Journal of the Operations Research
Society of Japan
Vol. 20, No.3, September, 1977

THREE-MACHINE SCHEDULING PROBLEM WITH PRECEDENCE CONSTRAINTS

TADASHI KURISU

Osaka University

(Received May 2, 1977; Revised July 5, 1977)

Abstract This paper deals with a three-machine flow-shop problem in which some of the job sequences are infeasible. It is assumed that jobs are grouped into several disjoint subsets within which a job order is pre-determined. Once the first job in a group has started on a machine, then the entire group must be completed on the machine without starting a job which does not belong to the group. It is further assumed that a precedence relation between groups is given such that the processing of the jobs in a group must be completed on each machine before the jobs in another group begin on the machine. It is shown that it suffices to consider only permutation schedules for minimizing the total elapsed time and then some restricted cases are solved.

1. Introduction

Johnson [4] considered the following problem (which will be called a three-machine n-job flow-shop problem). There are given jobs 1, 2, ... , n, each to be processed on three machines I, II and III in the same order I, II, III. Each machine can handle only one job at a time and each job must be processed on only one machine at a time. Given the processing times on these machines, the problem is to find a job order for each machine so as to minimize the total elapsed time necessary to process all these jobs. This problem was considered by many researchers. Johnson has shown that it is sufficient to consider only schedules in which the same job order occurs on three machines. Johnson [4], Arthanari et al. [1], Burns et al. [2], Szwarc [8] and Smith et al. [7] solved for some restricted cases. In all these papers, it is assumed that every permutation of n jobs is feasible. In most of the practical situations, however, certain orderings are prohibited either by technological constraints or by externally imposed policy. Such situations may occur many times in everyday life:

(*a*) If setup times are highly dependent on sequence, then one may group jobs with similar setups, sequence within these groups for minimum changeover time, and arrange the groups to minimize the total elapsed time.

(*b*) If due-date is associated with each job, then it may be effective to process the jobs with earlier due-date before the jobs with later due-date.

(*c*) If there are jobs which should be re-processed after once they have been processed, then the first processing must be completed before the second one starts.

The object of this paper is to obtain a schedule minimizing the total elapsed time subject to such general precedence constraints.

2. Problem and Notation

Consider a flow-shop consisting of n jobs 1, 2, ... , n and three machines I, II and III. All jobs are to be processed on these machines according to the order I, II, III. Each job can be processed at a time on a machine and each machine can process only one job at a time. Associated with each job i are processing times A_i, B_i and C_i on machines I, II and III, respectively, and they are known prior to making scheduling decisions.

An ordered set of jobs $I_i = (s, t, \dots, u)$ is called a string if and only if the jobs $s, t, \dots, u$ must be processed in that order, without preemption between jobs, on each machine. Of course, there may be idle times, on machines II and III, between jobs in a string. However, once the first job in a string has started on a machine, then all jobs in the string must be processed according to the fixed order to be completed on the machine without starting a job which does not belong to the string. We assume that the original n jobs have been grouped into m strings $I_1, I_2, \dots, I_m$ and we set $X = \{I_1, I_2, \dots, I_m\}$. Let n_i be the number of jobs in the string I_i and let A_{ij}, B_{ij} and C_{ij} be the processing times on machines I, II and III, respectively, for the j-th job in the string I_i (A_{ij}, B_{ij} and C_{ij} are equal to A_k, B_k and C_k, respectively, for some k). We further assume that a precedence relation ">" on X is given such that if $I_i > I_j$, then the processing of jobs in I_i must be completed on each machine before the jobs in I_j start on the machine. If $I_i > I_j$ and if there is no string, I_k, such that $I_i > I_k > I_j$, then we denote by $I_i >> I_j$. It is convenient to illustrate these relationships on a precedence graph such as $G^* = (X, U)$ indicated in Fig. 1, where U denotes the set of arrows. The nodes of the graph represent the strings and the arrows represent "directly precedes" relationships between the strings. The precedes relationship exists between two strings if there is a path of arrows between them.

