Journal of the Operations Research
Society of Japan
Vol. 27, No. 3, September 1984

OPTIMAL TOOL MODULE DESIGN PROBLEM FOR NC MACHINE TOOLS

Ryuichi Hirabayashi
Tokyo Science University

Hisatoshi Suzuki
Tokyo Institute of Technology

Noboru Tsuchiya
Hitachi Production Engineering Research Laboratory

(Received September 21, 1983: Final March 15, 1984)

Abstract In order to enhance factory automation and unmanned production, numerical controlled (NC) machine tools are widely used in many mechanical processing factories. Whenever one machining part is changed to another, the operation of an NC machine tool has to be stopped in order to change some of the cutting tools on the turret tool holder. In this paper, a new concept of 'tool module' is used and the problem of reducing the number of tool changing operations for NC machine tools is discussed. The problem is formulated as a *0–1 integer programming* (0–1 ILP) problem on a bipartite graph and a branch and bound algorithm is proposed to select an optimal tool module. Since the LP relaxation problem obtained by dropping the integrality condition for the 0–1 ILP problem has a special network structure like the well-known transportation problem, this LP is solved by certain network flow algorithm with utilizing its special structure Some numerical experiments are reported, and indicate that the algorithm is reasonably efficient. Furthermore, possible extensions of this method are discussed.

1. Introduction

Recently, a lot of numerical controlled machine tools (briefly, NC machine tools), such as machining centers and turret punch presses, have been introduced in many mechanical processing factories. Nowadays, NC machine tools are indispensable to high quality and high efficient production. Since a worker can handle several NC machine tools at the same time, it is possible to reduce man-hours and lead-times greatly. Thus, NC machine tools are useful to achieve the factory automation as well as the unmanned production, which in turn increases the productivity. These bring an innovation in the aspect of factory management.

However, the machine set-up operations which include the changing

of cutting-tools play an important part in the productivity improvement. Hence, our main attention is paid to the tool changing operations and reduce the number as much as possible. In this paper, this problem will be treated as a mathematical programming problem.

In the succeeding sections of this paper, mainly the machining center is considered. However, the similar procedure is adaptable for many other types of NC machine tools. At a machining center simultaneously several cutting-tools can be installed such as face cutters, drills, endmills, taps and so on, with the range from 12 to 80 for each NC machine tool (the maximum size of tool magazine is a constant for each NC machine tool). Usually, for processing one part, only 40 - 60% of installable tools are used. Moreover, some of these tools are common for processing different kinds of parts. Hence, we consider a production method based on the group technology concept. In this method, all parts are partitioned into several groups, and each group is called a "parts-family". The parts contained in each family may require a lot of processing tools in common and can be processed by the same set of tools within the maximum size of tool magazine. We call this set of tools as a "tool module", and when a parts-family changes, whole set of installed tools i.e., a tool module is changed. Therefore, the number of set-up operations is the same as the number of parts-families (i.e., the least number of tool modules needed for processing all parts). Thus, if we construct some suitable tool modules, the total set-up operations will reduce considerably.

We denote by $M = \{1,2,\cdots,m\}$ the set of tools, by $N = \{1,2,\cdots,n\}$ the set of parts to be processed and by $E \in M \times N$ the relation between tools and parts, i.e., $(i,j) \in E$ means that the tool i is used to process the part j. Let a positive integer k be the maximum size of tool magazine. Then, the problem mentioned above can be restated as follows; cover N by the least number of parts families each of which can be processed by one tool module with at most k tools. We call this problem as the *optimal parts grouping problem* (OPGP).

The OPGP was first studied in [5]. If $I(j)$ be a set of tools necessary for processing the part j and $|I(j)|$ be the cardinal number of $I(j)$, the Hamming distance d_{ij} between $i \in N$ and $j \in N$ is defined by $d_{ij} = |I(i) \oplus I(j)| = |I(i)| + |I(j)| - 2|I(i) \cap I(j)|$. In [5], using this Hamming distance as a metric, several heuristic methods for parts grouping methods were proposed, namely, if d_{ij} is large, then i and j should belong to different families. Conversely, if d_{ij} is small, then i and j

should belong to the same family. In this situation, each parts-family should require at most k tools for processing.

In this paper, given a profit π_j (nonnegative) for each part j ($j = 1,2,\cdots,n$), we consider a problem to construct an optimal tool module that maximizes the profit. We call this problem the *optimal tool module design problem* (OTMP). Initially, this is considered as a problem on a bipartite graph and subsequently converted to a 0-1 ILP problem. Let $\pi_j = 1$ for every $j \in N$, then the OTMP is to find a tool module which maximizes the number of parts to be processed by it. Let π_j be the processing time of part j, then the problem is to find a tool module so as to maximize its continuous utilization during processing. The latter problem is very important for unmanned night shift production.

So far we have discussed the OTMP instead of analyzing the OPGP directly. By the following two reasons, it can be shown that the former consitutes a subproblem of the latter. If we can obtain an optimal tool module for given M and N, then by deleting from N the parts which can be processed by the tool module, the next OTMP for M and the remaining N is determined. Repeating this procedure until N becomes empty, we can get a sub-optimal solution for the OPGP. The second reason is that the OPGP can be formulated as a set covering problem and that the OTMP is useful to generate coefficient column vectors of the set covering problem. More details are stated in the last section of this paper. Thus, it is valuable to study the OTMP to solve the OPGP as the first step.

2. Description of the Problem

Given a bipartite graph (M,N,E), where $M = \{1,2,\cdots,m\}$, $N = \{1,2,\cdots,n\}$ and $E \subset M \times N$, we define $I(j) = \{i \in M \mid (i,j) \in E\}$ for each $j \in N$. Let k be a given positive integer and π_j ($j \in N$) be nonnegative constants. Then we consider the following problem (see Figure 1):

$$(P_0) \qquad \text{maximize} \sum_{I(j) \subset S} \pi_j$$

subject to

$$S \subset M,$$
$$|S| \leq k.$$

Reading M as the set of tools, as N the set of parts, E as the binary relation $E = \{(i,j) \in M \times N \mid$ the processing part j needs the tool $i\}$, k as the maximum size of tool magazine for a machining-center

and π_j as the profit for processing the part j, (P_0) becomes the OTMP explained in the section 1. Since $m \simeq 50,\cdots, 400$, $n \simeq 50,\cdots,200$, $k = 12,16,20,30,\cdots,80$ in the real OTMP in hand, we consider to reduce the size of (P_0) before solving it.

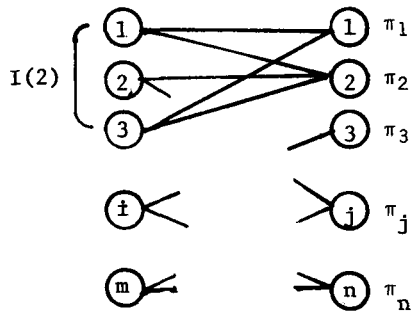


Figure 1. Optimal Tool Module Design Problem

Reducing Rule

(1) If $|I(j)| > k$ then let $N = N\backslash\{j\}$ and $E = E \cap (M \times N)$.

(2) For some $i \in M$, if $(i,j) \notin E$ holds for any $j \in N$, then let $M = M\backslash\{i\}$.

(3) If $|M| = m \leq k$ then let $S = M$ and (P_0) is solved.

Since the reduced problem has the same structure as the original one, we also denote the reduced problem by (P_0).

