Determination of Operation Sequences with Minimum Tool-changes in Mechanical Machining*

Amnuay SAE-TANG**, Masanori IGOSHI***,
Shigeru AOMURA*** and Eiji ARAI****

In the manufacturing process of a mechanical part, many operations with alternatives can be generated. A method for selecting optimum operations from the alternatives, and then sequencing them in order to satisfy some objectives such as the lowest cost or minimum time, is required. This paper proposes a new method for determining operation sequences that minimize the number of tool-changes. First, sub-sets of operations, in each of which operations can be consecutively executed by one tool are determined from the information on tools, and the precedence constraints of operations. Based on the sub-sets of the operations, an integer programming problem is introduced next to determine the operation sequences that minimize the number of tool-changes. Finally, the effectiveness of the method is clarified through a case study.

Key Words: Computer Aided Manufacturing, Process Planning, Operation Planning, Operation Sequencing, Integer Programming, Graph Theory

1. Introduction

The main purpose of process planning systems is to determine the manufacturing instructions from the design specifications of a product or part. Generally, the systems start with the extraction of machining features such as slots, holes from geometry data of the parts. Then, the systems determine operations including information of machines, tools, machining parameters for machining the features. The number of the generated operations, however, is not always equal to that of the machining features. A considerable number of operations can be generated for one part because many operations are possibly generated for

one machining feature, and alternative operations can be generated with the change of cutting tools. Therefore, a method for determining the operation sequences is required for the process planning. The purpose of the method is to select operations from the alternatives, and sequence them according to their precedence constraints in order to satisfy particular objectives such as the minimum time or lowest cost.

Due to the difficulty estimating the cost for each operation, most research consider time as their objective function⁽¹⁾⁻⁽⁴⁾. For example, Mittal and Lewis⁽¹⁾ proposed a method based on integer programming to find an optimum operation sequence minimizing the machining time. Since this method considers operations (with alternatives) and their precedence as variables and constraints of the integer programming problem respectively, the number of constraint equations generated by the method is very large in comparison with that of operations. This causes the method to be time consuming for calculation. Rho et al.⁽²⁾ proposed a method using the Left & Right movement method to select an optimum operation sequence in order to minimize the number of required tools and tool-changes. The solution obtained from the method

* Received 1st March, 1996

** Graduate School of Tokyo Metropolitan University, 1-1 Minami-Osawa, Hachioji-shi, Tokyo 192-03, Japan

*** Department of Precision Engineering, Tokyo Metropolitan University, 1-1 Minami-Osawa, Hachioji-shi, Tokyo 192-03, Japan. E-mail: aomura-shigeru@c.metro-u.ac.jp

**** Department of Welding and Production Engineering, Osaka University, 2-1 Yamada-Oka, Suita-shi, Osaka 565, Japan

can reduce the non-cutting time with a minimum number of tool-changes. However, the method has a tendency to be time-consuming and the optimum operation sequence is not always obtainable.

The purpose of this paper is to propose a new method to find the optimum solution of operation sequences for a machining part to be machined on one setup. The method uses graph theory and integer programming techniques to solve the problem while taking the precedence relations between operations as constraints and considering the execution of each individual operation with alternative cutting tools. Even though the method concerns only the operations of a single setup, the sequence of all operations of a machining part can be determined by the method when a multifunctional machine such as a horizontal machining center is used to machine the part. The machine can execute all operations of a single part with only one required setup^{(1),(3)}.

In the remaining of this paper, a method for grouping operations which can be consecutively executed by the same tool is first proposed in Section 2. In Section 3, the term conflict is introduced for any two different sets of operations obtained in Section 2 when the precedence constraint between them cannot be determined. A method to determine the new sets of the operations from the two conflicting ones is explained. In Section 4, a method for determining the optimum operation sequences using the integer programming method, in which all sets of operations obtained from Sections 2 and 3 are variables and the minimum tool-changes is the objective function, is explained. After that the effectiveness of the method is clarified through a case study. Finally, discussions and conclusions are presented.

