International Journal of Management Science
Vol. 1, No. 1, December 1995

Selection of Assembly Sequences Based on Flexible Assembly Systems Performance

Bongju Jeong

Abstract

In planning an assembly system, choosing the proper assembly sequence is one of the most important decisions because it significantly affects the costs associated with the assembly process. This paper deals with the selection of assembly sequences in flexible assembly systems. The selection criterion is the minimization of makespan to complete all assembly products. This problem is formulated as a "modified FAS scheduling problem" (MFASSP) and its scheduling procedure is described. The experimental results show that the proccdure is very efficient for both quality of solution and computation time.

1. INTRODUCTION

The flexible assembly system (FAS) has been gaining increasing interest eversince the flexibility concept was intruced in manufacturing systems. Even though there are various definitions of FAS, Donath, et. al.[7] has that defined it clearly as a collection of flexible workplaces that may perform any subset or all of the required operations on a variety of parts. In the flexible assembly system environment, the choice of assembly sequence becomes the key factor to the efficiency of the system, in which multiple assembly sequences are allowed to produce multiple products with many different parts. In other words, whenever production plans are changed by the introduction of new products or different parts, new assembly sequences may be

required to maximize the system performance.

Most previous researches on the evaluation and selection of assembly sequences have been based on the performance of assembly activities, thus the chosen sequence may need easier and simpler operation than any other sequences ([3], [4], [5], [9], [12], [15], [16], [19], [22]). However, the assembly sequence affects not only the performance of the assembly activities themselves but also the efficency of the whole assembly system. The objective of this research is to evaluate and select the most efficient assembly sequence in the flexible assembly system (FAS) environment. The evaluation criterion is the makespan to finish all products. Therefore, the scheduling problem in the FAS environment is solved, which is newly modelled as a modified flexible assembly systems scheduling problem (MFASSP).

Donath, et al.[7] developed a mathematical model for the flexible assembly systems scheduling problem (FASSP) in which the objective function is to minimize the makespan for completing all products under the FAS environment. Donath showed that the FASSP is NP-complete; thus, it is unlikely that there is a polynomial time optimal algorithm for solving it. Since subproblems of MFASSP are "parallel processor scheduling problems," we need to develop new algorithms for this type of problems. Given n single-operation jobs and m processors on which any set of jobs can be processed, the problem is to schedule n jobs on m processors in parallel with the objective of minimizing the makespan. This problem can be classified into four types of problem according to types of job and processor as shown in table 1. All these types of problem are known to be NP-complete. No polynomial time bounded algorithm for optimal solutions has been developed for general cases of any of the four types of problem. much recent research has been established for type Ⅰ, Ⅱ, and Ⅲ problems ([18], [2], [6], [23], [11], [20], [21], [10], [13], [8]), but very little is known about Type Ⅳ problem([17], [1]). If we have a product and one assembly sequence in the MFASSP, the MFASSP becomes a Type Ⅳ problem which is more general than any otherrrr problem.

Table 1. Types of scheduling Problems with Parallel Processors

		Jobs	
		Independent	Dependent
Processor	Identical	Type I	Type II
	Non-identical	Type III	Type IV

2. Modified FAS Scheduling Problem: MFASSP

A new way to find the best assembly sequence in flexible assembly systems is the use of the Modified Flexible Assembly Systems Scheduling Problem (MFASSP). It can be described as follows:

Suppose there are a number of products to be assembled in FAS and each product has various alternatives for the assembly operation sequence. The FAS consists of a number of flexible workstations that can perform all subassembly operation types. Each workstation may have a different performance. The execution of a single subassembly operation can occur at any workstation during a given period of time, but not at more than one workstation simultaneously. The problem is to schedule all subassembly operations according to one assembly operation sequence for each product, while minimizing the makespan to complete all products.

From the problem description of the MFASSP, the mathematical model is formulated as follows:

Given P : the set of all products

I : the set of all subassembly operations

J_{lp} : the lth subassembly operation sequence of product p

N_p : the number of subassembly operation sequence of product p

K : the set of all workstations

m : the number of workstations, i. e., $1 \le k \le m$, for any $k \in K$

t_{ik} : the duration of subassembly operation i at workplace k

s_{ik} : the starting time of subassembly operation i at workplace k

d : a displacement time such $d > \max t_{ik}$ for all $i \in I$ and for all $k \in K$.

Minimize Maximum $(s_{ik}+t_{ik})$ for all $i \in I$, $k \in K$

subject to:

(Constraint Set 1) Each workstation k can process no more than one subassembly operation at a time. For all workstations $k \in K$ and all task combinations involving workstation k, the following must be satisfied for $i, j \in I$:

$$s_{ik} + t_{ik} \leq s_{jk}$$

or

$$s_{ik} \geq s_{jk} + t_{jk}$$

or

$$s_{ik} + s_{jk} = 0$$

(Constraint Set 2) Each subassembly operation should meet technological sequence requirements. For all subassembly operation combinations $i, j \in J_{lp}$, where the subassembly operation i must precede j for $k, n \in k$ and $n \neq k$, the following should be satisfied:

$$s_{ik} + t_{ik} \leq s_{jn}$$

or

$$s_{ik} = 0$$

or

$$s_{jn} = 0$$

or

$$s_{ik} + t_{jn} = 0$$

(Constraint Set 3) For each product, exactly one of its subassembly operation sequence should be performed. For all products $p \in P$, the following should be satisfied:

$$\sum \sum s_{ik} > 0 \text{ and } \sum_{i \in J_{l,p}} \sum_{k \in K} \sum s_{ik} = 0 \text{ for all } l \text{ such that } l \neq 1$$

or

$$\sum_{i \in J_{l,p}} \sum_{k \in K} s_{ik} > 0 \text{ and } \sum_{i \in J_{l,p}} \sum_{k \in K} s_{ik} = 0 \text{ for all } l \text{ such that } l \neq 2$$

or

.....

or

$$\sum_{i \in J_{Np,p}} \sum_{k \in K} s_{ik} > 0 \text{ and } \sum_{i \in J_{l,p}} \sum_{k \in K} s_{ik} = 0 \text{ for all } l \text{ such that } l \neq N_p$$

(Constraint Set 4) All subassembly operations in the chosen assembly sequence for each product should be scheduled while satisfying the condition that each operation must be assigned to only one workstation. Also the starting time offest of d is needed to avoid the inclusion of t_{ik} in the objective function, even though the associate s_{ik}=0 for unscheduled operations. Thus, the following set should be satisfied for all subassembly operatons $i \in J_{lp}$, where l is the chosen sequence for a product $p \in P$.