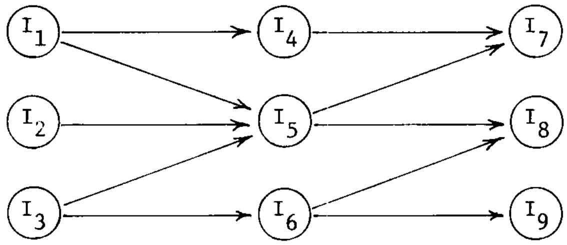

Fig. 1. Precedence graph G*

For a precedence graph $G = (X, U)$, we set

$$P(I_i, G) = \{I_j \in X \mid I_j >> I_i\},$$

$$Q(I_i, G) = \{I_j \in X \mid I_i >> I_j\},$$

$$P(G) = \{I_i \in X \mid P(I_i, G) = \phi\}$$

and

$$Q(G) = \{I_i \in X \mid Q(I_i, G) = \phi\}.$$

P(G) and Q(G) denote the sets of strings which can be sequenced first and last, respectively, in a feasible schedule.

In the following, we develop algorithms to produce a schedule which minimizes the total elapsed time for the three-machine flow-shop problem with precedence constraints represented by a precedence graph G.

3. Permutation Schedules

Much of the simplicity of the two-machine flow-shop problem can be attributed to the fact that it is sufficient to consider only permutation schedules, which are completely described by a particular permutation of the job identification numbers. Johnson has shown that it suffices to consider only permutation schedules, for three-machine flow-shop problems, when all jobs are simultaneously available. This is generalized by the following theorem:

Theorem. In a three-machine flow-shop problem, for minimizing the total elapsed time subject to precedence constraints, it suffices to consider only schedules in which the same string order is prescribed on machines I, II and III.

Proof: (i) If a feasible schedule Π' does not have the same string order on machines I and II, then somewhere in the schedule for machine I there must be a string I_i that is ordered directly before a string I_j where I_i follows I_j possibly with intervening strings on machine II. Since I_j is ordered before I_i on machine II in Π', the positions of these two strings can be reversed on machine I without yielding the infeasibility of the schedule. Furthermore, this exchange does not cause an increase in the starting time of any job on machine II, and therefore on machine III. Thus, this exchange does not cause an increase in the completion time of any job, and hence, not an increase in the total elapsed time. Therefore, we may consider only schedules in which the same string order occurs on machines I and II. (ii) Suppose that a feasi-

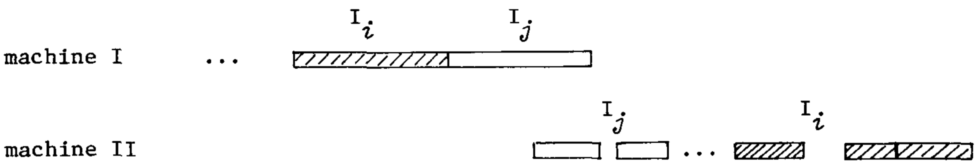

Fig. 2. Schedule Π'

ble schedule does not have the same string order on machines II and III. Then somewhere in the schedule for machine III there must be a string I_i that directly follows a string I_j, where I_i is ordered before I_j on machine II. Obviously, the positions of these two strings can be reversed on machine III without increasing the maximum completion time of the jobs in these strings, and hence, the total elapsed time is not increased by the exchange. Since I_i is ordered before I_j on machine II, the exchanged schedule is feasible for the precedence constraints. This terminates our proof.

We denote by T(S) the total elapsed time for a sequence S. Furthermore, we represent the job sequenced in the i-th position in S by $[i]$ and the string sequenced in the i-th position in S by $I_{(i)}$. Thus, for the three-machine case, T(S) is denoted as follows (see Johnson [4]):

$$(1) \qquad T(S) = \max_{1 \leq u \leq v \leq n} \{ \sum_{i=1}^{u} A_{[i]} + \sum_{i=u}^{v} B_{[i]} + \sum_{i=v}^{n} C_{[i]} \}.$$

4. Some Special Cases

Prior to discussion on three-machine problems, we briefly review two-machine n-job flow-shop problem. Let A'_i be the processing time for job i on machine I and let B'_i be the corresponding time on machine II. As in the three-machine case, we assume that each job can be processed at most on a machine at a time and that each machine can handle only one job at a time. Then the total elapsed time $\bar{T}(S)$ from the start of the first job on machine I until the completion of the last job on machine II is represented by

$$(2) \qquad \bar{T}(S) = \max_{1 \leq u \leq n} \{ \sum_{i=1}^{u} A'_{[i]} + \sum_{i=u}^{n} B_{[i]} \}.$$

For the two-machine flow-shop problem with precedence constraints, the author developed in [5, 6] an efficient algorithm to produce a sequence minimizing the total elapsed time. Now, we treat three-machine flow-shop problems with precedence constraints.

