J. Operations Research Soc. of Japan
Vol. 19, No. 1, March, 1976

TWO-MACHINE SCHEDULING UNDER REQUIRED PRECEDENCE AMONG JOBS

TADASHI KURISU
Osaka University

(Received July 20; Revised October 23, 1974)

Abstract

In a general two-machine n-job scheduling problem, it is assumed that every possible sequence of jobs can be executed, so that whichever best served a given measure can be selected. This paper considers two more restricted cases in which certain orderings are prohibited, either by technological constraints or by externally imposed policy. In the first case, some of the decisions of a schedule have already been made and the schedule must be completed without altering what has been decided. In the second case, jobs are grouped into disjoint subsets within which a job order is specified, but which may be preempted between jobs. For each of these two cases, a rule is given for determining the sequence in which jobs are to be processed on the machines in order to minimize the total elapsed time.

1. Introduction

Bellman [1] and Johnson [5] considered a problem involving the scheduling of n jobs on two machines. In their formulation, we are given two machines, I and II, and a set of n jobs. Also given are the processing times (including any set-up or tear-down times), A_i and B_i, for each job i on machines I and II respectively. Each machine can handle only one job at a time and each job must be processed through machine I and then machine II. Johnson gave a simple decision rule for the optimal scheduling so that the total elapsed time is minimum.

Mitten [5, 6] treated a scheduling problem which is similar to the Bellman-Johnson problem. In his model, associated with each job i is processing times, A_i and B_i on machines I and II respectively, and a start-lag

D_i (≥ 0) and a stop-lag E_i (≥ 0). The start-lag is defined to be the minimum time which must elapse between starting job i on machine I and starting it on machine II, while the stop-lag is defined to be the minimum time which must elapse between completing job i on machine I and completing it on machine II. He gave a decision rule to obtain the sequence in which the jobs are to be processed on the machines, using the same sequence for both machines, in order to minimize the total elapsed time.

Johnson [4] considered a more difficult general case, where different job sequences are allowed for the two machines. He gave a necessary condition for a reversal of order of consecutive jobs i, j on machine I to j, i on machine II in a pair of mutually optimal sequences (S_I, S_{II}), where S_I and S_{II} are the optimal sequences on machines I and II respectively. He also gave a sufficient condition under certain restrictions.

We remark that the Mitten-Johnson problem with $D_i = A_i$ and $E_i = B_i$ reduces to the Bellman-Johnson problem.

In all the preceding papers, it is assumed that every possible sequence of the jobs can be executed, so that whichever best served a given measure can be selected. In this paper, we consider situations in which certain orderings are prohibited, either by technological constraints or by externally imposed policy.

2. String Problems

In the Bellman-Johnson two-machine scheduling problem, we consider a situation, in which some of the decisions of a schedule have already been made and in which we have the task of completing the schedule without altering what has been decided. In general, suppose that the original n jobs have been grouped into w disjoint subsets of jobs called strings. Assume that the membership of each string is fixed, that the order of jobs within each string is fixed, and that once started an entire string must be processed to be completed. We denote a string by $I_i = (\alpha_1, \alpha_2, \ldots, \alpha_{n_i})$ which indicates that jobs $\alpha_1, \alpha_2, \ldots, \alpha_{n_i}$ must be processed without interruption according to this order. In this section, we give a method to obtain a string sequence which minimizes the total elapsed time.

Let A_i^* and B_i^* denote the total processing times of the string I_i on machines I and II respectively. Assuming that only the string I_i is processed,

let c_i be the total elapsed time of the string. Under the same assumption, let a_i and b_i be the total idle times of the string on machines II and I respectively. If $A(i, j)$ is the processing time of the j-th job in the string I_i on machine I and $B(i, j)$ is the corresponding time on machine II, then it is obvious that

$$A_i^* = \sum_{j=1}^{n_i} A(i, j),$$

$$B_i^* = \sum_{j=1}^{n_i} B(i, j),$$

$$c_i = \max_{1 \leqq u \leqq n_i} \{ \sum_{j=1}^{u} A(i, j) + \sum_{j=u}^{n_i} B(i, j) \},$$

$$a_i = c_i - B_i^* = \max_{1 \leqq u \leqq n_i} \{ \sum_{j=1}^{u} A(i, j) - \sum_{j=1}^{u-1} B(i, j) \} > 0$$

and

$$b_i = c_i - A_i^* = \max_{1 \leqq u \leqq n_i} \{ \sum_{j=u}^{n_i} B(i, j) - \sum_{j=u+1}^{n_i} A(i, j) \} > 0.$$

Lemma 1. For a string problem, it suffices to consider the schedules in which the total idle time, on machine II, of each string is put before the start of the first job in the string.