In this paper, instead of solving (P_0) directly, we solve it after representing as a 0-1 ILP problem. Note that a set $S \subset M$ can be represented by the m-dimensional 0-1 variable vector $u = (u_i)$ by

$$u_i = \begin{cases} 1 & \text{if } i \in S \\ 0 & \text{if } i \notin S \end{cases}$$

and the n-dimensional 0-1 variable vector $v = (v_j)$ by

$$v_j = \begin{cases} 1 & \text{if } I(j) \subset S \\ 0 & \text{if } I(j) \not\subset S. \end{cases}$$

Then (P_0) can be rewritten as the following 0-1 integer program (P):

(P) maximize $\sum_{j \in N} \pi_j v_j$

subject to

$$v_j \leq u_i, \qquad (i,j) \in E,$$

$$\sum_{i \in M} u_i \leq k,$$

$$u_i \in \{0,1\}, \quad i \in M,$$
$$v_j \in \{0,1\}, \quad j \in N.$$

In the problem (P), when we determine the value for the vector u, the vector v is determined automatically by $v_j = \min_{(i,j)\in E} u_i$ for all $j \in N$. Therefore we can consider that the essential variable vector is u. Thus it is clear that the constraints $v_j \in \{0,1\}$, $j \in N$ can be omitted.

3. Branch and Bound Method

In this section we will describe a branch and bound algorithm for solving the problem (P). We will denote by SP(ℓ) the subproblem on the ℓ-th node. Let

$$I^0 = \{i \in M \mid u_i \text{ is fixed at 0 in the } \ell\text{-th node}\},$$
$$I^1 = \{i \in M \mid u_i \text{ is fixed at 1 in the } \ell\text{-th node}\},$$
$$I^F = \{i \in M \mid u_i \notin I^0 \cup I^1\}.$$

Also, let

$$J^0 = \bigcup_{i\in I^0} J(i) \cup \{j \in N \mid j \in J^0 \text{ for the parent node}\},$$
$$J^1 = \{j \in N \mid I(j) \subset I^1\} \cup \{j \in N \mid j \in J^1 \text{ for the parent node}\},$$
$$J^F = \{j \in N \mid j \notin J^0 \cup J^1\}.$$

Here, $J(i) = \{j \in N \mid (i,j) \in E\}$. Since $v_j = \min_{i\in I(j)} u_i$, there exists an optimal solution of SP(ℓ) satisfying

$$v_j = \begin{cases} 0 & \text{if } j \in J^0 \\ 1 & \text{if } j \in J^1. \end{cases}$$

Hence, letting $E^\ell = E \cap \{(i,j) \mid i \in I^F, j \in J^F\}$, the subproblem SP($\ell$) is:

SP(ℓ) maximize $\sum_{j\in J^F} \pi_j v_j + \sum_{j\in J^0} \pi_j$

subject to

$$v_j \le u_i, \quad (i,j) \in E^\ell,$$
$$\sum_{i\in I^F} u_i \le k - |I^1|,$$

$$u_i \in \{0,1\}, \quad i \in I^F.$$

Also, we will denote by $\overline{SP}(\ell)$ the LP relaxation problem of $SP(\ell)$ and by $\bar{z}(\ell)$ the objective function value of $\overline{SP}(\ell)$. The algorithm for solving $\overline{SP}(\ell)$ is described in the section 5.

We will consider (P_0) to get an initial feasible solution. Without loss of generality, we can assume $\pi_1 \geq \pi_2 \geq \cdots \geq \pi_n$. Then, let $S = I(1)$ first, and, for $\ell = 2,3,\cdots,n$ in order, let $S = S \cup I(\ell)$ if $|S \cup I(\ell)| \leq k$. The resulting S is a greedy solution for (P_0). There are some other heuristics to find initial feasible solutions of (P) or (P_0). Clearly, $u_1=u_2=\cdots=u_k=1$, $u_{k+1}=u_{k+2}=\cdots=u_n=0$ is a trivial initial feasible solution. Another heuristic to find a feasible solution is as follows. Select some $j \in N$ and let $S = I(j)$. The set $I(\ell)$, $\ell \in N$, with the nearest Hamming distance to S has priority to join with S (i.e., S is replaced with $S \cup I(\ell)$) and repeat this process as long as $|S \cup I(\ell)| \leq k$.

Branch and Bound Algorithm.

(step 1) Find an initial feasible solution of (P) and obtain z^I the corresponding value of the objective function. Let $\ell = 1$ and all u_i's, v_j's be undetermined, i.e., $I^F = M$ (J^F becomes N) and SP(1) is equal to (P). Let the active node set $\mathcal{N} = \emptyset$. Reduce SP(1) by the following reducing rule.

Reducing Rule

(1) For every $j \in J^F$ such that $|I(j) \cap I^F| > k - |I^1|$, let $v_j = 0$, $J^0 = J^0 \cup \{j\}$, $J^F = J^F \setminus \{j\}$ and $E^\ell = E^\ell \setminus (I(j) \times \{j\})$.

(2) For every $j \in J^F$ such that $I(j) \cap I^F = \emptyset$, let $v_j = 1$, $J^1 = J^1 \cup \{j\}$ and $J^F = J^F \setminus \{j\}$.

(3) For every $i \in I^F$ such that $J(i) \cap J^F = \emptyset$, let $u_i = 0$, $I^0 = I^0 \cup \{i\}$ and $I^F = I^F \setminus \{i\}$.

(4) For every $i \in I^F$ such that $J^F \subset J(i)$, let $u_i = 1$, $I^1 = I^1 \cup \{i\}$, $I^F = I^F \setminus \{i\}$ and $E^\ell = E^\ell \setminus (\{i\} \times J(i))$.

(5) For every $j \in J^F$ such that $|I(j) \cap I^F| = k - |I^1|$, let $S = I(j) \cap I^F$ and set $z(\ell) = \sum_{I(p) \cap I^F \subset S} \pi_p + \sum_{p \in J^1} \pi_p$. In the case $z(\ell) > z^I$, let $z^I = z(\ell)$. Let $v_j = 0$, $J^0 = J^0 \cup \{j\}$, $J^F = J^F \setminus \{j\}$ and $E^\ell = E^\ell \setminus (I(j) \times \{i\})$.

(6) If $|I^F| \leq k - |I^1|$ then let $u_i = 1$ for any $i \in I^F$, $I^1 = I^1 \cup I^F$, $I^F = \emptyset$, $v_j = 1$ for any $j \in J^F$, $J^1 = J^1 \cup J^F$, $J^F = \emptyset$ and set $\bar{z}(\ell) = z(\ell) = \sum_{j \in J^1} \pi_j$. In this case SP(1) is solved.

Reconstruct the subproblem SP(1) by using the new I^0, I^1, I^F, J^0, J^1, J^F and E^{ℓ}. Solve the relaxation problem $\overline{SP}(1)$ and get $\bar{z}(1)$. If $\bar{z}(1) \leq z^I$ then go to step 6. If $\bar{z}(1) > z^I$ and all u_i's and v_j's are integer values, then let $z^I = \bar{z}(1)$ and go to step 6. Else record all u_i's, v_j's and $\bar{z}(1)$ at the node 1 and let $N = \{1\}$.

(step 2) If $N = \emptyset$ then go to step 6 else select the node p which is lastly generated. If $\bar{z}(p) \leq z^I$ then $N = N \setminus \{p\}$ and return to the top of step 2.

(step 3) Select a u_i which is fractional in the node p. Then, we will fix the variable u_i to $\hat{u}_i$ which is either 0 or 1. If both 0 and 1 are already used to fix u_i then let $N = N \setminus \{p\}$ and go to step 2. If u_i can be selected then fix $u_i = \hat{u}_i$, let $\ell = \ell + 1$, generate the subproblem $\overline{SP}(\ell)$, record it to the node ℓ and let $N = N \cup \{\ell\}$.