Case 1: $\min_i A_i \geq \max_i B_i$.

Then

$$A_i \geq B_j \qquad \text{for all } i \text{ and } j,$$

and hence, the maximum value in (1) is attained by setting $u = v$ for each v. Therefore,

$$T(S) = \max_{1 \leq v \leq n} \{ \sum_{i=1}^{v} A_{[i]} + B_{[v]} + \sum_{i=v}^{n} C_{[i]} \}$$

$$= \max_{1 \leq v \leq n} \{ \sum_{i=1}^{v} (A_{[i]} + B_{[i]}) + \sum_{i=v}^{n} (B_{[i]} + C_{[i]}) \} - \sum_{i=1}^{n} B_i.$$

Since $\sum_{i=1}^{n} B_i$ is a constant, we may get a sequence minimizing

$$(3) \qquad T^*(S) = \max_{1 \leq v \leq n} \{ \sum_{i=1}^{v} (A_{[i]} + B_{[i]}) + \sum_{i=v}^{n} (B_{[i]} + C_{[i]}) \}.$$

Comparing (2) and (3), it is seen that case 1 has a two-machine n-job structure. Thus, to solve the three-machine flow-shop problem with precedence constraints, we can use the technique for getting an optimal sequence for the two-machine problem with the precedence constraints, assuming that the processing times of job i on machines I and II are $A_i + B_i$ and $B_i + C_i$, respectively. The total elapsed time T(S) for the three-machine problem is

$$T(S) = T^*(S) - \sum_{i=1}^{n} B_i,$$

where $T^*(S)$ defined by (3) denotes the total elapsed time of the sequence S for the equivalent two-machine problem.

Example 1. Consider nine strings with the precedence graph G^* as was indicated in Fig. 1. We assume that each string I_i consists of a job i and that the processing times of these jobs are given in Table 1. Since $\min_i A_i = 6 = \max_i B_i$, we are justified in applying the method described above.

Table 1. Processing times for Example 1

i	1	2	3	4	5	6	7	8	9
A_i	6	8	8	7	7	10	8	9	6
B_i	3	4	2	6	6	3	4	2	5
C_i	3	7	9	10	9	9	5	10	2

Table 2. Processing times for equivalent two-machine problem

i	1	2	3	4	5	6	7	8	9
$A_i + B_i$	9	12	10	13	13	13	12	11	11
$B_i + C_i$	6	11	11	16	15	12	9	12	7

The processing times for the equivalent two-machine problem are shown in Table 2. Applying the algorithm in [6], we get three candidate sequences

$$S_1 = (3, 1, 4, 2, 5, 6, 8, 7, 9),$$
$$S_2 = (3, 6, 2, 1, 5, 8, 4, 7, 9)$$

and

$$S_3 = (3, 2, 1, 5, 4, 6, 8, 7, 9).$$

For the equivalent two-machine problem, the total elapsed times of these sequences are as follows:

$$T^*(S_1) = 114, \qquad T^*(S_2) = 116, \qquad T^*(S_3) = 115.$$

Thus, S_1 is an optimal sequence and the total elapsed time of this sequence is 79 time units for the original three-machine problem.

Case 2: $\min_i C_i \geq \max_i B_i$.

Then the maximum value in (1) is attained by setting $v = u$ for each u. Thus,

$$\begin{aligned} T(S) &= \max_{1 \leq u \leq n} \{ \sum_{i=1}^{u} A_{[i]} + B_{[u]} + \sum_{i=u}^{n} C_{[i]} \} \\ &= \max_{1 \leq u \leq n} \{ \sum_{i=1}^{u} (A_{[i]} + B_{[i]}) + \sum_{i=u}^{n} (B_{[i]} + C_{[i]}) \} - \sum_{i=1}^{n} B_i . \end{aligned}$$

Therefore, we can obtain an optimal sequence by the similar method as in case 1.

Case 3: $\max_i A_i \leq \min_i B_i$.