2. Operations Grouping

For any two operations op_i and op_j , if op_i has to be executed before op_j , the precedence constraint between these two operations can be represented by the following binomial relation.

$$(op_i \text{ } op_j) \quad (1)$$

Let $OP = \{op_1, op_2, \dots, op_n\}$ be a set of n operations. We can represent the precedence constraints among the operations in OP by a matrix or a directed graph called a precedence matrix and a precedence graph, respectively. Figure 1 shows a precedence matrix P and its corresponding directed graph representing the precedence constraints among 10 operations. The value of any element p_{ij} in the matrix is set to 1 if $(op_i \text{ } op_j)$ exists, and to 0 if otherwise.

In the following, we assume that all precedence constraints are transitive. Namely, if $(op_i \text{ } op_j)$ and $(op_j \text{ } op_k)$ exist, the following Eq.(2) is satisfied.

Fig. 1 Example of precedence constraints of 10 operations represented by matrix and directed graph

Fig. 2 Example of transitive precedence matrix and its directed graph where $\alpha=9$

$$(op_i \text{ } op_j) \wedge (op_j \text{ } op_k) \rightarrow (op_i \text{ } op_k) \quad (2)$$

Such a constraint $(op_i \text{ } op_k)$, we refer to as a transitive constraint. This assumption is valid for many practical cases. Since the matrix in Fig. 1 shows only precedence constraints between any two adjacent operations, we refer to it as a non-transitive precedence matrix. In order to obtain the transitive precedence constraints among operations from any given non-transitive precedence matrix, the procedure 1 based on the graph theory⁽⁵⁾ is used.

(Procedure 1: Transformation of non-transitive precedence constraints into transitive ones)

(step 1) : Set $\alpha \leftarrow 2$, $S \leftarrow P$ where P is a non-transitive precedence matrix of n operations. (Initialization)

(step 2) : Calculate S^α .

(step 3) : If $s_{ij}^\alpha = 1$, then set $p_{ij} \leftarrow 1$. s_{ij}^α and p_{ij} are the elements of the matrix S^α and P , respectively. If $\alpha = n-1$, then exit. Otherwise, set $\alpha \leftarrow \alpha + 1$, and go back to (step 2).

By applying the above procedure to the precedence matrix in Fig. 1, a matrix representing transitive precedence constraints among operations is obtained as shown in Fig. 2 with its corresponding directed graph. This matrix is referred to here as a transitive precedence matrix and represented by P^α where P is an original non-transitive precedence matrix. Operations that have to be executed before and after any operation op_k can be determined from

non-zero value of elements in the k -th column and row of the matrix P^a , respectively. For example, operations that have to be executed before and after an operation op_3 in Fig. 2 are op_1, op_2, op_9 and op_4, op_5, op_8 , respectively.

Next, allow us to explain about a tool matrix for a set of m tools, $T=\{t_1, t_2, \dots, t_m\}$. The matrix is used to represent the relation between tools and operations. It has a set of tools and a set of operations lying along its vertical and horizontal axes respectively. For any element q_{ij} of the tool matrix Q , $q_{ij}=1$ if the tool t_i can be used to execute the operation op_j , and $q_{ij}=0$ if otherwise.

Alternative tools for any operation op_j can be determined from non-zero value of elements in the j -th column of the matrix Q while operations executable by the tool t_i can be determined from non-zero value of elements in the i -th row of the matrix Q . Figure 3 shows a tool matrix Q of 7 tools available for the operations shown in Fig. 1. From this matrix Q , for example, alternative tools for the operation op_3 are t_1 and t_2 while operations executable by the tool t_1 are op_1 and op_3 .

Many operations can be executed by one tool but it does not mean that the tool can be consecutively used to execute all of the operations at one time. The same tool may be used many times. This means that the number of tool-changes is also increased. From the above transitive precedence matrix P^a and the tool matrix Q , sets of operations that can be consecutively executed by any tool t_i are determined by the following procedure.