(step 4) Reduce the subproblem SP(ℓ) by the reducing rule described in step 1 and reconstruct SP(ℓ) by using the new $I^0, \cdots, J^F$. Solve its relaxation $\overline{SP}(\ell)$ and obtain the objective value $\bar{z}(\ell)$. Record all values of u_i's and v_j's at the node ℓ.

(step 5) In the case of $\bar{z}(\ell) \leq z^I$, let $N = N \setminus \{\ell\}$ and go to step 2. In the case where $\bar{z}(\ell) > z^I$ and there exist a fractional u_i, go to step 2. In the case when $\bar{z}(\ell) > z^I$ and every u_i is integer, let $z^I = \bar{z}(\ell)$, delete all nodes $\ell' \in N$ from N such that $\bar{z}(\ell') \leq z^I$ and go to step 2.

(step 6) The present value z^I is the optimal value for the original problem (P) and the solution which gives the objective value z^I is an optimal solution for (P).

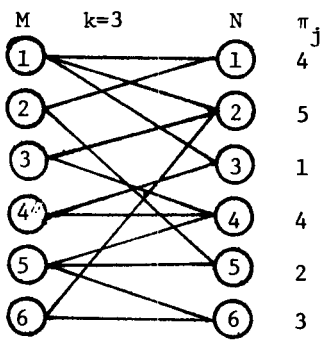


Figure 2. An example of the Optimal Tool Module Design Problem

Consider the OTMP given by Figure 2. The result of the branch and bound method is given by Figure 3. In this example, the initial solution is $S = I(2) = \{1,3,5\}$ which is a greedy solution and $z^I = 5$. At node 1, by the reducing rule 5 described in the branch and bound algorithm, J^0 becomes $\{2,4\}$. Then by the reducing rule 3, I^0 becomes $\{3\}$ and the reduced subproblem SP(1) is illustrated as Figure 4. Solving $\overline{SP}(1)$, we get $\bar{z}(1) = 27/4$, $u = (3/4,3/4,0,0,3/4,3/4)$ and $v = (3/4,0,0,0,3/4,3/4)$.

At node 2, we fix u_1 at 1, then I^1 is $\{1\}$ and $k-|I^1|$ becomes 2. Hence, by reducing rule 5, we have the new integer solution $u = (1,1,0,0,1,0)$, $v = (1,0,0,0,1,0)$ and the objective function value $z = 6$. Since $z > z^I$, we replace z^I by z (= 6). J^0 is now $\{2,4,5\}$. Next, applying the reducing rule 5 again to $6 \in J^F = \{1,3,6\}$, we have $I^1 = \{1\}$, $I^0 = \{3\}$, $J^1 = \emptyset$ and $J^0 = \{2,4,5,6\}$. Now apply the reducing rule 3 and we have $I^1 = \{1\}$, $I^0 = \{3,5,6\}$, $J^1 = \emptyset$ and $J^0 = \{2,4,5,6\}$. Since $|I^F| = |\{2,4\}| = 2 = k - |I^1|$, by reducing rule 6, I^1 becomes $\{1,2,4\}$ and $\bar{z}(2) = z(2) = 5$.

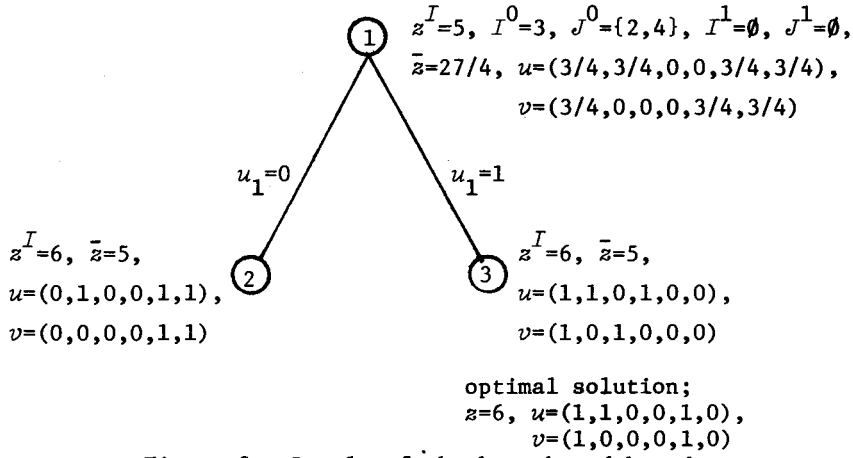


Figure 3. Result of the branch and bound algorithm for the problem in Fig.2

At node 3, since we fix u_1 at 0, $I^0 = \{1,3\}$ and $J^0 = \{1,2,3,4\}$. Then by the reducing rule 3 and 6, we have $I^0 = \{1,3,4\}$, $I^1 = \{2,5,6\}$, $I^F = \emptyset$, $J^0 = \{1,2,3,4\}$, $J^1 = \{5.6\}$, $J^F = \emptyset$ and $\bar{z}(3) = z(3) = 5$.

Since both node 2 and 3 are fathomed, the active nodes set becomes empty and the algorithm terminates. The optimal solution obtained is $u = (1,1,0,0,1,0)$, $v = (1,0,0,0,1,0)$ and the objective functional value is 6.

4. Relaxation of the Problem (P)

Now, let us consider a subproblem of (P) where some u_i's are fixed at either 0 or 1. Since such problem has the same structure as the original one as shown in the section 3, we will study the structure of (P) instead of those of its subproblems.

Consider an LP-relaxation problem ($\bar{P}$) of (P):

($\bar{P}$) Maximize $\sum_{j\in N} \pi_j v_j$

subject to

$$v_j \le u_i \qquad (i,j) \in E,$$
$$\sum_{i\in M} u_i \le k,$$
$$0 \le u_i \le 1, \qquad i \in M,$$
$$0 \le v_j \le 1, \qquad j \in N.$$

As in the problem (P), we can omit the constraints $0 \le v_j \le 1$, $j \in N$ from ($\bar{P}$). On the other hand, let $x = (x_{ij})$ $(i,j) \in E$ be the Lagrangean variable vector for the constraints $v_j \le u_i$, $(i,j) \in E$ and consider the Lagrangean relaxation problem:

(L_x) maximize $\sum_{j\in N} \pi_j v_j - \sum_{(i,j)\in E} x_{ij}(v_j - u_i)$

subject to

$$\sum_{i\in M} u_i \le k,$$
$$u_i \in \{0,1\}, \qquad i \in M,$$
$$v_j \in \{0,1\}, \qquad j \in N.$$

Let us denote by $z(\cdot)$ the maximal objective value of a problem $(\cdot)$. Then $z(P) \le z(L_x)$ for any $x \ge 0$ holds in general, and $\min_{x\ge 0} z(L_x)$ gives the best upper bound of $z(P)$ [1]. We will show that the next theorem holds;

Theorem 1. $\min_{x\ge 0} z(L_x) = z(\bar{P})$.