$$s_{i,1} > d \text{ and } s_{ik} = 0 \text{ for all } k \in \mathrm{K} \text{ such that } k \neq 1$$

or

$$s_{i,2} > d \text{ and } s_{ik} = 0 \text{ for all } k \in \mathrm{K} \text{ such that } k \neq 2$$

or

.....

or

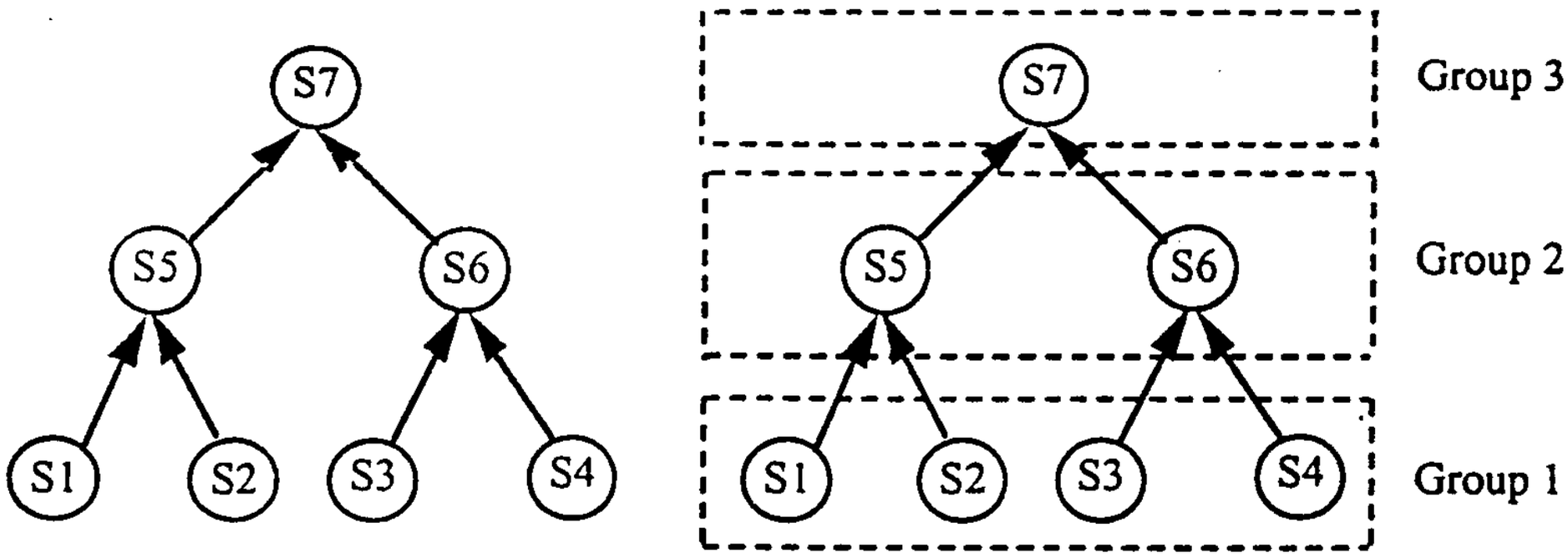
$$s_{i,m} > d \text{ and } s_{ik} = 0 \text{ for all } k \in \mathrm{K} \text{ such that } k \neq m$$

Since the MFASSP is NP-complete, the MFASSP is not likely to be solved by polynomial time algorithm[14]. This is why a heuristic procedure is needed to solve the MFASSP.

3. SCHEDULING PROCESS FOR MFASSP

3.1. Scheduling by Decomposition

Suppose that there is only one product and one assembly sequence in the MFASSP. The problem is to schedule all subassembly operations according to an assembly operation sequence, while minimizing the makespan in the FAS. Consider an subassembly operation tree as shown in Figure 1, where a node corresponds to a subassembly operation and the directed arcs represents the precedence between two nodes. The tree can be decomposed into several groups so that within each group, subassembly operations are independent of each other. If there is no precedence relation among assembly operations, they are independent of each other.



(a) Subassembly Operation Tree (b) 3 Groups Generated by Decomposition

Figure 1 Tree Decomposition

Scheduling within a Group

Since the subassembly operations within a group are independent of each other, the scheduling problem is stated as follows:

Problem(P): Suppose there are n independent subassembly operations and m workstations with different performances (i. e, non-identical, unrelated workstations). Each workstation can perform any subset of subassembly operations. Each subassembly operation can be processed by at most one workstation at a time and preemption is not allowed. The problem is to find the schedule that minimizes makespan.

Solving the problem (P) is the basic process for solving the MFASSP because the MFASSP can be solved by the successive solving of (P) group by group. Since the problem (P) is the scheduling problem with "parallel non-identical machines and independent jobs," a type III problem, it is also NP-complete. Consider following two conditions;

(1) Condition for the ready time of operations (o_i)

Each operation has a ready time at which the operation is available for processing. For an operation i, either $o_i = 0$ or $o_i > 0$.

(2) Condition for the ready time of workstations (w_j):

Each workstation has the ready time at which the operation is available for processing. For an workstation j, either $w_j = 0$ or $w_j > 0$.

(For convenience, i is used as a subscript for workstations instead of k which is used in the MFASSP model.)

Suppose there is an assembly sequence for each product, and we are trying to schedule all operations by product, sequentially as shown in Figure 2. Each assembly operation tree is decomposed into groups. Then the scheduling problem (P) within a group is classified into three subproblems according to the above two conditions as follows;