(Procedure 2: Grouping of operations consecutively executable by any tool t_k)

		operations										
		op	1	2	3	4	5	6	7	8	9	10
$Q =$	t	1	1	0	1	0	0	0	0	0	0	0
	t	2	0	0	1	1	1	1	0	0	0	0
	o	3	0	0	0	0	0	1	0	0	1	0
	o	4	0	0	0	0	0	0	1	1	0	1
	$/$	5	0	1	0	0	0	0	1	0	1	1
	s	6	0	0	0	1	1	1	0	0	1	0
	s	7	0	0	0	0	0	0	0	1	1	0

Fig. 3 Example of tool matrix for 7 tools and 10 operations

{1 } ₁	{3 } ₁	{3 6 } ₂	{4 5 } ₂
{6 9 } ₃	{7 8 10 } ₄	{2 7 9 10 } ₅	
{6 9 } ₆	{4 5 } ₆	{9 } ₇	{8 } ₇

Fig. 4 Consecutively executable operation sets

(step 1) : Let $\widehat{op}_k = \{op_1, op_2, \dots, op_n\}$ where $\widehat{op}_k \subseteq OP$ be a set of n operations executable by a given tool t_k , and let $R_i[P^a]$ and $C_i[P^a]$ be a row and column vector generated from the i -th row and column of P^a , respectively.

(step 2) : Determine a row vector A and a column vector B from

$$A = \bigcup_{op_i \in \widehat{op}_k} R_i[P^a] \text{ and } B = \bigcup_{op_i \in \widehat{op}_k} C_i[P^a] \quad (3)$$

where $\cup$ is a disjunction Boolean operation⁽⁶⁾ while $\bigcup_{op_i \in \widehat{op}_k} R_i[P^a] = R_1[P^a] \cup R_2[P^a] \cup \dots \cup R_n[P^a]$ and $\bigcup_{op_i \in \widehat{op}_k} C_i[P^a] = C_1[P^a] \cup C_2[P^a] \cup \dots \cup C_n[P^a]$ denote the disjunction results of all $R_i[P^a]$ and $C_i[P^a]$ determined from operations in $\widehat{op}_k$, respectively. Moreover, when $U = [u_1 \ u_2 \ \dots \ u_n]$ and $V = [v_1 \ v_2 \ \dots \ v_n]$ in which $u_1, \dots, u_n$ and $v_1, \dots, v_n$ are logical variables, $U \cup V = [(u_1 \cup v_1)(u_2 \cup v_2) \dots (u_n \cup v_n)]$.

(step 3) : In both A and B , set $a_i \leftarrow 0$ and $b_i \leftarrow 0$ if an operation $op_i \in \widehat{op}_k$.

(step 4) : Determine a row vector D from

$$D = (A \cap B^T) \quad (4)$$

where $\cap$ is a conjunction Boolean operation⁽⁶⁾, and when $U = [u_1 \ u_2 \ \dots \ u_n]$ and $V = [v_1 \ v_2 \ \dots \ v_n]$ in which $u_1, \dots, u_n$ and $v_1, \dots, v_n$ are logical variables, $U \cap V = [(u_1 \cap v_1)(u_2 \cap v_2) \dots (u_n \cap v_n)]$.

If $D=0$, it means that no other operation that is executed by a different tool t_j ($j \neq k$) exists between any two operations in $\widehat{op}_k$. So we can say that all operations in $\widehat{op}_k$ are consecutively executable by the tool t_k . In this case, $\widehat{op}_k$ is said to be a consecutively executable operation set of the tool t_k . Then, the procedure is finished. If $D \neq 0$, go to (step 5).

(step 5) : Find a sub-set $\widetilde{op}$ of $\widehat{op}_k$, whose each element operation op_i satisfies the Eq.(5).

$$(D \cdot C_i[P^a]) = 0 \quad (5)$$

The $\widetilde{op}$ can be considered as the same with $\widehat{op}$ when $D=0$ in (step 4). So we can consider the sub-set $\widetilde{op}$ as a consecutively executable operation set of the tool t_k . Then, set $\widehat{op} \leftarrow \widehat{op} - \widetilde{op}$, and go back to (step 2).

By applying the above procedure to each of operation sets executable by the tool set T of the tool matrix Q shown in Fig. 3, several consecutively executable operation sets are obtained as shown in Fig. 4. The subscript of each brace shows the tool number used to execute the operation set.

3. Conflict between Consecutively Executable Operation Sets

In the previous section, we show that only the precedence relations among operations in each operation set executable with any given tool t_i are considered to determine consecutively executable operation

sets from the operation set. Because of this reason, when one operation set executable with any given tool t_i is split into two or more consecutively executable operation sets, the precedence relations among the consecutively executable operation sets can be always determined. The precedence relations between two operation sets consecutively executable with different tools, however, may or may not be able to determined.