Proof: $z(L_x) = \max\{ \sum_{j\in N} \pi_j v_j - \sum_{(i,j)\in E} x_{ij}(v_j - u_i) \mid \sum_{i\in M} u_i \le k,$

$u_i \in \{0,1\}$ $(i \in M)$, $v_j \in \{0,1\}$ $j \in N)\}$

$= \max\{ \sum_{j\in N} (\pi_j - \sum_{i\in I(j)} x_{ij}) v_j + \sum_{i\in M} (\sum_{j\in J(i)} x_{ij}) u_i \mid$

$$\sum_{i\in M} u_i \le k,\ u_i \in \{0,1\}\ (i \in M),\ v_j \in \{0,1\}\ (j \in N)\}$$

$$= \max\{\sum_{j\in N}(\pi_j - \sum_{i\in I(j)} x_{ij})v_j + \sum_{i\in M}(\sum_{j\in J(i)} x_{ij})u_i \mid$$

$$\sum_{i\in M} u_i \le k,\ 0 \le u_i \le 1\ (i \in M),\ 0 \le v_j \le 1\ (j \in N)\}$$

$$= \max\{\sum_{j\in N} \pi_j v_j - \sum_{(i,j)\in E} x_{ij}(v_j - u_i) \mid \sum_{i\in M} u_i \le k,$$

$$0 \le u_i \le 1\ (i \in M),\ 0 \le v_j \le 1\ (j \in N)\}$$

$$= z(\bar{L}_x),$$

where $(\bar{L}_x)$ is the Lagrange relaxation problem for $(\bar{P})$. Moreover, since $(\bar{P})$ is an LP problem, $\min_{x\ge 0} z(\bar{L}_x) = z(\bar{P})$. Therefore,

$$\min_{x\ge 0} z(L_x) = \min_{x\ge 0} z(\bar{L}_x) = z(\bar{P}).$$

By this theorem, it is verified that $z(\bar{P})$ gives the best upper bound for $z(P)$ in the sense of Lagrange relaxation. □

Next we will discuss a method solving for the LP-relaxation problem $(\bar{P})$. For solving $(\bar{P})$, let us introduce the dual variable x_{ij} $((i,j) \in E)$, λ, μ_i $(i \in M)$ and construct the dual problem $(\bar{D})$ of the primal problem $(\bar{P})$:

$(\bar{D})$ minimize $k\lambda + \sum_{i\in M} \mu_i$

subject to

$$\sum_{i\in I(j)} x_{ij} = \pi_j, \qquad j \in N,$$

$$\sum_{j\in J(i)} x_{ij} \le \lambda + \mu_i, \qquad i \in M,$$

$$x_{ij} \ge 0, \qquad (i,j) \in E,$$

$$\lambda \ge 0,$$

$$\mu_i \ge 0 \qquad i \in M.$$

For the optimal solution of $(\bar{D})$, the next theorem holds;

Theorem 2. Let (x^*,λ^*,μ^*) be an optimal solution of $(\bar{D})$. Suppose

$$\sum_{j\in J(1)} x_{1j}^* \ge \sum_{j\in J(2)} x_{2j}^* \ge \cdots \ge \sum_{j\in J(m)} x_{mj}^*$$

holds, then

$$\lambda^* = \sum_{j\in J(k)} x_{kj}^*,$$

$$\mu_i^* = \begin{cases} \sum_{j \in J(i)} x_{ij}^* - \lambda^* & \text{if } i \le k \\ 0 & \text{if } i > k \end{cases} \quad \text{for all } i \in M,$$

holds and the optimal value of objective function is given by

$$z(\bar{D}) = \sum_{i=1}^{k} \sum_{j \in J(i)} x_{ij}^* = \sum_{i=1}^{k} (\lambda^* + \mu_i^*).$$

Proof: Let $\alpha_i = \sum_{j \in J(i)} x_{ij}^*$ for every $i \in M$, then (λ^*, μ^*) should be an optimal solution of the next problem $(\hat{D})$:

$$(\hat{D}) \qquad \text{minimize } k\lambda + \sum_{i \in M} \mu_i$$

subject to

$$\lambda + \mu_i \ge \alpha_i, \qquad i \in M,$$

$$\lambda \ge 0,$$

$$\mu_i \ge 0, \qquad i \in M.$$

Let $\lambda^* = \alpha_k$ and, for every $i \in M$,

$$\mu_i^* = \begin{cases} \alpha_i - \alpha_k & \text{if } i \le k \\ 0 & \text{if } i > k \end{cases}$$

then, since $\alpha_1 \ge \alpha_2 \ge \cdots \ge \alpha_n \ge 0$ holds from the assumption of the theorem, (λ^*, μ^*) is a feasible solution of $(\hat{D})$. The dual problem $(\hat{P})$ of the problem $(\hat{D})$ is:

$$(\hat{P}) \qquad \text{maximize } \sum_{i \in M} \alpha_i u_i$$

subject to

$$\sum_{i \in M} u_i \le k,$$

$$0 \le u_i \le 1, \qquad i \in M.$$

By the condition $\alpha_1 \ge \alpha_2 \ge \cdots \ge \alpha_n \ge 0$, it is easily seen that the optimal solution and the optimal value of $(\hat{P})$ are

$$u_i^* = \begin{cases} 1 & \text{if } i \le k \\ 0 & \text{if } i > k \end{cases} \quad \text{for all } i \in M,$$

$$z(\hat{P}) = \sum_{i \in M} \alpha_i u_i^* = \sum_{i=1}^{k} \alpha_i.$$

Since $k\lambda^* + \sum_{i\in M} \mu_i^* = \sum_{i=1}^{k} \alpha_i = z(\hat{P})$, by the duality theorem (λ^*,μ^*) is the optimal solution of (D). □

Corollary 1. Let $\hat{\lambda} = (1/m) \sum_{j\in N} \pi_j$ and $\hat{\mu}_i = 0$ for every $i \in M$. If there exists a feasible flow vector $\hat{x} = (\hat{x}_{ij})$ on the bipartite graph (M,N,E) such that $\sum_{j\in J(i)} \hat{x}_{ij} \le \hat{\lambda}$, $i \in M$ and $\hat{x}_{ij} \ge 0$, $(i,j) \in E$, then $(\hat{x}, \hat{\lambda},\hat{\mu})$ is an optimal solution of the dual problem $(\bar{D})$.

Proof: It can be easily shown that $(\hat{x},\hat{\lambda},\hat{\mu})$ is a feasible solution of $(\bar{D})$ and the objective value is $k\hat{\lambda} + \sum_{i\in M} \hat{\mu}_i = (k/m) \sum_{j\in N} \pi_j$. On the other hand, let (x^*,λ^*,μ^*) be an optimal solution of $(\bar{D})$ and, for simplicity, satisfy the assumption of Theorem 2. Then, by the Theorem 2,

$$\begin{aligned} z(\bar{D}) &= \sum_{i=1}^{k} \sum_{j\in J(i)} x_{ij}^* \\ &\ge (k/m) \sum_{i\in M} \sum_{j\in J(i)} x_{ij}^* \\ &= (k/m) \sum_{j\in N} \sum_{i\in I(j)} x_{ij}^* \\ &= (k/m) \sum_{j\in N} \pi_j . \end{aligned}$$

Hence, $(\hat{x},\hat{\lambda},\hat{\mu})$ is a better feasible solution than (x^*,λ^*,μ^*). Therefore $(\hat{x},\hat{\lambda},\hat{\mu})$ is an optimal solution of $(\bar{D})$. □

5. Algorithm for solving $(\bar{D})$

In this section we will describe a primal-dual algorithm (see, for example [4]) for solving $(\bar{D})$. Adding two nodes u_0 and v_0, $(\bar{P})$ can be reformulated as the following problem $(\bar{P}')$:

$$\begin{aligned} (\bar{P}') \qquad & \text{maximize} && \sum_{j\in N} \pi_j (v_j - v_0) \\ & \text{subject to} \\ & && \sum_{i\in M} (u_0 - u_i) \ge m - k, \\ & && u_i - v_j \ge 0, && (i,j) \in E, \\ & && u_0 - u_i \ge 0, && i \in M, \\ & && v_j - v_0 \ge 0, && j \in N, \\ & && u_0 = 1,\ v_0 = 0. \end{aligned}$$

($\bar{P}'$) can be regarded as a *project scheduling problem* with an additional linear constraint and can be solved by a general solving method developed in [2,3]. There, k is treated as a parameter and ($\bar{P}'$) is solved parametrically from $k = m-1$ to $k = k$. The algorithms developed in [2,3] are for general network programming, however, ($\bar{P}$) has a more simple structure as a bipartite graph. Therefore, we develop a simpler primal-dual algorithm for ($\bar{P}$) and ($\bar{D}$) in which k is regarded as a fixed scalar.