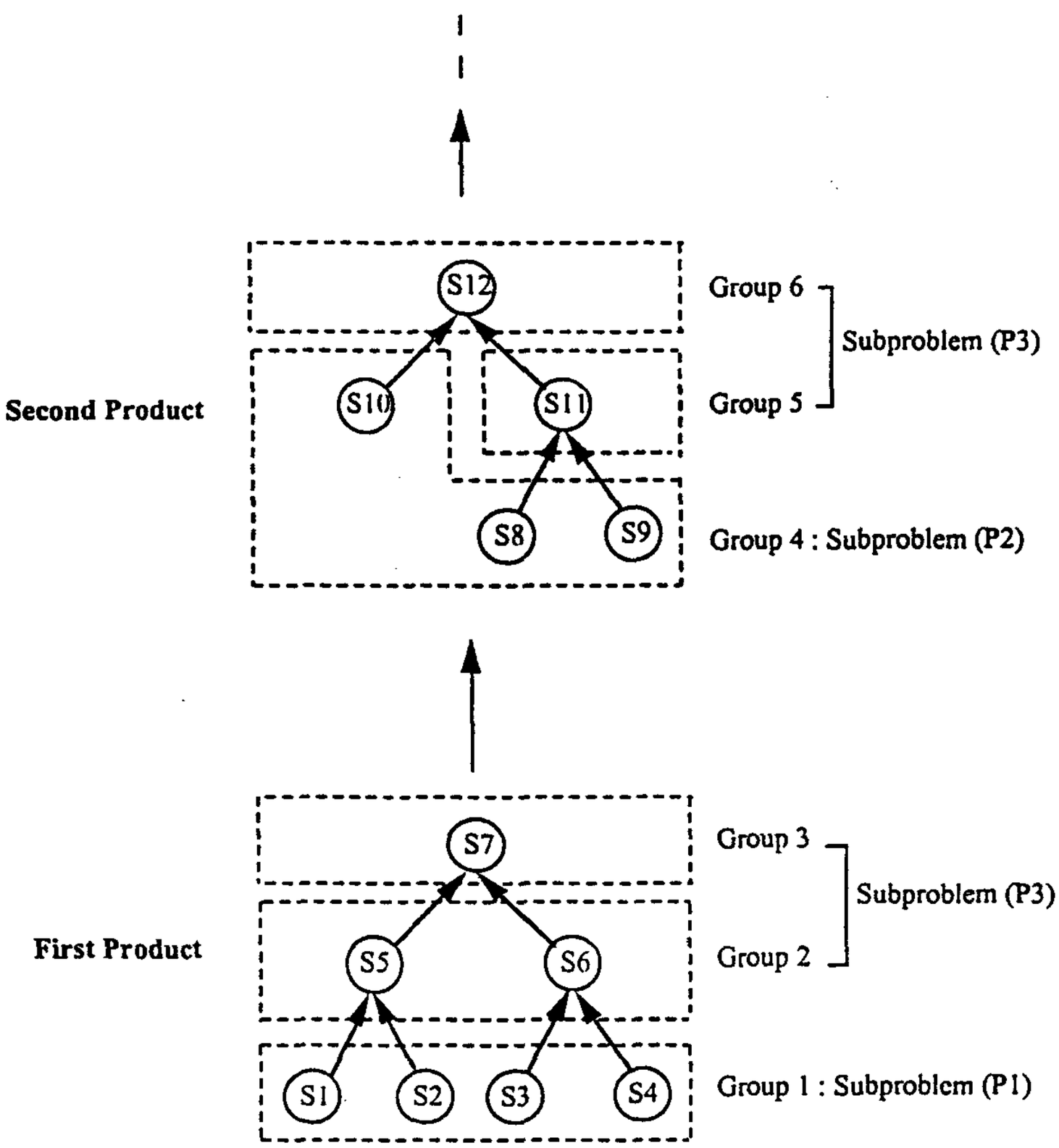


Figure 2. The Subproblems Corresponding to Each group

Subproblem (P1): Problem (P) with o_i=0 and w_j=0 for all i, j.
All operations and workstations have the same ready times of zero. This problem corresponds to the first group of assembly sequences for the first product as shown in Figure 2.

Subproblem (P2): Problem (P) with o_i=0 and $w_j \geq 0$ for all i, j.
All operations have the same ready times of zero. Workstations may have different ready times. Thus this problem is more general than problem (P1). This problem corresponds to the first group of assembly sequences for the second and subsequent products.

Subproblem (P3): Problem (P) with $o_i \geq 0$ and $w_j \geq 0$ for all i, j.
All operations and workstations may have different ready times. Thus this problem is the most general case of Type III problems. This problem corresponds to the second and subsequent groups of assembly sequences for all products.

To solve the MFASSP, three types of subproblem should be solved. The objective function of all subproblems is the minimization of makespan. However, note that each group except the last group of the last product affect the subsequent groups due to their completion time. Even if the alternative schedules have the same makespan, a schedule with a smaller sum of makespans (SOM) is preferred because its subsequent group can start earlier at some workstations. Thus, the objective of our subproblems is not only to minimize the makespan but also to minimize SOM if there are alternative schedules with the same makespan.

3. 2. The Scheduling Procedure

As described in the previous section, the subassembly operations of an assembly sequence can be scheduled by decomposing the corresponding subassembly operation tree into groups and scheduling each group successively. Similarly, by extending this idea, the MFASSP is solved. To solve the MFASSP, three types of subproblems should be solved. For each subproblem, its own algorithm is needed. Let Algorithm (A1), (A2), and

(A3) be the algorithm to solve the subproblem (P1), (P2), and (P3), respectively. The scheduling procedure for the MFASSP is as follows:

Scheduling Procedure for the MFASSP

Step 1. Let S be the null schedule and P be the set of products to be scheduled.

Step 2. Choose a product p from P arbitrarily.
Let J_p be the set of subassembly operation trees of the product p.
Let T be a large enough integer.

Step 3. Choose a subassembly operation tree j from J_p and decompose it.
If the product p is the first chosen product, then go to step 4.
otherwise, go to step 5.

Step 4. Given the schedule S, schedule the subassembly operations within the first group by algorithm (A1), and within each subsequent group by algorithm (A3).
Let S_j be a new schedule in which j is added to S, and T_j be the makespan of S_j.
Go to step 6.

Step 5. Given the schedule S, schedule the subassembly operations within the first group by algorithm (A2), and within each subsequent group by algorithm (A3). Let S_j be the new schedule in which j is added to S, and T_j be the makespan of S_j.

Step 6. If $T<T_j$, then $T\leftarrow T_j$. Otherwise $J_p\leftarrow J_p-j$.
If $J_p=\phi$, $S\leftarrow S_j$ and go to step 7.
Otherwise, go to step 3.

Step 7. $P\leftarrow P-p$.
If P$=\phi$, then stop.
Otherwise, go to step 2.

The process is based on "product by product" scheduling. Beginning with the first product, the best assembly sequence for each product is scheduled one by one. The best assembly sequence for the first product is chosen after all of its alternative assembly sequences are evaluated independently. The best assembly sequence is an assembly sequence with a minimal makespan.

Given the schedule of the previous product, the best assembly sequence of next products is chosen in the same way. Each alternative assembly sequence is scheduled by decomposing its subassembly operation tree into groups and scheduling each group successively.

4. HEURISTIC ALGORITHMS FOR SUBPROBLEMS OF MFASSP

4. 1. Background of Heuristics

Unlike the general type III problem, the subproblems of MFASSP have the following properties:

(1) The subproblems minimize the makespan and also minimize SOM (sum of makespans) under the same makespan.