Proof. If the total idle time, on machine II, of a string is put before the start of the first job in the string, and so, jobs in the string are processed successively on machine II, then the completion-time of the last job in the string does not increase while those of the other jobs in the string may increase. Hence, starting times of the jobs which belong to other strings do not increase. Therefore, the total elapsed time does not increase even if the total idle time, on machine II, of each string is put before the first job in the string.

If the total idle time, on machine II, of each string is put before the start of the first job in the string and the corresponding time on machine I

is put after the end of the last job in the string, then string I_i is neither started on machine II sooner than a_i time units after it was started on machine I, nor finished on machine II sooner than b_i time units after it was finished on machine I. Thus, the string problem reduces to the Mitten-Johnson problem with w jobs. In the reduced problem, associated with each job i which corresponds to string I_i in the original string problem is processing times A_i^* and B_i^*, on machines I and II respectively, and the start-lag a_i and the stop-lag b_i.

Let $M_i = \max\ (a_i - A_i^*,\ b_i - B_i^*)$ and let S_I' and S_{II}' be the sequences of strings on machines I and II respectively. The following lemma can be proved in the same manner as a theorem of Johnson [4].

Lemma 2. For a string problem, a necessary condition for a reversal of order of consecutive strings I_i, I_j on machine I to I_j, I_i on machine II in a pair of mutually optimal sequences (S_I', S_{II}') is that

$$M_i > M_j + \max\ (A_j^*,\ B_j^*).$$

This is also a sufficient condition provided string I_i is not reversed with its preceding string on machine I and string I_j is not reversed with its following string on machine II.

Theorem 1. An optimal ordering for a string problem is given by the following rule. String I_i precedes string I_j if

$$\min\ (a_i,\ b_j) < \min\ (a_j,\ b_i).$$

If there is equality, either ordering is optimal, provided it is consistent with all the definite preferences.

Proof. As noted previously, a string problem reduces to the Mitten-Johnson problem. Since

$$a_i - A_i^* = b_i - B_i^* = -\min_{1 \leq u \leq n_i} \left\{ \sum_{j=u+1}^{n_i} A(i, j) + \sum_{j=1}^{u-1} B(i, j) \right\} \leq 0,$$

we have

$$M_i = \max\ (a_i - A_i^*,\ \ b_i - B_i^*) \leqq 0$$

and

$$M_i + \max\ (A_i^*,\ \ B_i^*) = \max\ (a_i - A_i^*,\ \ b_i - B_i^*) + \max\ (A_i^*,\ \ B_i^*)$$

$$= \max\ (a_i,\ \ b_i) > 0,$$

and thus, $M_i < M_j + \max\ (A_j^*,\ \ B_j^*)$ for any i and j. Therefore, from Lemma 2, it suffices to consider the sequences in which orderings are identical for both machines. Since $A_i^* + M_i = a_i$ and $B_i^* + M_i = b_i$, the theorem is apparent by a theorem of Mitten [6]. This terminates our proof.

3. Relations between Strings

In this section, we prove relations between strings.

For strings I_i and I_j, $I_i \succ I_j$, $I_i \sim I_j$ and $I_i \gg I_j$ represent

$$\min\ (a_i,\ \ b_j) \leqq \min\ (a_j,\ \ b_i),$$

$$\min\ (a_i,\ \ b_j) = \min\ (a_j,\ \ b_i)$$

and

$$\min\ (a_i,\ \ b_j) < \min\ (a_j,\ \ b_i),$$

respectively. We remark that the relation $\gg$ is transitive.

For strings $I_i = (\alpha_1, \alpha_2, \ldots, \alpha_{n_i})$ and $I_j = (\beta_1, \beta_2, \ldots, \beta_{n_j})$, we denote a string $(\alpha_1, \alpha_2, \ldots, \alpha_{n_i}, \beta_1, \beta_2, \ldots, \beta_{n_j})$ by (I_i, I_j).

For strings I_1, I_2 and I_3, we put $I_4 = (I_1, I_2)$ and $I_5 = (I_2, I_3)$. It should be noted that

$$a_i - b_i = A_i^* - B_i^* \qquad \text{for all } i,$$

$$a_4 = \max\ (a_1,\ \ a_1 + a_2 - b_1),$$

$$a_5 = \max\ (a_2,\ \ a_2 + a_3 - b_2),$$

$$b_4 = \max\ (b_1 + b_2 - a_2,\ \ b_2),$$

$$b_5 = \max\ (b_2 + b_3 - a_3,\ \ b_3),$$

$$a_4 - b_4 = a_1 - b_1 + a_2 - b_2$$

and

$$a_5 - b_5 = a_2 - b_2 + a_3 - b_3.$$

Lemma 3. Let I_1, I_2 and I_3 be strings. If $I_1 \gg I_2 \gg I_3$, then $(I_1, I_2) \gg I_3$ and $I_1 \gg (I_2, I_3)$.