In order to solve ($\bar{D}$) by using the primal-dual algorithm, it is enough to find a solution (u,v,x,λ,μ) satisfying the primal feasibility;

$$v_j - u_i \le 0, \qquad (i,j) \in E, \tag{5.1}$$

$$\sum_{i\in M} u_i \le k, \tag{5.2}$$

$$0 \le u_i \le 1, \qquad i \in M, \tag{5.3}$$

the dual feasibility;

$$\sum_{i\in I(j)} x_{ij} = \pi_j, \qquad j \in N, \tag{5.4}$$

$$\sum_{j\in J(i)} x_{ij} \le \lambda + \mu_i, \qquad i \in M, \tag{5.5}$$

$$x_{ij} \ge 0, \qquad (i,j) \in E, \tag{5.6}$$

$$\lambda \ge 0. \tag{5.7}$$

$$\mu_i \ge 0, \qquad i \in M, \tag{5.8}$$

and the complementarity condition;

$$(v_j - u_i)x_{ij} = 0 \qquad (i,j) \in E, \tag{5.9}$$

$$(\sum_{i\in M} u_i - k)\lambda = 0, \tag{5,10}$$

$$(u_i - 1)\mu_i = 0 \qquad i \in M, \tag{5.11}$$

$$(\sum_{j\in J(i)} x_{ij} - (\lambda + \mu_i))u_i = 0, \quad i \in M. \tag{5.12}$$

We first construct an initial solution (u,v,x,λ,μ) which satisfies the conditions (5.1)-(5.12) except (5.9) and modify the solution until the solution satisfies all the conditions (5.1)-(5.12).

In the rest of this section we will discuss a primal-dual algorithm to solve ($\bar{D}$). Let us define the notation k-max which is used in the algorithm. Given the constants $\alpha_1,\alpha_2,\cdots,\alpha_p$, we will denote by $k\text{-max}\{\alpha_i \mid i = 1,2,\cdots,p\}$ the k-th largest number in the set $\{\alpha_1, \alpha_2,\cdots, \alpha_p\}$. If $\alpha_1 \ge \alpha_2 \ge \cdots \ge \alpha_p$, then $k\text{-max}\{\alpha_i \mid i = 1,2,\cdots,p\} = \alpha_k$. Before describing the algorithm completely, we will explain the outline briefly. In the step 2 of algorithm D, using the present flow (x_{ij}), the dual variables λ, μ are determined naturally by Theorem 2. u_i $(i \in I_0 \cup I_1)$

is determined by (5.11) and (5.12), while the remaining u_i ($i \in I_F$) is averaged so that it may satisfy (5.2) and (5.10). Thus it can be seen that except (5.9) all other equations (5.1)-(5.12) hold. On the other hand, by Corollary 1, $\sum_{j \in J(i)} x_{ij}$ will be optimal if it is equal to $\bar{x}$. Hence, in the step 3 - step 8, we will try to make $\sum_{j \in J(i)} x_{ij}$ to be $\bar{x}$. Here, σ_i and τ_j are labels for $i \in M$ and $j \in N$ respectively. $\sigma_i = -1$ ($\tau_j = -1$) means that $i \in M$ ($j \in N$) is unlabeled, while $\sigma_i \geq 0$ ($\tau_j \geq 0$) means that $i \in M$ ($j \in N$) is the labeled state. In the step 10, we examine each node $i \in I$ whether it is necessary or not to change $\sum_{j \in J(i)} x_{ij}$ by using the present values of labels σ_i ($i \in I$).

Algorithm D.

(step 1) For each $j \in N$, select $i_0 \in I(j)$ and let $x_{i_0 j} = \pi_j$ and $x_{ij} = 0$ for all other x_{ij}, $(i,j) \in E$. Let $I = M$.

(step 2) Find

$$\lambda = k\text{-max}\{\sum_{j \in J(i)} x_{ij} \mid i \in M\},$$

$$I_0 = \{i \in I \mid \sum_{j \in J(i)} x_{ij} < \lambda\},$$

$$I_1 = \{i \in I \mid \sum_{j \in J(i)} x_{ij} > \lambda\},$$

$$I_F = \{i \in I \mid \sum_{j \in J(i)} x_{ij} = \lambda\},$$

$$\mu_i = \max\left(0, \sum_{j \in J(i)} x_{ij} - \lambda\right), \qquad i \in M,$$

$$u_i = \begin{cases} 1 & \text{if } i \in I_1 \\ (k - |I_1|)/|I_F| & \text{if } i \in I_F \\ 0 & \text{if } i \in I_0 \end{cases} \qquad \text{for every } i \in I,$$

$$v_j = \min\{u_i \mid i \in I(j)\}, \qquad j \in N.$$

If $(v_j - u_i)x_{ij} = 0$ are satisfied for all $(i,j) \in E$, then go to step 11.

(step 3) Let $\bar{x} = \sum_{i \in I} \sum_{j \in J(i)} x_{ij} \,/\, |I|$

and $$S_i = \sum_{j \in J(i)} x_{ij} - \bar{x}, \qquad i \in I.$$

(step 4) Let

$$\sigma_i = \begin{cases} 0 & \text{if } S_i > 0 \\ -1 & \text{otherwise} \end{cases} \qquad \text{for each } i \in I.$$

If $\sigma_i = -1$ for all $i \in I$ then go to step 9. Otherwise, let

$$\tau_j = -1 \qquad j \in N,$$

$$s_i = S_i \qquad \text{for } i \in I \text{ and } S_i > 0.$$

(step 5) For every $i \in I$ which has become $\sigma_i \geq 0$ in the previous step, if there exists $j \in J(i)$ such that $\tau_j = -1$ and $x_{ij} > 0$ then for all $j \in J(i)$ satisfying the condition, let $t_j = \min(x_{ij}, s_i)$ and $\tau_j = i$ else go to step 9.

(step 6) For every $j \in N$ which has become $\tau_j \geq 0$ in the previous step, if there exists $i \in I(j) \cap I$ such that $\sigma_i = -1$ then for all $i \in I(j) \cap I$ satisfying the condition, let $S_i = t_j$ and $\sigma_i = j$ else go to step 9.

If a node $i \in I$ which satisfies $S_i > 0$ is found above then let

$$i_0 = i,$$

$$\bar{s} = \min(s_{j_0}, -S_{j_0}),$$

$$S_{j_0} = S_{j_0} + \bar{s}$$

else go to step 5.

(step 7) Let $j_0 = \sigma_{j_0}$,

$$x_{i_0 j_0} = x_{i_0 j_0} + \bar{s},$$

$$i_0 = \tau_{j_0},$$

$$x_{i_0 j_0} = x_{i_0 j_0} - \bar{s}.$$

If $\sigma_{i_0} > 0$ then repeat step 7 else go to step 8.

(step 8) Let $S_{i_0} = S_{i_0} - \bar{s}$ and go to step 4.