(2) The subproblems have the general constraints for ready times of workstation and operation. These constraints make the problem more difficult. The improvement procedures consider only the Type III problem with no constraints for ready times.

In order to solve the subproblems efficiently, the scheduling process of new heuristics is done by updating the Assignment Table as shown in Figure 3.

o_i \ j	w_1	w_2	.....	w_m
o_1	$t_{1,1}$	$t_{1,2}$	.....	$t_{1,m}$
o_2	$t_{2,1}$	$t_{2,2}$	.....	$t_{2,m}$
o_3	$t_{3,1}$	$t_{3,2}$	.....	$t_{3,m}$
.	.	.	.....	
.	.	.	.....	
.	.	.	.....	
o_n	$t_{n,1}$	$t_{n,2}$	.....	$t_{n,m}$
f	f_1	f_2	.....	f_m

(a) o_i , w_j are specified.

i \ j	1	2	.....	m
1	$t_{1,1}$	$t_{1,2}$	.....	$t_{1,m}$
2	$t_{2,1}$	$t_{2,2}$	.....	$t_{2,m}$
3	$t_{3,1}$	$t_{3,2}$	.....	$t_{3,m}$
.	.	.	.....	
.	.	.	.....	
.	.	.	.....	
n	$t_{n,1}$	$t_{n,2}$	.....	$t_{n,m}$
f	f_1	f_2	.....	f_m

(b) o_i , w_j are not specified.

Figure 3. Assignment Tables

The assignment table contains all input information, and each output is generated and represented on this table. The assignment table consists of information about ready times for operations and workstations (o_i, w_j), processing time (t_{ij}), and the makespan of each workstation (f_j). If the ready times for operations or workstations are not specified, they are replaced with indices of operation or workstation. Of this information, a processing time t_{ij} is contained in a cell (i, j). Thus, each cell (i, j) corresponds to a pair of operation i and workstation j. There are four types of cells.

- Assigned Cell: a cell(i, j) such that operation i is assigned to workstation j.
- Not-Assigned Cell: a cell that is not an assigned cell.
- Leaving Cell: an assigned cell that will be a not-assigned cell in the next iteration during execution of an algorithm.
- Entering Cell: a not-assigned cell that will be an assigned cell in the next iteration during execution of an algorithm.

In the assignment table, the assigned cell is identified by a circle on the cell. At every iteration of the heuristics, the updating process of the assignment table is to choose the leaving and entering cells, update the assignment, and compute the makespan and SOM for the new assignment. The basic process is to reduce the maximum completion time of the current schedule by making new assignments at every iteration.

4. 2. Development of a Heuristic for Subproblem(P1)

The subproblem (P1) is described as follows:

Subproblem (P1): Suppose there are n indepenent subassembly operations and m workstations with different performances (i.e., non-identical, unrelated workstations). Each workstation can perform any subset of subassembly operations. Each subassembly operation can be processed by at most one workstation at a time and preemption is not allowed. Operations are simultaneously available at time zero. Also, all workstations are available at time zero. The problem is to find a schedule that minimizes the makespan.

As shown in Figure 2, the subproblem (P1) deals with the first group of an assembly operation tree for the first product. Since the assembly operations

are in the first group of the first product, there are no preceding operations and no workstations occupied. Based on the problem description, subproblem (P1) can be formulated as an integer programming problem as follows:

Given m: the number of workstations

n : the number of independent operations

t_{ij} : the duration of operation i on workstation j

Minimize y

Subject to $$y - \sum_{i=1}^{n} t_{ij} \cdot x_{ij} \geq 0 \qquad j=1 \text{ to } m$$

$$\sum_{j=1}^{m} x_{ij} = 1 \qquad i=1 \text{ to } n$$

$$x_{ij} = 0 \text{ or } 1 \text{ for all } i, j.$$

where y : the makespan of the schedule

x_{ij}: the indicator variable defined as follows:

$x_{ij} = 1$ if operation i is assigned to workstation j

$x_{ij} = 0$ otherwise.

Let algorithm (A1) be a heuristic algorithm for subproblem (P1). The input and output of algorithm (A1) are:

Input

$1 \leq i \leq n$: index of operatons

$1 \leq j \leq m$: index of workstations

t_{ij} : the processing time of operation i on workstation j for all i and j.

Output

x_{ij} : the indicator variable

x_{ij} =1 if operation i is assigned to workstation j

$x_{ij} = 0$, otherwise.

f : the makespan to complete all operations

$$f = \max_{1 \leq j \leq m} \left\{ f_j = \sum_{i=1}^{n} t_{ij} \cdot x_{ij} \right\}$$

where f_j is a makespan on workstation j.

f_t : SOM

$$f_t = \sum_{j=1}^{m} f_j$$

The ready times for operations and workstations are zero in subproblem (P1), and so are ignored as input. Also even after the scheduling is done, the starting time of each operation need not be specified, because for a given workstation its makespan is not affected by the assigned order of operations on that workstation. Algorithm (A1) for solving subproblem (P1) is as follows:

ALGORITHM (A1)
(An Algorithm for Subproblem (P1))

Step 1. (Initial Solution)

Assign each operation i to the workstation j that gives the minimal processing time (if a tie occurs, choose arbitrarily). Then, make the initial assignments:

$$x_{ik} \leftarrow 1,$$

$$x_{ik} \leftarrow 0 \text{ for all } j \text{ such that } j \neq k$$

Compute a makespan for each workstation j given as:

$$f_j = \sum_{i=0}^{n} t_{ij} \cdot x_{ij} \text{ for all } j$$

Compute SOM: $$f_t = \sum_{j=1}^{m} f_j$$

Also, let R be a set of workstations that have the same maximal makespans.

Arbitrarily choose one workstation $u \in R$ and $R \leftarrow R - u$

$L_1 \leftarrow \phi$, $L_2 \leftarrow \phi$.

$k \leftarrow 1$.

At ⟨ iteration k⟩,

Step 2. (Choose the Leaving and Entering Cells)

Choose the leaving and entering cells for the following two cases.

(Case 1) Let Q be a set of cells such that

$Q = \{(i, j) \mid x_{iu} = 1,\ t_{ij} < f_u - f_j, (i, j) \notin L_1\}$ for *all* i, j *but* $j \neq u$.