For any two different consecutively executable operation sets x_i and x_j , x_i has to be executed before x_j if some operations in x_i have to be executed before those of x_j , and none of operations in x_j has to be executed before that of x_i . The precedence constraint between them can also be represented by the binomial relation. However, if the second statement of the above condition is false, that is to say, if some operations in x_j also have to be executed before those of x_i , it is not possible to determine the precedence constraint between x_i and x_j . In this case, x_i and x_j are said to be conflicting with each other. We can represent the conflict between x_i and x_j by Eq.(6)

$$(op_u \ op_s) \wedge (op_r \ op_v) \quad (6)$$

where $op_u, op_v \in x_i$ and $op_r, op_s \in x_j$.

op_v and op_s are referred to here as conflicting operations of x_i and x_j , respectively. For example, two consecutively executable operation sets $\{2 \ 7 \ 9 \ 10\}_5$ and $\{3 \ 6\}_2$ in Fig. 4 are said to be conflicting with each other because $(op_6 \ op_{10})$ and $(op_2 \ op_3)$ exist as shown in Fig. 2. So if we have to execute $\{2 \ 7 \ 9 \ 10\}_5$ and $\{3 \ 6\}_2$ together, the conflict between them will not allow us to execute them separately with only two of tool-changes (each for the tools t_5 and t_2). At least three of tool-changes are needed for this case, for example, each of which for $\{6\}_2$, $\{2 \ 7 \ 9 \ 10\}_5$ and $\{3\}_2$.

Let G_i be an n -dimensional row vector of x_i , where $g_k=1$ if $(op_k \in x_i) \wedge (op_k \notin x_j)$ and $g_k=0$ if otherwise. If two consecutively executable operation sets x_i and x_j satisfy the Eq.(7), they are conflicting with each other.

$$(G_i \cdot \bigcup_{op_r \in x_j} (C_r[P^a]) \neq 0) \wedge (G_j \cdot \bigcup_{op_u \in x_i} (C_u[P^a]) \neq 0) \quad (7)$$

Also if any operation op_u in x_i satisfies the Eq. (8), it is a conflicting operation of x_i regarding to x_j . The same equation can also be used to find the conflicting operation of x_j regarding to x_i .

$$(G_j \cdot (C_u[P^a]) \neq 0) \quad (8)$$

If $\overline{op}^i$ and $\overline{op}^j$ are two sets of conflicting operations of x_i and x_j respectively, we can find four new consecutively executable operation sets without the conflict from x_i and x_j by using Eq.(9).

$$x_{i1} = x_i - \overline{op}^i, \ x_{i2} = \overline{op}^i \text{ and } x_{j1} = x_j - \overline{op}^j, \ x_{j2} = \overline{op}^j \quad (9)$$

For example, $\{10\}_5$ and $\{3\}_2$ are obtained as sets of

conflicting operations from two conflicting sets $\{2 \ 7 \ 9 \ 10\}_5$ and $\{3 \ 6\}_2$ respectively. Four new consecutively executable operation sets without the conflict can be determined from Eq.(9) as follows.

$$\{2 \ 7 \ 9\}_5, \{10\}_5, \{6\}_2, \{3\}_2$$

As shown in Section 4, all pairs of the conflicting sets x_i and x_j , and their new consecutively executable operation sets are also taken into consideration to find optimum operation sequences.

4. Determination of Optimum Operation Sequences

Let $X = \{x_1, x_2, \dots, x_l\}$ be a set of l consecutively executable operation sets obtained from the methods described in Section 2 and 3. An existence matrix E is used to represent the existence of operations in each element of the set X . The matrix has a set of operations and a set of consecutively executable operation sets lying along its vertical and horizontal axes, respectively. The value of an element e_{ij} is set to 1 if the operation op_i exists in x_j , and to 0 if otherwise. Figure 5 shows a matrix E of all consecutively executable operation sets obtained from the matrix P^a in Fig. 2 and the matrix Q in Fig. 3.

With the matrix E , we can determine operation sequences that minimize the number of tool-changes by solving the following integer programming problem.