(step 9) Find $\lambda = k\text{-max}\{\sum_{j \in J(i)} x_{ij} \mid i \in M\}$,

$$I_0 = \{i \in I \mid \sum_{j \in J(i)} x_{ij} < \lambda\},$$

$$I_1 = \{i \in I \mid \sum_{j \in J(i)} x_{ij} > \lambda\},$$

$$I_F = \{i \in I \mid \sum_{j \in J(i)} x_{ij} = \lambda\},$$

$$\mu_i = \max(0, \sum_{j \in J(i)} x_{ij} - \lambda), \qquad i \in M,$$

$$u_i = \begin{cases} 1 & \text{if } i \in I_1 \\ (k - |I_1 \cup \{i \in M \mid i \in I \text{ and } u_i = 1\}|) / |I_F| & \text{if } i \in I_F \\ 0 & \text{if } i \in I_0, \end{cases}$$

$$v_j = \min \{u_i \mid i \in I(j)\}, \qquad j \in N.$$

For every $(i,j) \in E$, if $(v_j - u_i)x_{ij} = 0$ is satisfied, then go to step 11.

(step 10)

1) In the case of $\lambda > \bar{x}$, let $I = \{i \in I \mid \sigma_i \geq 0\}$.

2) In the case of $\lambda < \bar{x}$, let $I = \{i \in I \mid \sigma_i = -1\}$.

3) In the case where $\lambda = \bar{x}$ and $|\{i \in M \backslash I \mid u_i = 1\}| + |\{i \in I \mid \sigma_i \geq 0\}| \geq k$, let $u_i = 0$ for every $i \in \{i \in I \mid \sigma_i = -1\}$ and $I = \{i \in I \mid \sigma_i \geq 0\}$.

4) In the case where $\lambda = \bar{x}$ and $|\{i \in M \backslash I \mid u_i = 1\}| + |\{i \in I \mid \sigma_i \geq 0\}| < k$, let $u_i = 1$ for every $i \in \{i \in I \mid \sigma_i \geq 0\}$ and $I = \{i \in I \mid \sigma_i = -1\}$.

Go to step 3.

(step 11) The present solution (u,v,x,λ,μ) is an optimal solution for $(\bar{P})$ and $(\bar{D})$.

Consider the OTMP given in Figure 4. We will solve its relaxation problem by the algorithm D. The initial values of x_{ij} $((i,j) \in E)$, λ and other variables are decided by step 1 - 4 as shown in Figure 5. After several steps, the algorithm moves to step 10 and we show the values of variables and labells at that step in Figure 6. Since the case 3 happens at the step 10, the set I becomes $\{1,2,5,6\}$. Next, the algorithm moves to the step 3 and $\bar{x}$ becomes $\sum_{i \in I} \sum_{j \in J(i)} x_{ij} / |I| = (3+2+2+2)/4 = 9/4$. After several steps, the algorithm moves to the step 4 and the case that $\sigma_i = -1$ for all $i \in I$ happens. Hence, we get an optimal solution shown in Figure 7 and the algorithm D terminates.

In the remaining of this section it will be proved that the algorithm D is finite and the obtained solution (u,v,x,λ,μ) is an optimal solution of $(\bar{P})$ and $(\bar{D})$.

Lemma 1. In the case of 1) or 3) at the step 10 in the algorithm D, the $\ell \in M$ which is in the old I, but not in the new I satisfies (5.1) - (5.12) for every $(\ell,j) \in E$ in the following steps.

Proof: Proof will be given for case 1). The case 3) can be proved similarly. By the labelling rules at step 4,5 and the changing rules of x_{ij} at step 6,7, every $x_{\ell j}$ $(j \in J(\ell))$ remains constant in the subsequent steps. Moreover, considering the step 9, u_ℓ is always 0 in the subsequent steps. Hence, $v_j = \min_{i \in I(j)} u_i = u_\ell = 0$ for any $j \in J(\ell)$. There-

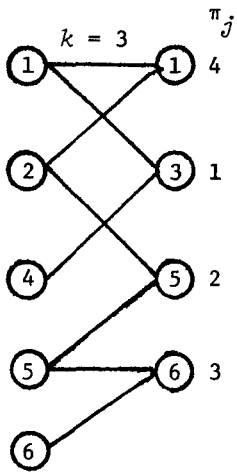
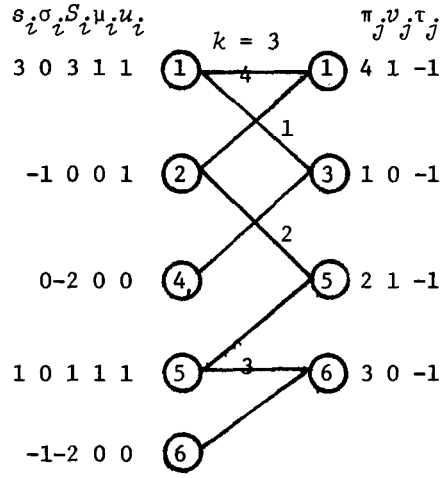
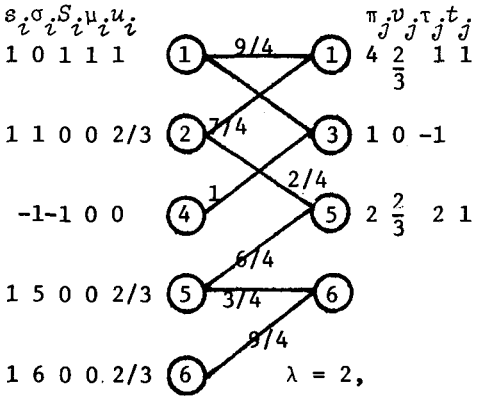


Figure 4. Reduced subproblem SP(1) for the Problem in Fig. 2



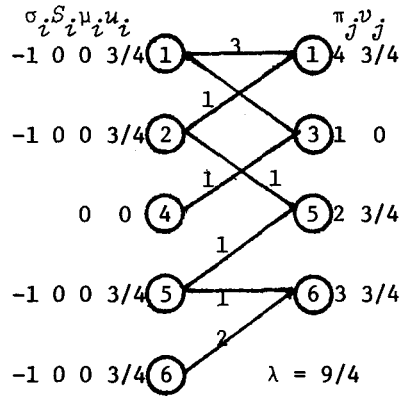
$\lambda = 2$,
$I_0 = \{4,6\}$,
$I_1 = \{1,5\}$,
$I_F = \{2\}$,
$I = \{1,2,4,5,6\}$,
$\bar{x} = 10/5 = 2$

Figure 5. Initial values of variables for the problem in Fig. 4



$\lambda = 2$,
$I_0 = \{4\}$,
$I_1 = \{1\}$,
$I_F = \{2,5,6\}$,
old $I = \{1,2,4,5,6\}$,
new $I = \{1,2,5,6\}$,
$\bar{x} = 10/5 = 2$

Figure 6. Values of variables at the step 10 for the problem in Fig. 4



$\lambda = 9/4$
$I_0 = \emptyset$,
$I_1 = \emptyset$,
$I_F = \{1,2,5,6\}$,
$I = \{1,2,5,6\}$,
$\bar{x} = 9/4$

Figure 7. Values of an optimal solution for the problem in Fig. 4

fore the complementarity (5.9) holds for every $(\ell,j) \in E$. Also, it could be easily verified that the other conditions (5.1) - (5.8) and (5.10) - (5.12) except (5.11) hold for $\ell \in M$ and $j \in J(\ell)$ in the subsequent steps by taking account of the step 9.