Proof. We prove $(I_1, I_2) \gg I_3$, i.e.,

(1) $$\min (a_4, b_3) < \min (a_3, b_4).$$

Since $I_1 \gg I_2 \gg I_3$, we have

$$\min (a_1, b_2) < \min (a_2, b_1)$$

and

$$\min (a_2, b_3) < \min (a_3, b_2).$$

Case 1. $b_3 < a_3, b_2$.

Then $b_3 < a_3$ and $b_3 < b_2 \leqq b_4$ so that $b_3 < \min (a_3, b_4)$.

Case 2. $a_1 < a_2, b_1$ and $a_2 < a_3, b_2$.

Then we get

$$a_4 = \max (a_1, a_1 + a_2 - b_1) \leqq \max (a_1, a_2) = a_2 < a_3$$

and $a_4 \leqq a_2 < b_2 \leqq b_4$. Thus, we obtain $a_4 < \min (a_3, b_4)$.

The proof of $I_1 \gg (I_2, I_3)$ is similar. This terminates our proof.

Lemma 4. Let I_1, I_2 and I_3 be strings. If $I_1 \prec I_2$ and $(I_1, I_2) \prec I_3$, then $I_1 \prec (I_2, I_3)$.

Proof. It suffices to show

(2) $$\min (a_5, b_1) \leqq \min (a_1, b_5).$$

Since $I_1 \prec I_2$ and $(I_1, I_2) \prec I_3$, we get

$$\min (a_2, b_1) \leqq \min (a_1, b_2)$$

and

$$\min (a_3, b_4) \leqq \min (a_4, b_3).$$

Case 1. $b_1 \leqq a_1, a_2, b_2$ and $b_4 \leqq a_3, a_4, b_3$.

Then $b_1 \leqq a_1$ and $b_1 \leqq b_2 \leqq b_4 \leqq b_3 \leqq b_5$ so that $b_1 \leqq \min (a_1, b_5)$.

Case 2. $a_2 \leqq a_1, b_1, b_2$ and $b_4 \leqq a_3, a_4, b_3$.

Then we get

$$b_5 \geqq b_3 \geqq b_4 = \max (b_1 + b_2 - a_2, b_2) \geqq \max (b_1, b_2) \geqq b_1$$

and

$$a_1 - b_1 \geqq a_1 - b_1 + a_2 - b_2 = a_4 - b_4 \geqq 0.$$

Hence, (2) is proved.

Case 3. $b_1 \leqq a_1, a_2, b_2$ and $a_3 \leqq a_4, b_3, b_4$.

Then $b_1 \leqq a_1$ and

$$b_1 \leqq b_2 \leqq \max (b_2, b_3) \leqq \max (b_2 + b_3 - a_3, b_3) = b_5.$$

Hence, we have $b_1 \leqq \min (a_1, b_5)$.

Case 4. $a_2 \leqq a_1, b_1, b_2$ and $a_3 \leqq a_4, b_3, b_4$.

Then $b_5 - a_5 = b_2 - a_2 + b_3 - a_3 \geqq 0$. If $a_1 \geqq b_1$, then we get (2). If $a_1 < b_1$, then

$$a_5 = \max (a_2, a_2 + a_3 - b_2) \leqq \max (a_2, a_3) \leqq \max (a_1, a_4) = a_4$$

$$= \max (a_1, a_1 + a_2 - b_1) \leqq \max (a_1, a_2) = a_1.$$

Hence, (2) is proved.

This terminates our proof.

The same type of proof as above establishes

Corollary 1. Let I_1, I_2 and I_3 be strings. If $I_1 \prec (I_2, I_3)$ and $I_2 \prec I_3$, then $(I_1, I_2) \prec I_3$.

4. Chain Problems

In this section, we consider a more general situation in which directly-

precedes relationships are given between certain pairs of jobs such that a given job has at most one predecessor and at most one successor. Thus, original jobs are partitioned into disjoint subsets called chains within which a job order is specified, while interruptions between jobs are allowed. Let a chain $C_i = \{\alpha_1, \alpha_2, \cdots, \alpha_{n_i}\}$ denote the relation that jobs $\alpha_1, \alpha_2, \cdots, \alpha_{n_i}$ must be processed according to this order, while preemption between these jobs is admissible. In what follows, we give a method to obtain a job sequence which minimizes the total elapsed time under the constraint that original n jobs be partitioned into w chains, with $n_1, n_2, \cdots, n_w$ as the numbers of jobs in the corresponding chains.