Then

$$A_i \leq B_j \qquad \text{for all } i \text{ and } j,$$

and hence, the maximum value in (1) is attained by setting $u = 1$ for each v. Therefore, we have

$$\begin{aligned} T(S) &= \max_{1 \leq v \leq n} \{ A_{[1]} + \sum_{i=1}^{v} B_{[i]} + \sum_{i=v}^{n} C_{[i]} \} \\ &= A_{(1)1} + \max \left\{ \begin{array}{l} \max\limits_{1 \leq v \leq n_{(1)}} \{ \sum\limits_{j=1}^{v} B_{(1)j} + \sum\limits_{j=v}^{n_{(1)}} C_{(1)j} \} + \sum\limits_{i=n_{(1)}+1}^{n} C_{[i]}, \\ \sum\limits_{i=1}^{n_{(1)}} B_{[i]} + \max\limits_{n_{(1)}+1 \leq v \leq n} \{ \sum\limits_{i=n_{(1)}+1}^{v} B_{[i]} + \sum\limits_{i=v}^{n} C_{[i]} \} \end{array} \right\} \\ &= A_{(1)1} + \max\{ T'(I_{(1)}) + \sum_{i \notin I_{(1)}} C_i , \ \sum_{i \in I_{(1)}} B_i + T_1(S) \}, \end{aligned}$$

where

$$T'(I_i) = \max_{1 \leq v \leq n_i} \{ \sum_{j=1}^{v} B_{ij} + \sum_{j=v}^{n_i} C_{ij} \}$$

and

$$T_1(S) = \max_{n_{(1)}+1 \leq v \leq n} \{ \sum_{i=n_{(1)}+1}^{v} B_{[i]} + \sum_{i=v}^{n} C_{[i]} \}.$$

Obviously, $T_1(S)$ denotes the total elapsed time of the sequence $(I_{(2)}, I_{(3)}, \ldots, I_{(m)})$ for the two-machine problem with the processing times B_j and C_j, for job j, on machines I and II, respectively. The string sequenced in the first position must be an element in P(G) and $T_1(S)$ must be minimized subject to precedence constraints. Noticing that $T_1(S)$ is minimized by using the procedure proposed by the author in [6], we obtain the following algorithm to produce an optimal sequence for the three-machine problem with a precedence graph G = (X, U):

Algorithm 1.

Step 1. For each string I_i in P(G), eliminate from G the string I_i and the arrows starting from I_i, and let the resultant graph G_i.

Step 2. Assuming that the processing times, for job j, on machines I and II are B_j and C_j, respectively, obtain an optimal sequence for the two-machine problem with the precedence graph G_i. Let S_i be an optimal sequence and let $T_1(S_i)$ be the total elapsed time of the sequence for the two-machine problem.

Step 3. Calculate

$$D_i = A_{i1} + \max\{ \max_{1 \leq v \leq n_i} (\sum_{j=1}^{v} B_{ij} + \sum_{j=v}^{n_i} C_{ij}) + \sum_{j \notin I_i} C_j, \quad \sum_{j \in I_i} B_j + T_1(S_i)\}.$$

Step 4. Find

$$D_{i_0} = \min_{I_i \in P(G)} D_i.$$

Then an optimal sequence for the original three-machine problem with the precedence graph G is given by (I_{i_0}, S_{i_0}). The total elapsed time of the sequence is D_{i_0}.

Example 2. As in Example 1, we assume that each string in G* shown in Fig. 1 consists of a job. The processing times of these jobs are given in Table 3. Since $\max_i A_i = 6 = \min_i B_i$, we are justified in applying the method mentioned above. An optimal sequence is obtained as follows:

Table 3. Processing times for Example 2

i	1	2	3	4	5	6	7	8	9
A_i	3	4	2	6	6	3	4	2	5
B_i	6	8	8	7	7	10	8	9	6
C_i	3	7	9	10	9	9	5	10	2

Step 1. We get three precedence graphs G_1, G_2 and G_3 as indicated in Figs. 3, 4 and 5, respectively. (In these figures, B_i is shown above the description of job i and C_i is shown below the description of job i.)