Now, we show that the condition (5.11) holds for $\ell \in M$ in the subsequent steps. In the step 10,

$$\sum_{j \in J(\ell)} x_{\ell j} \leq \bar{x},$$

and (5.13) $$\sum_{j \in J(i)} x_{ij} \geq \bar{x}, \qquad i \in I \ (I \text{ is the new } I),$$

holds. Moreover, the value of $\sum_{j \in J(\ell)} x_{\ell j}$ is fixed in the subsequent steps. Since we assumed the case 1) and (5.13), λ in the subsequent steps satisfies $\lambda \geq \bar{x}$ for the present value of $\bar{x}$. Hence, in the any subsequent step, $\lambda > \sum_{j \in J(\ell)} x_{\ell j}$. Since $\mu_\ell = \max (0, \sum_{j \in J(\ell)} x_{\ell j} - \lambda) = 0$, condition (5.11) holds. □

Lemma 2. In the case of 2) or 4) on the step 10 in Algorithm D, the suffix $\ell \in M$ which is in the old I, but not in the new I satisfies (5.1) - (5.12) for all $(\ell,j) \in E$ in the following step.

Proof: We prove it for case 2). The case 4) can be proved similarly. By the same reason mentioned in the proof of Lemma 1, every $x_{\ell j}$ ($j \in J(\ell)$) and $u_\ell = 1$ remain constant in the subsequent steps. Hence the conditions (5.1) - (5.12) except (5.9) always hold by considering the step 9.

Now, we show that condition (5.9) holds in the subsequent steps for every $(\ell,j) \in E$. Since $u_\ell = 1$, it suffices to show that for every $j \in J(\ell)$ such that $v_j < 1$, $x_{\ell j} = 0$ holds. Suppose there exists $j_0 \in J(\ell)$ such that $v_{j_0} < 1$ and $x_{\ell j_0} > 0$. Then, there exists a node $i_0 \in M$ which satisfies $u_{i_0} = v_{j_0} < 1$.

(1) Suppose $i_0 \in I$ holds for the old I. Since the case 2) is assumed, the node ℓ is labelled (i.e. $\sigma_\ell \geq 0$). Hence, by taking acount of $x_{\ell j_0} > 0$, the node j_0 is labelled (i.e. $\tau_{j_0} \geq 0$) by the labelling rule (step 5) and the node i_0 is also labelled by the labelling rule (step 6). Hence, by assumeng the case 2), $\sum_{j \in J(i_0)} x_{i_0 j} \geq \bar{x} > \lambda$ holds and $u_{i_0} = 1$ by considering the step 9. This is a contradiction.

(2) Suppose that $i_0 \notin I$ for the old I. Let us consider the preceding step 10 at which i_0 is in the old I, but not in the new I at the preceding step. Since $\sum_{j \in J(\ell)} x_{\ell j} \geq \sum_{j \in J(i_0)} x_{i_0 j}$ holds, by taking acount of

the reducing rule of the set I (step 10), it is easily verified that the case 1) or 3) happens at the preceding step. Then we can easily show by the similar way to (1) that for every $i \in I(j_0) \cap I$ (I is the new I at the preceding step 10), $x_{ij_0} = 0$ holds. Hence, by the labelling rule (step 4 and 5), $x_{ij_0} = 0$ ($i \in I(j_0) \cap I$) remains constant in the subsequent steps. Therefore, $x_{ij} = 0$ holds and this is a contradiction. □

Lemma 3. When $\sigma_i = -1$ holds for any $i \in I$ at step 4, (5.1) - (5.12) hold for every $(i,j) \in E$ at step 9 and Algorithm D stops.

Proof: By Lemma 1 and 2, it is sufficient to show that the complementarity (5.9) holds for any $(i,j) \in E$ where $i \in I$. If $S_i > 0$ for some $i \in I$ the $\sigma_i = 0$. Then, by the assumption of Lemma 3, $S_i = 0$ for every $i \in I$. Hence, $\sum_{j \in J(i)} x_{ij} - \bar{x} = S_i = 0$ and $u_i = (k - |I_1 \cup \{i \in M \mid i \notin I$ and $u_i = 1\}|) / |I_F|$ for every $i \in I$. Suppose there exists an $i_0 \in I$ and a $j_0 \in J(i_0)$ such that the complementarity (5.9) does not hold. Since $v_{j_0} = \min_{i \in I(j_0)} u_i < u_{i_0}$, $x_{i_0 j_0} > 0$ and $v_{j_0} = 0$ hold, then we can derive a contradiction in the same way to prove the Lemma 2. □

Theorem 3. Algorithm D stops in the finite steps and gives optimal solutions for $(\bar{P})$ and $(\bar{D})$.

Proof: At first, we will prove that the node set I will decrease monotonously at the step 10. If $\sum_{j \in J(i)} x_{ij} = \bar{x}$ holds for every $i \in I$, then the case that $\sigma_i = -1$ for all $i \in I$ happens at the step 4 and Algorithm D does not move to the step 10. Hence, there exists $i_0, i_1 \in I$ such that $\sum_{j \in J(i_0)} x_{i_0 j} > \bar{x} > \sum_{j \in J(i_1)} x_{i_1 j}$. Then, by step 3 - step 8, $\sigma_{i_0} = 0$ and $\sigma_{i_1} = -1$ hold. Suppose the case 1) or 3) occurs, $i_1 \notin I$ holds for the new I. Suppose the case 2) or 4) occurs, $i_0 \notin I$ for the new I. In any case, the set I is monotone decreasing.

Furthermore, when I decreases monotonously, for all elements $i \in M$ leaving I, conditions (5.1) - (5.12) hold in the subsequent steps by Lemma 1 and 2. Suppose $|I| = 1$ then assumption of Lemma 3 holds by considering the step 3 and 4. Hence, in the algorithm D, the assumption of Lemma 3 holds at some step and conditions (5.1) - (5.12) hold for every $(i,j) \in E$ by Lemma 3. Therefore, the algorithm stops in the finite steps and the solution (u,v,x,λ,μ) obtained gives optimal solutions for $(\bar{P})$ and $(\bar{D})$. □

6. Numerical Examples and Computational Experiences

Consider the OTMP given by Table 1 where the maximum size of tool magazine is $k = 8$. We will solve it and get an optimal solution as shown in the following. $u = (1,0,1,1,1,0,1,0,0,1,1,0,0,1,0)$, $v = (1,0,1,0,0,1,0,0,0,0,1,0,0,0,0,0,0,0,1,1,0,1,0,0,0)$ or the tool module $U = \{1,3,4,5,7,10,11,14\}$ and the parts family $V = \{j \in N \mid I(j) \subset U\} = \{1,3,6,11,19,20,22\}$ is an optimal solution for the problem. This solution is obtained by executing the branch and bound algorithm for 49 nodes and the CPU time is 0.88 seconds by Hitac M-180.

The other computational examples are shown in Tables 2 - 7. There, for the same size of m and n, the structures of problems (i.e. the set E and π) are the same and only k varies. The structures of the problems are chosen from the real OTMP's available. Some real OTMP's are shown in Table 7. By these examples, we verified that the algorithm we proposed is quite efficient and can solve practical problems.

7. Concluding Remarks

Among the set-up operations hindering the improvement of productivity of NC machines and their unmanned operations, the tool changing operation is essential and unavoidable. To reduce the frequency of set-up operations for the tool changing operation, *tool module* method has been proposed. However, the problem of designing an optimal tool module is unsolved. In this paper, we have pointed out that for this *tool module* method, there are two mathematical programming problems, i.e.,

(1) Optimal Parts Grouping Problem,

(2) Optimal Tool Module Design Problem.