The entering cell (p, j_1) is chosen such that

$$t_{pj_1} - t_{pu} = \min_{(i,j) \in Q} \{t_{ij} - t_{iu}\} = t^1.$$

If a tie occurs, choose the cell that includes the operation with the maximal processing time on workstation u.

Then the leaving cell is (p, u)

$$L_1 \leftarrow L_1 \cup (p, u)$$

(Case 2) Let T be a set of (i_1, i_2, j) which satisfies:

$$x_{i_1u}=1, \qquad x_{i_2j}=1,$$

$$t_{i_1u} > t_{i_1j}, \qquad t_{i_1j}-t_{i_2u}<f_u-f_j,$$

$(i_2, u) \notin L_2$ or $(i_1, j_2) \notin L_2$

for all $i_1 \neq i_2$ and all j but $j \neq u$.

Choose (q, r, j_2) such that

$$(t_{qj}-t_{rj})+(t_{ru}-t_{qu}) = \min_{(i_1, i_2, j)\in T}\{(t_{i_1j}-t_{i_2j})+(t_{i_2u}-t_{i_1u})\}=t^2.$$

If a tie occurs, choose one (q, r, j_2) that results in the largest reduction of f_u.

Then the leaving cells are (q, u) and (r, j_2).

The entering cells are (r, u) and (q, j_2).

$L_2 \leftarrow L_2 \cup (q, u) \cup (r, j_2)$.

If $Q = T = R = \phi$, stop.

If $Q = T = \phi$ and $R = \phi$, arbitrarily choose one workstation $u \in R$ and do step 2 again.

Otherwise, go to next step.

Step 3. (Update Solution)

Let t = min $\{t^1, t^2\}$.

(Case 1) If $t = t^1$, make new assignments: $x_{pu} \leftarrow 0$ and $x_{pj_1} \leftarrow 1$.

Compute makespans f_u, f_{j1}, and f_t :

$$f_u \leftarrow f_u - t_{pu}, \qquad f_{j1} \leftarrow f_{j1} + t_{pj_1},$$

$$f_t \leftarrow f_t + t_1.$$

(Case 2) If $t = t^2$, make new assignments

$$x_{qu} \leftarrow 0, \qquad x_{rj_2} \leftarrow 0,$$

$$x_{ru} \leftarrow 1, \qquad x_{qj_2} \leftarrow 1.$$

Compute makespans f_u, f_{j2}, and f_t :

$$f_u \leftarrow f_u-(t_{qu}-t_{ru}), f_{j_2} \leftarrow f_{j_2} + (t_{qj_2}-t_{rj_2}) + (t_{qj_2}-t_{rj_2}),$$

$$f_t \leftarrow f_t+t^2.$$

Construct R and arbitrarily choose one workstation $u \in R$.

$R \leftarrow R-u$ and increase iteration $k \leftarrow k+1$. Go to step 2.

In steps 2 and 3, the objective is to decrease the largest makespan f_u by partially changing the assignment of operations so that the whole makespan for the current assignment is also decreased. There are two ways to do this.

Case 1: The makespan f_u is decreased by deleting one assigned operation on workstation u (choose the leaving cell). For illustration, consider an assignment table with eight operations and four workstations as shown in Figure 4. The assigned cells are marked with circles. As shown in (a) of Figure 4, the

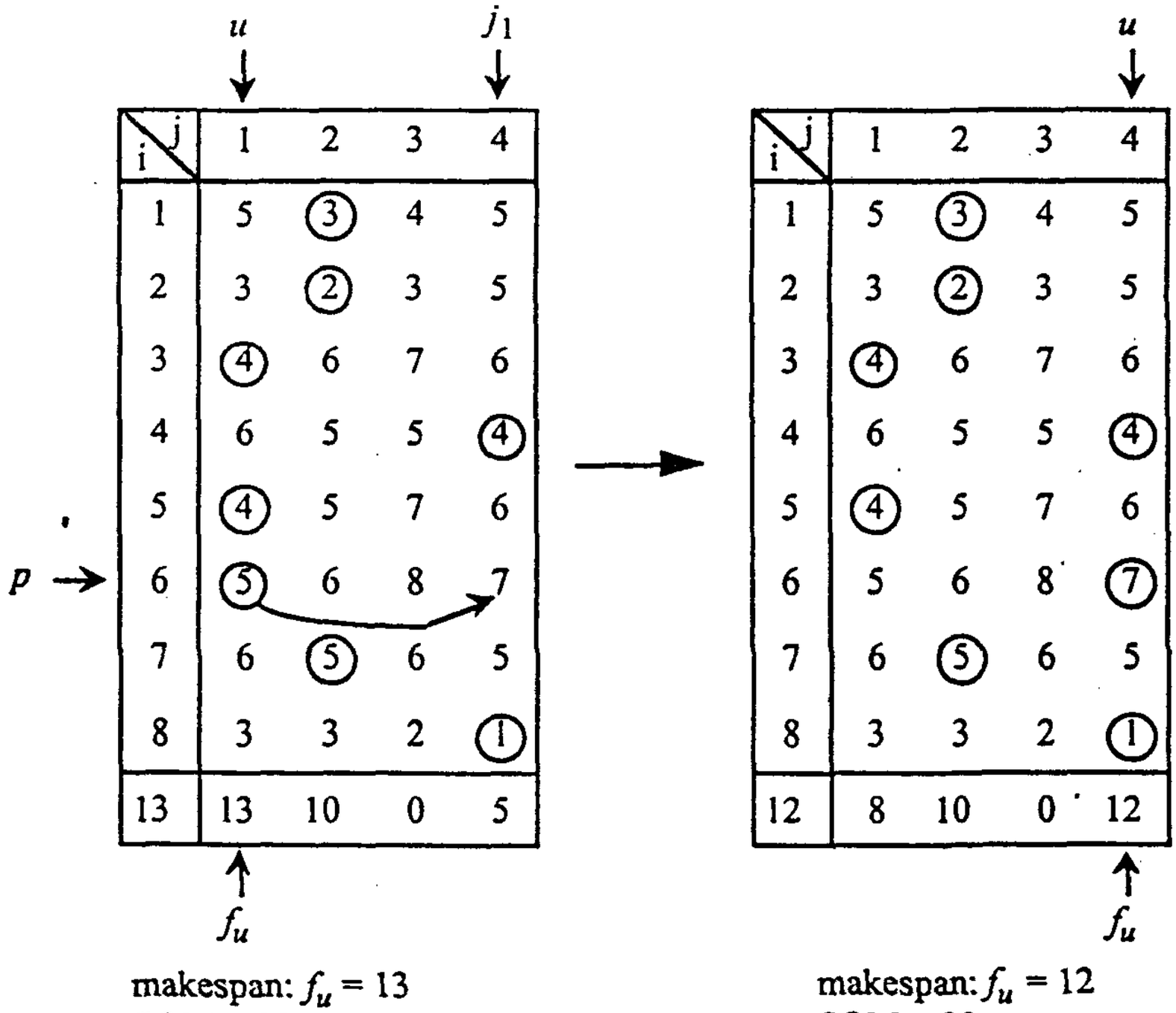


Figure 4 The Improvement of Solution: Case 1 of Algorithm (A1)

makespan for the initial assignment is 13 and there is one corresponding workstation u=1. Thus the leaving cell is chosen on workstation 1. The set Q of candidate entering cells is:

Q = {(3,3), (3,4), (5,3), (5,4), (6,3), (6,4)}.