Objective :

$$\text{minimize } \left(\sum_{j=1}^l z_j \right) \quad (10)$$

Subject to

$$\sum_{j=1}^l e_{ij} z_j \geq 1 \quad (11)$$

for all $op_i \in OP$

$$z_i + z_j \leq 1$$

for all of any two conflicting sets x_i and x_j (12)

$$z_j \in \{0, 1\} \quad (13)$$

where $z_j=1$ if x_j is selected, and $z_j=0$ if otherwise.

		X														
		1 2 3 4 5 6 7 8 9 10 11 12 13 14 15														
$E =$	op	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	o	2	0	0	0	0	0	0	1	0	0	0	0	0	0	1
	p	3	0	1	1	0	0	0	0	0	0	0	1	0	0	0
	e	4	0	0	0	1	0	0	0	0	1	0	0	0	0	0
	r	5	0	0	0	1	0	0	0	0	1	0	0	0	0	0
	a	6	0	0	1	0	1	0	0	1	0	0	0	0	1	0
	t	7	0	0	0	0	0	1	1	0	0	0	0	0	0	1
	o	8	0	0	0	0	0	1	0	0	0	0	1	0	0	0
	n	9	0	0	0	0	1	0	1	1	0	1	0	0	0	1
	s	10	0	0	0	0	0	1	1	0	0	0	0	0	1	0

Fig. 5 Example of existence matrix

The solution of the above integer programming problem is a sub-set of X , in which the number of the consecutively executable operation sets represents the minimum number of the required tool-changes. Constraint (11) ensures that all operations to be executed must exist in the solution. As described in Section 3, if any two conflicting operation sets are selected in the same solution of operation sequence, the number of required tool-changes will be increased (The number of required tool-changes will not equal to the number of the selected consecutively executable operation sets). Since conflicting operation sets and their new operation sets without the conflict are also considered as variables in the integer programming problem, the constraint (12) is used to ensure that no two conflicting operation sets x_i and x_j appears in the same solution. Since the value of the variables is limited to be only 0 or 1 as shown in constraint (13), we consider this integer programming problem as a combinatorial optimization problem and solve it by using the additive algorithm⁽⁷⁾.

From the matrix E shown in Fig. 5, an integer programming problem is formed as shown in Fig. 6. By solving the problem, two solutions are obtained as shown in Fig. 7. The solutions show that five of tool-changes are required for the execution of all operations.

After the solutions are obtained, the sequence among operations in each solution is determined. This operation sequencing can be separately done for x_j s in each solution and for operations in each x_j . These two sequencing procedures are explained below.

(Procedure 3: Sequencing of x_j s in each solution)

(step 1) : Let In and Out be a set of x_j s minimizing

$$\begin{array}{c} E \\ \left[\begin{array}{cccccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{array} \right] \\ Z \\ \left[\begin{array}{c} z_1 \\ z_2 \\ z_3 \\ z_4 \\ z_5 \\ z_6 \\ z_7 \\ z_8 \\ z_9 \\ z_{10} \\ z_{11} \\ z_{12} \\ z_{13} \\ z_{14} \\ z_{15} \end{array} \right] \end{array} \geq 1$$

$$z_3 + z_7 \leq 1$$

$$Z_3 = \{3, 6\} \quad Z_7 = \{2, 7, 9, 10\}$$

Fig. 6 Integer programming problem

the number of tool-changes, and a null order set respectively.

(step 2) : Find a column vector

$$A_j = \bigcup_{op_i \in x_j} (C_i[P^a]) \quad (14)$$

for each x_j in In .

(step 3) : For each A_j , set $a_k \leftarrow 0$ if an operation $op_k \in x_j$.

(step 4) : Find A_j that is a zero vector, and then set $Out \leftarrow Out + \{x_j\}$, $In \leftarrow In - \{x_j\}$ and $Y \leftarrow x_j$.

(step 5) : If $In = 0$, stop the procedure and use Out as a solution of the procedure. Otherwise go to (step 6).

(step 6) : For each of the remaining A_j , set $a_k \leftarrow 0$ if an operation $op_k \in Y$. Then, go back to (step 4).

After the sequence among x_j s minimizing the number of tool-changes is obtained, the sequence among operations in each x_j is determined.