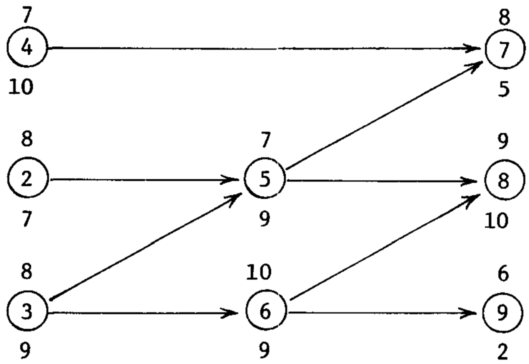

Fig. 3. Precedence graph G_1

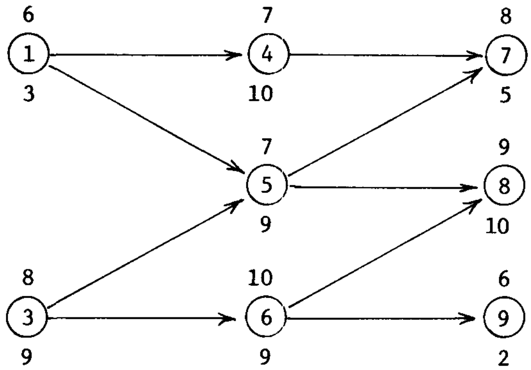

Fig. 4. Precedence graph G_2

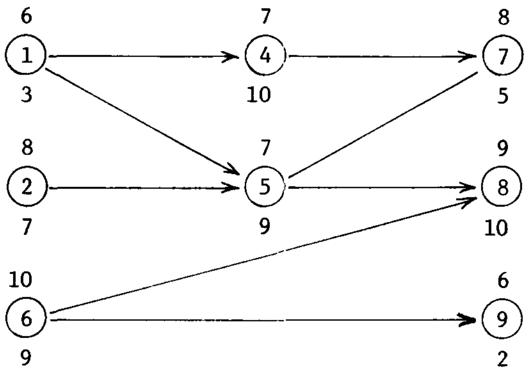

Fig. 5. Precedence graph G_3

Step 2. From G_1, we get one candidate sequence (and hence, an optimal sequence for G_1)

$S_1 = (4, 3, 2, 5, 6, 8, 7, 9)$.

For the two-machine problem, the total elapsed time of this sequence is 68 time units, i.e., $T_1(S_1) = 68$. From G_2, we get two candidate sequences

$S_2 = (3, 1, 4, 5, 6, 8, 7, 9)$

and

$S_2' = (3, 1, 5, 4, 6, 8, 7, 9)$.

The total elapsed time of these sequences is 66 time units, i.e., $T_1(S_2) = T_1(S_2') = 66$. Furthermore, we get, from G_3, two candidate sequences

$S_3 = (1, 4, 2, 5, 6, 8, 7, 9)$

and

$S_3' = (2, 1, 5, 4, 6, 8, 7, 9)$.

The total elapsed times of sequences S_3 and S_3' are 65 and 66, respectively. Thus, S_3 is an optimal sequence for the problem with precedence graph G_3.

Step 3. We have

$$D_1 = 3 + \max\{6 + \sum_{i=1}^{9} C_i, \quad 6 + T_1(S_1)\} = 77,$$

$$D_2 = 4 + \max\{8 + \sum_{i=1}^{9} C_i, \quad 8 + T_1(S_2)\} = 78$$

and

$$D_3 = 2 + \max\{8 + \sum_{i=1}^{9} C_i, \quad 8 + T_1(S_3)\} = 75.$$

Step 4. Since

$$D_3 = \min_{I_i \varepsilon P(G)} D_i,$$

(3, 1, 4, 2, 5, 6, 8, 7, 9) is optimal for the original three-machine problem. The total elapsed time of this sequence is 75 time units.

Case 4: $\max_i C_i \leq \min_i B_i$.

Then the maximum value in (1) is attained by setting $v = n$ for each u. Therefore, we have

$$T(S) = \max_{1 \leq u \leq n} \{ \sum_{i=1}^{u} A_{[i]} + \sum_{i=u}^{n} B_{[i]} + C_{[n]} \}$$

$$= \max \left\{ \begin{array}{l} \max\limits_{1 \leqq u \leqq n'} \{ \sum\limits_{i=1}^{u} A_{[i]} + \sum\limits_{i=u}^{n'} B_{[i]} \} + \sum\limits_{i=n'+1}^{n} B_{[i]} , \\ \\ \sum\limits_{i=1}^{n'} A_{[i]} + \max\limits_{1 \leqq u \leqq n_{(m)}} \{ \sum\limits_{j=1}^{u} A_{(m)j} + \sum\limits_{j=u}^{n_{(m)}} B_{(m)j} \} \end{array} \right\} + C_{mn_{(m)}}$$