To choose an entering cell compute the following:

for cell (3,3), $t_{3,3}-t_{3,1}=7-4=3$

for cell (3,4), $t_{3,4}-t_{3,1}=6-4=2$

for cell (5,3), $t_{5,3}-t_{5,1}=7-4=3$

for cell (5,4), $t_{5,4}-t_{5,1}=6-4=2$

for cell (6,3), $t_{6,3}-t_{6,1}=8-5=3$

for cell (6,4), $t_{6,4}-t_{6,1}=7-5=2$

Since min {3, 2, 3, 2, 3, 2,} = 2, one of three cells (3,4), (5,4), and (6,4) is chosen as the entering cell. Since operation 6 has a maximum processing time, the cell (6, 4) is chosen as the entering cell. Then, the leaving cell is (6, 1). The improved assignment is shown in (b), and the improved makespan is 12 on workstation 4. The SOM is 30.

Case 2: In this case, the basic process is to switch two assigned operations between two workstations. One operation is from workstation u, and the other operation is from other workstations.

The upper bound for computing time of Algorithm (A1) is $O(n^4 m^3)$. The chosen workstation u is one of the workstations with a maximal makespan and is always improved at each iteration $f_u^{k+1} < f_u^k$. Even if there are more than one workstation with a maximal makespan (i. e., $R \neq \phi$), one of them is chosen at each iteration and improved. Finally, all of them are improved until the stopping criterion is satisfied. Algorithm (A2) is exactly same as algorithm (A1) except the computation of makespans.

4. 2. Development of a Heuristic for Subproblems (P3)

Subproblem (P3) deals with all groups except the first group of an assembly operation tree for each product. Since there are the precedent operations and workstations may be available at different times, this problem may have the non-zero ready times for operations as well as workstations. Thus, subproblem (P3) is the most general problem of type

III problems. In the subproblem (P1) and (P2), the operations assigned to a given workstion do not need a particular scheduling procedure because any schedule without inserted idle time gives a minimal makespan. However, in the subproblem (P3) the operations assigned to a given workstion must be scheduled to minimize the makespan. Therefore, the design of a heuristic procedure for the subproblem (P3) must consider a procedure to schedule the operations assigned to a given workstation. This problem is called the "Single Processor Scheduling Problem" (SPSP). Description of the SPSP and its mathematical model are as follows:

Single Processor Scheduling Problem (SPSP): Suppose there is a set of independent subassembly operations with a given processor (a workstation). Each operation has the ready time at which the operation is available for processing. The problem is to find a schedule that minimizes the makespan.
Given I : the set of operations

t_i : the processing time of operation i

o_i : the ready time of operation i

s_i : the starting time of operation i

Minimize Maximum(s_i+t_i) for all $i \in I$

Subject to:

(Constraint Set 1) The workstation cannot process more than one operation at a time. For all operation combinations i_1, $i_2 \in I$, the following should be satisfied:

$$s_{i1} + t_{i1} \leq s_{i2}$$

or

$$s_{i1} \geq s_{i2} + t_{i2}$$

(Constraint Set 2) Each operation can start only after its ready time. For each operation the following must be satisfied:

$$s_i \geq o_i$$

In order to solve this problem, our optimal algorithm employs the earliest ready time (ERT) procedure. In the ERT procedure, first we sequence the operations in nondecreaseing order of ready times called ERT order and schedule the operations in this order. The ERT algorithm provides an opti-

mal schedule.

ALGORITHM ERT
(An Algorithm for the SPSP)

Step 1. Construct an ERT ordering of operations in I.
$f \leftarrow 0$.

Step 2. Schedule the operations in ERT order while satisfying the ready times. Each time assign an operation I, computer s_i and f as follows:

$$s_i = \max \{f, o_i\}$$
$$f = s_i + t_i$$

The computing time of algorithm ERT is O ($n \log n$).

Theorem

The algorithm ERT generates an optimal schedule with the minimal makespan.

Proof: Let S be the schedule generated by the algorithm ERT.

(Case 1) If there is no inserted idle times, obviously S is an optimal schedule. Then the minimal makespan f is:

$$f = o_{i(1)} + \sum_{i \in I} t_i$$

where $o_{i(k)}$: the ready time of operation $i(k)$

i(k) : an index for the kth operation in the ERT order.

(Case 2) Suppose there are inserted idle times in S. There may be a number of inserted idle time periods. Suppose there are p inserted idle time periods. The jth idle time period has a time period has a time period $[is_j, is_j + it_j]$. Let n_j be the number of operations scheduled before the jth idle time period. Consider the first idle time period. Then n_1 operations are scheduled without inserted idle times, i. e., $is_1 = o_{i(1)} + \sum_{k=1}^{n1} t_{i(k)}$. This means that the schedule for these n_1 operations is an optimal schedule with the makespan is_1. Let us check if any operation except any of these n_1 operations can be scheduled during the first idle time period $[is_1, is_1 + it_1]$.

There is no such operation because for all operations $i_{(k)}$, $k>n_1$, $o_{i(k)} \geq is_1 + it_1$. Thus any operation $i(k)$ such that $k>n_1$ should be assigned after is_1+it_1. Also the next n_2 operations are scheduled without inserted idle times, i. e., $is_2 = o_{i(n1+1)}) + \sum_{k=n1+1}^{n2} t_{i(k)}$. Thus the schedule for these n_1 and n_2 operations is an optimal schedule with the makespan is_2. By induction, the schedule for the $\sum_{i=1}^{j} n_i$ operations is also an optimal schedule with the makespan $is_j = o_i(n_{j-1}+1)+ \sum_{k=nj-1+i}^{nj} t_{i(k)}$. Thus the schedule S for all operations is an optimal schedule with the makespan $f=o_i(n_p+1)+ \sum_{k=np+1}^{n} t_{i(k)}$.