If a chain $C_i = \{\alpha_1, \alpha_2, \cdots, \alpha_{n_i}\}$ is divided in such a manner that all the constraints of the order to be processed are satisfied, then the division is called a division of the chain C_i. If $\{(\alpha_1, \alpha_2, \cdots, \alpha_s), (\alpha_{s+1}, \alpha_{s+2}, \cdots, \alpha_k), \cdots, (\alpha_{l+1}, \alpha_{l+2}, \cdots, \alpha_m), \cdots, (\alpha_{u+1}, \alpha_{u+2}, \cdots, \alpha_{n_i})\}$ is a division of a chain C_i, then each $(\alpha_{l+1}, \alpha_{l+2}, \cdots, \alpha_m)$ is said to be a subchain of C_i. For simplicity, we denote a division of a chain by $\{(I_1), (I_2), \cdots, (I_j), \cdots, (I_q)\}$, where each I_j is a subchain. For subchains I_j and I_k, $I_j \succ I_k$, $I_j \sim I_k$ and $I_j \gg I_k$ denote $I_j \succ I_k$, $I_j \sim I_k$ and $I_j \gg I_k$, respectively, regarding I_j and I_k as strings.

We call a subchain $(\alpha_{l+1}, \alpha_{l+2}, \cdots, \alpha_t, \alpha_{t+1}, \cdots, \alpha_m)$ an elementary subchain if and only if $(\alpha_{l+1}, \alpha_{l+2}, \cdots, \alpha_t) \prec (\alpha_{t+1}, \alpha_{t+2}, \cdots, \alpha_m)$ for any subdivision $\{(\alpha_{l+1}, \alpha_{l+2}, \cdots, \alpha_t), (\alpha_{t+1}, \alpha_{t+2}, \cdots, \alpha_m)\}$ of the subchain. We call a division of a chain an elementary division of the chain if and only if all the subchains in the division are elementary subchains. For example, $\{(\alpha_1), (\alpha_2), \cdots, (\alpha_{n_i})\}$ is an elementary division of the chain $\{\alpha_1, \alpha_2, \cdots, \alpha_{n_i}\}$.

Theorem 2. Let $\{(I_1), (I_2), \cdots, (I_j), (I_{j+1}), \cdots, (I_q)\}$ be an elementary division of a chain. If $I_j \prec I_{j+1}$, then $\{(I_1), (I_2), \cdots, (I_{j-1}), (I_j, I_{j+1}), (I_{j+2}), \cdots, (I_q)\}$ is also an elementary division of the chain.

Proof. Putting $I_j = (\alpha_{l+1}, \alpha_{l+2}, \cdots, \alpha_k)$ and $I_{j+1} = (\alpha_{k+1}, \alpha_{k+2}, \cdots, \alpha_m)$, we consider a subdivision $\{(\alpha_{l+1}, \alpha_{l+2}, \cdots, \alpha_t), (\alpha_{t+1}, \alpha_{t+2}, \cdots, \alpha_m)\}$

of the subchain (I_j, I_{j+1}). By assumption, we have $I_j \prec I_{j+1}$. If $t < k$, then since I_j is an elementary subchain we get $(\alpha_{l+1}, \alpha_{l+2}, \cdots, \alpha_t) \prec (\alpha_{t+1}, \alpha_{t+2}, \cdots, \alpha_k)$. Thus, it follows from Lemma 4 that $(\alpha_{l+1}, \alpha_{l+2}, \cdots, \alpha_t) \prec (\alpha_{t+1}, \alpha_{t+2}, \cdots, \alpha_k, \alpha_{k+1}, \cdots, \alpha_m)$. If $t > k$, then by the similar argument, we get the same conclusion. If $t = k$, then obviously $(\alpha_{l+1}, \alpha_{l+2}, \cdots, \alpha_t) \prec (\alpha_{t+1}, \alpha_{t+2}, \cdots, \alpha_m)$. Hence, (I_j, I_{j+1}) is an elementary subchain. Since all other subchains are elementary subchains, we get the desired result.

In what follows, T(S) denotes the total elapsed time under a schedule S.