Here, we have proposed for the latter problem a method which is a combination of a branch and bound method and a network flow method. Moreover, we have verified through numerical examples that the proposed method is quite efficient.

Also, we have pointed out that the latter problem includes some practical problems such as a problem to find the tool module maximizing the number of processing parts or a problem to find a tool module making possible the longest time unmanned operation of a factory with no tool change. We can further point out that the latter problem is regarded as a subproblem for the former one. By the column generation approach, the OPGP can be formulated as a set covering problem described as follows.

Table 1. Matrix for an "Optimal Tool Module Design Problem"

(k=8)

M \ N	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
1	1		1			1			1		1		1												
2		1		1			1			1		1			1										
3			1					1			1	1													
4	1			1	1			1				1				1			1						
5						1			1				1		1			1		1	1				
6				1				1								1	1								
7	1	1	1								1		1					1		1	1	1			
8							1	1							1		1							1	
9					1							1			1			1			1				1
10			1			1			1					1		1	1					1		1	
11	1	1												1			1		1			1		1	
12				1						1													1		1
13							1		1				1			1					1				1
14		1			1						1			1				1		1		1			
15					1		1		1					1							1		1	1	
π_j	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 2. CPU time for k = 10

(sec)

m \ n	20	30	40	50	60
20	0.28	0.49	0.76	1.07	1.31
30	0.56	0.89	1.68	1.40	1.58
40	0.52	1.31	1.57	2.35	2.33
50	0.65	1.50	2.83	4.62	2.77
60	0.65	1.50	2.83	4.62	2.77

Table 3. CPU time for k = 12

(sec)

m \ n	20	30	40	50	60
20	0.19	0.08	0.95	1.35	1.86
30	0.94	1.64	1.73	1.74	4.15
40	1.21	1.73	3.51	4.58	1.02
50	1.27	1.74	2.92	7.11	7.21
60	1.29	4.15	6.35	7.61	6.61

Table 4. CPU time for k = 14

(sec)

m \ n	20	30	40	50	60
20	0.27	0.08	0.10	0.67	1.10
30	1.56	1.65	2.70	3.90	0.30
40	1.41	2.28	4.89	6.81	2.02
50	3.18	2.24	4.46	8.32	5.80
60	3.15	2.97	9.47	9.96	9.23

Table 5. CPU time for k = 16

(sec)

m \ n	20	30	40	50	60
20	0.04	0.22	0.10	0.16	0.50
30	1.45	1.70	0.25	3.73	5.20
40	1.58	2.76	1.46	6.75	6.52
50	3.07	2.98	3.74	7.50	10.37
60	3.08	3.84	11.49	9.09	11.02

Table 6. CPU time for k = 18

(sec)

m \ n	20	30	40	50	60
20	0.11	0.08	0.10	0.14	0.16
30	0.26	2.26	0.70	4.17	4.25
40	1.71	2.93	0.26	4.85	3.62
50	2.53	1.06	10.39	8.63	8.96
60	2.54	0.74	6.14	9.91	8.84

Table 7. CPU time for real problems in hand

No.	m	n	k	CPU(sec)
1	37	18	16	1.25
2	78	17	16	3.29
3	90	20	16	5.97
4	110	34	16	7.06
5	151	76	16	76.53

Enumerate all tool combinations for a given set of tools $i = 1,2,\cdots,m$. Since tools in a module must be within the maximum size k of tool magazine, the total number of modules is $T = \sum_{\ell=1}^{k} {}_m C_\ell$ and we denote each tool module by $t = 1,2,\cdots,T$. Denote by the m-dimensional 0-1 vector $u^{(t)} = (u_1^{(t)}, u_2^{(t)},\cdots,u_m^{(t)})^\top$ the t-th tool module and by n dimensional 0-1 vector $v^{(t)} = (v_1^{(t)},v_2^{(t)},\cdots,v_n^{(t)})^\top$ the set of parts that can be processed by $u^{(t)}$, where $u_i^{(t)} = 1$ (= 0) means the tool module t includes (does not include) the tool i and $v_j^{(t)} = 1$ (= 0) means that it is (not) possible to process part j by the tool module t. Then, the OPGP is formulated as the well-known set covering problem:

$$
\begin{aligned}
(Q) \qquad & \text{minimize} \quad \sum_{t=1}^{T} y_t \\
& \text{subject to} \quad \sum_{t=1}^{T} v^{(t)} y_t \geq e, \\
& \qquad\qquad y_t \in \{0,1\}, \qquad t = 1,2,\cdots,T,
\end{aligned}
$$

where e is an m-dimensional vector $(1,1,\cdots,1)^\top$, and $y_t = 1$ (= 0) means that the optimal set of tool modules contains (does not contain) the t-th tool module $u^{(t)}$. One of the difficulties of this approach is that it takes a lot of time to generate all tool modules $u^{(t)}$ and the corresponding part sets $v^{(t)}$ respectively.

Now, suppose some of coefficient column vectors (i.e., parts families) are known for (Q) such that every part belongs to at least one of these known parts-families, and we call this problem by (R). Denote by $(\bar{R})$ the LP relaxation problem of (R). By solving $(\bar{R})$, the dual optimal solutions π_j $(j = 1,2,\cdots,n)$ is obtained corresponding to each constraint. Since π_j is a shadow price for the constraint j (i.e., the part j), it can be considered that π_j shows a value to process the part j. In this sense, the problem to find a most valuable parts family $v^{(t)}$ and the corresponding tool module $u^{(t)}$ is:

$$
\begin{aligned}
(P) \qquad & \text{maximize} \quad \sum_{j \in N} \pi_j v_j \\
& \text{subject to} \\
& \qquad v_j \leq u_i, \qquad (i,j) \in E, \\
& \qquad \sum_{i \in M} u_i \leq k,
\end{aligned}
$$

$$u_i \in \{0,1\}, \quad i \in M,$$
$$v_j \in \{0,1\}, \quad j \in N.$$

This is evidently the OTMP disscussed in this paper.

Now, it is possible to develop an effective method for solving (Q). By solving (P), we obtain its optimal solution u^*, v^* where v^* represents the parts that can be processed by u^*. After that, we add v^* to (R) as a new column vector $v^{(t)}$. Thus, we can generate only meaningful columns of (Q), which are possibly needed to construct an optimal solution for (Q), and it will be enough to solve the set covering problem with smaller size instead of (Q).

Acknowledgment

The authors wish to thank the editor and reviewers for their helpful comments in revising the paper.

References

[1] Geoffrion, A. M.: Lagrangean Relaxation for Integer Programming, *Mathematical Prgramming Study*, Vol.2 (1974), 82-114.

[2] Kobayashi, T.: The Lexico-Shortest Route Algorithm for Solving the Minimum Cost Flow Problem with an Additional Linear Constraint. *Journal of the Operations Research Society of Japan*, Vol.26, No.3 (1983), 167-185.

[3] Kobayashi, T.: The Lexico-Bounded Flow Algorithm for Solving the Minimum Cost Project Scheduling Problem with an Additional Linear Constraint. *Research Report* B-82-4, Saitama University (August, 1982).

[4] Lawler, E.L.: *Combinatorial Optimization: Networks and Matroids*. Halt, Rinehart and Winston, New York, 1976.

[5] Tanaka, N., Tsuchiya, N. and et. al.: A Study in Parts Classification by Used Tools (in Japanese). *Proceeding of Spring Conference of Japan Industrial Management Association*, 1982, 129-130.

Ryuichi HIRABAYASHI: Department of Industrial Engineering, Tokyo Science University, Kagurazaka 1-3, Shinjuku-ku, Tokyo, 162, Japan.