For example, Figure 5 shows the application of Algorithm ERT to a six-operation problem. The detailed steps for this problem are as follows:

Step 1 : Construct an ERT ordering of operations. The ERT ordering is

3-5-1-2-6-4.

f=0.

Step 2 : Schedule the operations in ERT order.

Operation 3 : S_3 = max {0, 1}=1. f=1+2=3.

Operation 5 : S_5 = max {3, 2}=3. f=3+1=4.

Operation 1 : S_1 = max {6, 6}=6. f=6+3=9.

Operation 2 : S_2 = max {9, 7}=9. f=9+1=10.

Operation 6 : S_6 = max {10, 7}=10. f=10+2=12.

Operation 4 : S_4 = max {12, 13}=13. f=13+3=16.

Thus the minimum makespan is 16.

All procedures of Algorithm (A3) are the same as those for algorithm (A1), except the computing procedure of f_j and S_j. In algorithm (A3) the makespan f_j and the schedule S_j on workstation j are computed using algorithm ERT. The total computing time of algorithm (A3) is bounded by $O(n^5 (\log n)m^2)$, and the makespan of each iteration is always improved until the stopping criterion satisfies.

i	t_i	o_i
1	3	6
2	1	7
3	2	1
4	3	13
5	1	2
6	2	7

k	$i(k)$	$o_{i(k)}$
1	3	1
2	5	2
3	1	6
4	2	7
5	6	7
6	4	13

an ERT order of operations

i : an index of operations
t_i : the processing time of operation i
o_i : the ready time of operation i
k : the k th ERT order
i (k) : an operation of the k th ERT order

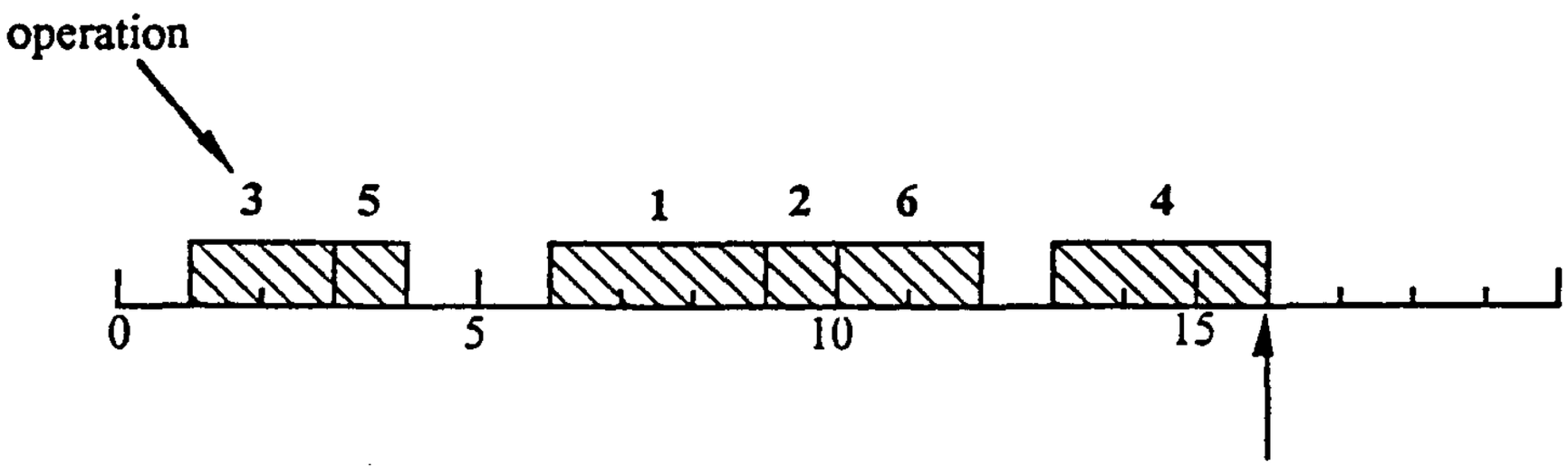


Figure 5. The Application of Algorithm ERT to a Six-Operation Problem

5. EXPERIMENTAL RESULTS

The efficiency of Algorithm (A1), (A3), and MFASSP scheduling procedure was investigated. All algorithms were coded in C language and run on a VAX computer. Algorithms (A1) and (A3) were tested on problems with combinations of n=10, 20, 50, and 100 operations and m=5, 10, 20 and 30 workstations. For each value of n and m, five problems were generated using a random number generator. Each problem was designed to have the best-known schedule. The efficiency of each heuristic is evaluated by the average relative deviation (D) from the known optimal makespan, i.e., D=100 $(f_h-f_{opt})/f_{opt}$ where f_h, f_{opt} are the makespans of the schedule generated by our heuristic and the optimal schedule, respectively. Also, the average relative deviation of SOM is computed. For the MFASSP scheduling procedure, we tested twelve problems with combinations of np=10, 20, 50, and 100 products and m=5, 10 and 20 workstations. And, for each value of np and m, five problems were generated. Since a product k has n_k assembly operations with $10 \le n_k \le 30$, each problem consists of np products and N assembly operation where n_k assembly operations with $10 \le n_k \le 30$, each problem consists of np products and N assembly operations where $N=\sum_{k=1}^{np} n_k$. For ease of experimentation, we simplified the MFASSP by assuming that each product has one assembly sequence and each assembly sequence has no precedence constraints. The results show that our heuristics provide quite good solutions which are very close to the optimal solutions, as follows.

Table 2. Efficiency of Heuristics

Heuristics	Deviation (%) from Opt. Sol.		
	min	max	avg.
Algorithm (A1)	0.00	1.09	0.40
Algorithm (A3)	0.00	1.53	0.42
MFASSP Scheduling	0.00	1.75	0.84

For all heuristics, the computation cost is low enough to be implemented in the real FAS. Algorithms (A1) and (A3) take less than a second and a minute, respectively, for 100-operation problems. The MFASSP scheduling heuristic also solves all large scale problems (i.e., np=100, N=2,000) in ten minutes.

6. CONCLUSIONS

Unlike the current evaluation systems of assembly sequences, good assembly sequences can be chosen based on the FAS performance. The selection criterion used in this study is the makespan to finish all assemblies. Of many alternative sequences, the assembly sequence with a minimal makespan is chosen as the best assembly sequence. In order to do this, the Modified FAS Scheduling Problem (MFASSP) was formulated and a new heuristic algorithm was developed to solve this problem. This heuristic algorithm provided very satisfactory results for both quality of solution and computation time.

