sizes, the original transfer batch scheduling problem for a flexible flowshop was converted to a tractable flowshop scheduling problem under the identical production pattern(strategy I), which allocates equal amount of a job's workload to every machine at each stage. These assumptions reduced the computational complexity of the original problem, thereby allowing us to solve the problem with reasonable computational efforts. A numerical example is presented to demonstrate the steps of the algorithm.

The effectiveness of the proposed Branch and Bound algorithm has been investigated by solving 640 problems. Computational result indicates that within a reasonable computing time the proposed algorithm can be used for solving transfer batch scheduling problem in a flexible flowshop with the number jobs less than 12.

The introduction of transfer batches in a flexible flowshop raises some of questions requiring further research. One of the issues may be the study of another approach different from the strategy of the identical production pattern. Another issue of importance is the examination of scheduling problem which relaxes the assumptions about the size of transfer batches and production lot sizes of jobs, in which the identical production patterns of identical machines at each stage can not be obtained and hence the scheduling problem will be more complex.

REFERENCES

[1] Baker, K.R., *Introduction to sequencing and scheduling*, John Wiley & Sons, Inc., USA, pp. 55-64, 1974.

[2] Baker, K.R., *Elements of Sequencing and Scheduling*, pp.8.14-8.18, 1995.

[3] Baker, K.R., "Lot streaming in the two-machine flow shop with setup times", *Annals of Operations Research*, Vol.57(1995), pp.1-11.

[4] Baker, K.R. and D. Jia, "A comparative study of lot streaming procedures", *Omega*, Vol. 21(1993), pp.561-566.

[5] Baker, K.R. and D.F. Pyke, "Solution Procedures for the Lot-Streaming Problem", *Decision Sciences*, Vol.21(1990), pp.475-491.

[6] Blazewicz, J., H. Eiselt, G. Finke, G. Laporte and J. Weglarz, "Scheduling tasks and vehicles in a flexible manufacturing systems", *International Journal of Flexible Manufacturing Systems*, Vol.4(1991), pp.5-16.

[7] Cetinkaya, F.C. and M.S. Kayaligil, "Unit sized transfer batch scheduling with setup times", *Computers and Industrial Engineering*, Vol.22(1992), pp.177-183.

[8] Guinet, A.G.P. and M.M. Solomon, "Scheduling hybrid flowshops to minimize maximum tardiness or maximum completion time", *International Journal of Production Research*, Vol.34(1996), pp.1643-1654.

[9] Guinet, A., M.M. Solomon, P.K. Kedia and A. Dussauchoy, "A computational study of heuristics for two-stage flexible flowshop", *International Journal of Production Research*, Vol.34(1996), pp.1399-1415.

[10] Johnson, S.M., "Optimal two- and three-stage production schedules with setup times in-

cluded", *Naval Research Logistics Quarterly*, Vol.1(1954), pp.61-68.

[11] Kim, Joong-Soon, *A study on scheduling problems for flexible flow lines*, Ph. D. Dissertation in Seoul National University, KOREA, 1993.

[12] Kim, J-S, S-H Kang and SM Lee, "Transfer batch scheduling for a two-stage flowshop with identical parallel machines at each stage", *Omega*, Vol.25(1997), pp.547-555.

[13] Potts, C.N. and K.R. Baker, "Flowshop scheduling with lot streaming", *Operations Research Letters*, Vol.8(1989), pp.297-303.

[14] Rajendran, C. and D. Chaudhuri, "Scheduling in an n-job, m-stage flowshop with parallel processors to minimize makespan", *International Journal of Production Economics*, Vol. 27(1992), pp.137-143.

[15] Rajendran, C. and D. Chaudhuri, "A multi-stage parallel processors flowshop problem with minimum flowtime", *European Journal of Operational Research*, Vol.57(1992), pp. 111-122.

[16] Sriskandarajah, C. and P. Ladet, "Some no-wait shops scheduling problems: complexity concept", *European Journal of Operational Research*, Vol.24(1986), pp.424-438.

[17] Sriskandarajah, C., and S.P. Sethi, "Scheduling algorithms for flexible flowshops: worst and average case performance", *European Journal of Operational Research*, Vol.43(1989), pp.143-160.

[18] Trietsch, D. and K. R. Baker, "Basic techniques for lot streaming", *Operations Research*, Vol.41(1993), pp.1065-1076.

[19] Vickson, R.G., and B.E. Alfredsson, "Two- and three-machine flow shop scheduling problems with equal sized transfer batches", *International Journal of Production Research*, Vol.30(1992), pp.1551-1574.

[20] Wittrock, R.J., "Scheduling algorithms for flexible flow lines", *IBM Journal of Research and Development*, Vol.29(1985), pp.401-412.

[21] Wittrock, R.J., "An adaptable scheduling algorithm for flexible flowlines", *Operations Research*, Vol.36(1988), pp.445-453.