(Procedure 4: Sequencing of operations in each x_j)

(step 1) : Let In and Out be a set of operations in x_j , and a null order set respectively.

(step 2) : For $C_i[P^a]$ of each operation op_i in x_j , set $c_k \leftarrow 0$ if an operation $op_k \notin x_j$.

(step 3) : Find $C_i[P^a]$ which is a zero vector, and set $Out \leftarrow Out + \{op_i\}$, $In \leftarrow In - \{op_i\}$ and $Y \leftarrow \{op_i\}$.

(step 4) : If $In = \phi$, stop the procedure and use Out as a solution of the procedure. Otherwise go to (step 5).

(step 5) : For each of the remaining $C_i[P^a]$, set $c_k \leftarrow 0$ if an operation $op_k \in Y$. Then, go back to (step 3).

By applying the procedure 3 and 4 to the solutions shown in Fig. 7, the sequence among operations in each solution can be obtained as shown in Fig. 8.

In one solution of the operation sequence, the same operation may appear for several times. For example, the operation op_7 appears twice in each of the operation sequences in Fig. 8. This means that two different tools can be used to execute the operation. In this case, only one tool has to be assigned for

$$\begin{array}{l} (1) \quad \{1\}_1 \quad \{3, 6\}_2 \quad \{7, 8, 10\}_4 \quad \{4, 5\}_2 \quad \{2, 7, 9\}_5 \\ (2) \quad \{1\}_1 \quad \{3, 6\}_2 \quad \{7, 8, 10\}_4 \quad \{4, 5\}_6 \quad \{2, 7, 9\}_5 \end{array}$$

Fig. 7 Two solutions of operation sequences obtained from integer programming shown in Fig. 6

$$\begin{array}{l} (1) \quad \{1\}_1 \quad \{9, 2, 7\}_5 \quad \{3, 6\}_2 \quad \{7, 10, 8\}_4 \quad \{4, 5\}_2 \\ (2) \quad \{1\}_1 \quad \{9, 2, 7\}_5 \quad \{3, 6\}_2 \quad \{7, 10, 8\}_4 \quad \{4, 5\}_6 \end{array}$$

Fig. 8 Operation sequences after the exact sequence is determined

centers⁽³⁾. However, there still exist many other important objectives such as the lowest cost or the shortest cutting time. The work on the operation sequencing taking many objectives into account will remain for future research.

References

- (1) Mittal, R.O. and Lewis, R.L., A MICRO Process Planning System Based on Integer Programming for Prismatic Part Produced on Horizontal Machining Centers, *Annals of Operation Research*, Vol. 17 (1989), pp. 273-290.
- (2) Rho, H.M., Van't Erve, A.H. and Kals, H.J.J., An Integrated Cutting Tool Selection and Operation Sequencing Method, *Annals of the CIRP*, Vol. 41, No. 1 (1992), pp. 517-520.
- (3) Agapiou, J.S., Sequence of Operations Optimization in Single-Stage Multifunctional Systems, *Journal of Manufacturing Systems*, Vol. 10, No. 3 (1991), pp. 194-208.
- (4) Shan, X.H. and Nee, A.Y.C. and Poo, A.N., Integrated Application of Expert Systems and Neural Networks for Machining Operation Sequencing, *ASME Neural Networks in Manufacturing and Robotics*, Vol. 57 (1992), pp. 117-126.
- (5) Onodeji, R., Extension and Application of Graph Theory, (in Japanese), (1973), Morikita Shuppan Publishing.
- (6) Takenaka, Y., Graph and Boolean Algebra, (in Japanese), (1981), Kyoritsu Shuppan Co., Ltd.
- (7) Kose, T., Mathematical Programming Part 1, (in Japanese), (1971), Kyoritsu Shuppan Co., Ltd.
- (8) Hisamune, M., Hanada, T. and Hoshi, S., Automatic Process Planning and Tool Selection for Block Parts, *Proceeding of JSPE*, (in Japanese), Spring, (1994), pp. 293-294.
- (9) Kim, I.T., Suh, H.W. and Kim, H.R., Optimum Operation Grouping and Sequencing Technique For Multistage Machining Systems, *Proceeding of the Computers in Engineering Conference and the Engineering Database Symposium ASME*, (1995), pp. 633-645.