$$= \max\{T_2(S) + \sum_{i \in I_{(m)}} B_i, \sum_{i \notin I_{(m)}} A_i + T''(I_{(m)})\} + C_{mn_{(m)}},$$

where

$$n' = n - n_{(m)},$$

$$T_2(S) = \max_{1 \leqq u \leqq n'} \{ \sum_{i=1}^{u} A_{[i]} + \sum_{i=u}^{n'} B_{[i]} \}$$

and

$$T''(I_i) = \max_{1 \leqq u \leqq n_i} \{ \sum_{j=1}^{u} A_{ij} + \sum_{j=u}^{n_i} B_{ij} \}.$$

Obviously, $T_2(S)$ denotes the total elapsed time of the sequence ($I_{(1)}$, $I_{(2)}$, ... , $I_{(m-1)}$) for the two-machine problem with the processing times A_j and B_j, for job j, on machines I and II, respectively. The string which is sequenced in the last position must be an element in Q(G). Hence, we develop the following algorithm to produce an optimal sequence for the three-machine problem with a precedence graph G = (X, U):

Algorithm 2.

Step 1. For each I_i in Q(G), eliminate, from G, the string I_i and the arrows terminating at I_i, and let the resultant graph G_i.

Step 2. Assuming that the processing times, for job j, on machines I and II are A_j and B_j, respectively, obtain an optimal sequence for the two-machine problem with the precedence graph G_i. Let S_i be an optimal sequence and let $T_2(S_i)$ be the total elapsed time of the sequence for the two-machine problem.

Step 3. Calculate

$$E_i = \max\{T_2(S_i) + \sum_{j \in I_i} B_j, \sum_{j \notin I_i} A_j + \max_{1 \leqq v \leqq n_i} (\sum_{j=1}^{v} A_{ij} + \sum_{j=v}^{n_i} B_{ij})\} + C_{in_i}.$$

Step 4. Find

$$E_{i_0} = \min_{I_i \in Q(G)} E_i.$$

Then (S_{i_0}, I_{i_0}) is an optimal sequence for the three-machine problem with the precedence graph G. The total elapsed time of this sequence is E_{i_0}.

Acknowledgements

The author would like to express his sincere thanks to Professor T. Nishida and Professor M. Sakaguchi of Osaka University for their continuing guidances and encouragement. He would also like to thank the referees for their helpful comments and suggestions.

References

1. Arthanari, T. S., and Mukhopadhyay, A. C.: A Note on a Paper by W. Szwarc. *Naval Research Logistics Quarterly*, Vol. 18 (1971), pp. 135-138.
2. Burns, F., and Rooker, J.: A Special Case of the $3\times n$ Flow Shop Problem. *Naval Research Logistics Quarterly*. Vol. 22 (1975), pp. 811-817.
3. Conway, R. W., Maxwell, W. L., and Miller, L. W.: *Theory of Scheduling*, Addison-Wesley, Reading Mass., 1967.
4. Johnson, S. M.: Optimal Two- and Three-Stage Production Schedules with Setup Times Included. *Naval Research Logistics Quarterly*, Vol. 1 (1954), pp. 61-68.
5. Kurisu, T.: Two-Machine Scheduling under Required Precedence among Jobs. *Journal of the Operations Research Society of Japan*, Vol. 19 (1976), pp. 1-13.
6. Kurisu, T.: Two-Machine Scheduling under Arbitrary Precedence Constraints. *Journal of the Operations Resaerch Society of Japan*, Vol. 20 (1977), pp. 113-131.
7. Smith, M. L., Panwalker, S. S., and Dudek, R. A.: Flow-Shop Sequencing Problem with Ordered Processing Time Matrices: A General Case. *Naval Research Logistics Quarterly*, Vol. 23 (1976), pp. 481-486.
8. Szwarc, W.: Mathematical Aspects of the $3\times n$ Job-Shop Sequencing Problem. *Naval Research Logistics Quarterly*, Vol. 21 (1974), pp. 145-153.

Tadashi KURISU: Department of Applied Physics, Faculty of Engineering Osaka University, Suita, Osaka, Japan