The heuristic for solving the MFASSP can be used to solve many variants of scheduling problems because the MFASSP consists of several different subproblems and each subproblem is a different type of scheduling problem. Also, if there is no alternative assembly sequence for each product, the MFASSP becomes the FAS Scheduling Problem (FASSP). The MFASSP is easily solved by applying the MFASSP scheduling procedure without a selection process.

REFERENCES

[1] Baer, J. L., "Optimal scheduling on two processors with different speeds," *Computer Architectures and Networks*, E. Gelenbe and R. Mahl (Editors), North-Holland, Amsterdam, pp. 27−45, 1974.

[2] Baker, K. R., Introduction to Sequencing and Scheduling, John Wiley, New York, 1974.

[3] Baldwin, D. F. et.al., "An integrated computer aid erating and

evaluating assembly sequences for mechanical products," *IEEE Transaction on Robotics and Automation*, Vol. 7, No. 1, pp. 78−94, Feb. 1991.

[4] Boothroyd, G. and Dewhurst, P., *Product Design for Assembly*, Boothroyd Dewhurst, Inc. 1989.

[5] Chay, D., Lenz, E., and Shpitalni, M., "Picking the parameters to ease automation of assembly," *Assembly Automation*, Vol. 8, No. 3, 1988.

[6] Coffman, E. G., Garey, M. R., and Johnson, D. S., "An application of bi-packing to multiprocessor scheduling" *SIAM Journal of Computation*, Vol. 7, No. 6, pp. 1−17, Feb. 1978.

[7] Donath, M., Graves, R., and Carlson, D. A., "Flexible assembly systems: The scheduling problem for multiple products," *Journal of the Manufacturing Systems*, Vol. 8, No. 1, pp. 27−33, 1989.

[8] Hariri, A. M. A and Potts, C. N., "Heuristics for scheduling unrelated parallel machines," *Computers & Operations Research*, Vol. 18, No. 3, pp. 323−331, 1991.

[9] Heemskerk, C. J. M., "The use of heuristics in assembly sequence planning," *Annals of CIRP*, Vol. 38, No. 1, pp. 37−40, 1989.

[10] Horowitz, E., "Exact and approximate algorithms for scheduling nonidentical processors," *Journal of ACM*, Vol. 23, No. 2, pp. 317−327, April 1976.

[11] Hu, T. C., "Parallel sequencing and assembly line problems," *Operations Research*, Vol. 9, No. 6, pp. 841−848, Nov. 1961.

[12] Huang, Y. F. and Lee, C. S. G., "A framework of knowledge-based assembly planning," *Proceedings of the 1991 IEEE, International Conf. on Robotics and Automation*, pp. 599−604, April 1991.

[13] Ibarra, O. H. and Kim, C. E., "Heuristic algorithms for scheduling independent tasks on nonidentical processors," *Journal of ACM*, Vol. 24, pp. 280−289, March 1977.

[14] Jeong, B. J., "A computer-aided evaluation system for assembly sequences in flexible assembly systems," Ph.D. dissertation, Pennsylvania State University, Dept. of I&MSE, University Park, PA, August 1993.

[15] Laperriere, L. and EIMaraghy, H. A. "Planning of products assembly and disassembly," *Annals of CIRP*, Vol. 41, No. 1, pp. 5−9, 1992.

[16] Lee, C. H., "Development of a computer integrated assembly planning

system for mechanical assembly." Ph.D. dissertation, Pennsylvania State University, Dept. of I&MSE, State College, PA, 1992.

[17] Liu, J. W. S. and Liu, C. L., "Bounds on scheduling algorithms for heterogeneous computing systems," *Information Processing* 74, J. L. Rosenfield (editor), North-Holland, Amsterdam, pp. 349−353, 1974.

[18] McNaughton, R., "Scheduling with deadlines and loss functions," *Management Science*, Vol. 6, No. 1, Oct. 1959.

[19] Miyakawa, S. and Ohashi, T., "The Hitachi assimilability evaluation method," *Proceedings of the 1st Int. Conf. on Product Design for Assembly, 1986.*

[20] Muntz, R. R. and Coffman, E. G., "Optimal preemptive scheduling on two-processor systems," *IEEE Transactions on Computers*, Vol. 18, No. 11, Nov. 1969.

[21] Muntz, R. R. and Coffman, E. G., "Preemptive scheduling of real-time tasks on multiprocessor systems," *Journal of the ACM*, Vol. 17, No. 2, April 1970.

[22] Poli, C., "A design for assembly spreadsheet," *Design and Synthesis*, H. Yoshikawa (editor), Elsevier Science Publishers, North-Holland, 1985.

[23] Wang, Q and Cheng, K. H., "A heuristic of scheduling parallel tasks and its analysis," *SIAM Journal of Commputation*, Vol. 21, pp. 281−294, April 1992.

Ick-Hyun Nam is an assistant professor at the College of Business Administration, Seoul National University. He received his B.B.A. and M.B.A. from Seoul National University. He received an M.S. in Operations Research from Stanford University. He completed his Ph.D at the Graduate School of Business, Stanford University, 1993. His research interests cover queueing network, inventory control, and logistics management.

Hyunkon Kahng is an assistant professor of MS / POM at the department of Business Administration, Yonsei University—Wonju. He received his Ph.D. in Business from the University of Chicago in 1992. His current research interests include computer and quantitative techniques to practical problems.

Gueongbeom Yi is an assistant professor in the Department of Chemical Engineering, Pusan National Fisheries University. He finished his undergraduate study at the Department of Chemical Engineering, Seoul National University (1983). He received his M.S. degree from the Department of Chemical Engineering, KAIST (1985) and his Ph.D. degree in school of Chemical Engineering, Purdue University (1992). He served as a process engineer at Ssangyong Oil Refinery during 1985—1988 and as a research manager at Honam Oil Refinery during 1992—1995. His research interest is placed on production planning and scheduling of chemical processes, process design, process control and optimization.

Bongju Jeong is an assistant professor at the Industrial Systems Engineering Department in Yonsei University. He received his B.S. and M.S. degrees in Industrial Engineering from Seoul National University and his Ph.D. degree in the same area from Pennsylvania State University in 1993. He worked as a manufacturing systems consultant at the Arthur Andersen Consulting Co. and as a senior researcher at Samsung Electronics Co. during 1993—1995. His research interests are in the areas of manufacturing systems engineering, information systems engineering, group technology applications, and robotics.