Lemma 5. If $I_j \prec I_{j+1}$, then for any strings J_1, J_2 and J_3, one of the relations:

$$T(\{(J_1), (I_j), (I_{j+1}), (J_2), (J_3)\}) \leqq T(\{(J_1), (I_j), (J_2), (I_{j+1}), (J_3)\}) \tag{3}$$

and

$$T(\{(J_1), (J_2), (I_j), (J_{j+1}), (J_3)\}) \leqq T(\{(J_1), (I_j), (J_2), (I_{j+1}), (J_3)\}) \tag{4}$$

is satisfied.

Proof. Since $I_j \prec I_{j+1}$, one of (i) $I_j \gg J_2$ and $J_2 \prec I_{j+1}$, (ii) $I_j \prec J_2$ and $J_2 \gg I_{j+1}$ and (iii) $I_j \prec J_2 \prec I_{j+1}$ is satisfied. In case (i), (3) holds. In case (ii), (4) holds. And in case (iii), (3) and (4) hold.

Theorem 3. Let I* be an elementary subchain. If I* is partitioned into $\{(L_1), (L_2), \cdots, (L_l)\}$, then for any set of $l+1$ strings $J_1, J_2, \cdots, J_{l+1}$, one of the relations:

$$T(\{(J_1), (I^*), (J_2), (J_3), \cdots, (J_{l+1})\}) \leqq T(\{(J_1), (L_1), (J_2), (L_2), \cdots, (J_l), (L_l), (J_{l+1})\})$$

and

$$T(\{(J_1), (J_2), \cdots, (J_l), (I^*), (J_{l+1})\}) \leqq T(\{(J_1), (L_1), (J_2), (L_2), \cdots, (J_l), (L_l), (J_{l+1})\})$$

is satisfied.

Proof. The proof proceeds by the induction on l. For $l = 2$, the theorem is a direct consequence of Lemma 5. Assuming that the theorem is proved for $l = m-1$, we consider the case $l = m$. If $L_1 \gg L_2 \gg L_3 \gg \cdots \gg L_{m-1} \gg L_m$, then it follows from Lemma 3 that $(L_1, L_2, \cdots, L_{m-1}) \gg L_m$. This contradicts the assumption that $I^* = (L_1, L_2, \cdots, L_{m-1}, L_m)$ is an elementary subchain. Hence, there exists a k such that $L_k \prec L_{k+1}$. For the k, we put

$$S_1 = \{(J_1), (L_1), (J_2), (L_2), \cdots, (J_k), (J_{k+1}), (L_k), (L_{k+1}), (J_{k+2}), (L_{k+2}), \cdots, (J_m), (L_m), (J_{m+1})\},$$

$$S_2 = \{(J_1), (L_1), (J_2), (L_2), \cdots, (J_k), (L_k), (L_{k+1}), (J_{k+1}), (J_{k+2}), (L_{k+2}), \cdots, (J_m), (L_m), (J_{m+1})\},$$

$$S_3 = \{(J_1), (L_1), (J_2), (L_2), \cdots, (J_k), (L_k), (J_{k+1}), (L_{k+1}), (J_{k+2}), (L_{k+2}), \cdots, (J_m), (L_m), (J_{m+1})\},$$

$$S_4 = \{(J_1), (J_2), \cdots, (J_m), (I^*), (J_{m+1})\}$$

and

$$S_5 = \{(J_1), (I^*), (J_2), (J_3), \cdots, (J_{m+1})\}.$$

By Lemma 5, one of $T(S_1) \leqq T(S_3)$ and $T(S_2) \leqq T(S_3)$ holds. By the induction hypothesis, one of $T(S_4) \leqq T(S_1)$ and $T(S_5) \leqq T(S_1)$ holds and one of $T(S_4) \leqq T(S_2)$ and $T(S_5) \leqq T(S_2)$ holds. Therefore, it follows that one of the relations $T(S_4) \leqq T(S_3)$ and $T(S_5) \leqq T(S_3)$ is satisfied. This terminates our proof.

Let $\{(I_1), (I_2), \cdots, (I_q)\}$ be an elementary division of a chain. We call the division an optimal division of the chain if and only if $I_1 \gg I_2 \gg \cdots \gg I_{q-1} \gg I_q$. An optimal division of a chain $\{\alpha_1, \alpha_2, \cdots, \alpha_{n_i}\}$ is given in the following fashion:

Step 1. Let $\{(\alpha_1), (\alpha_2), \cdots, (\alpha_{n_i})\}$ be an initial elementary division. Go to Step 2.

Step 2. For the current elementary division $\{(I_1), (I_2), \ldots, (I_j), (I_{j+1}), \ldots, (I_q)\}$, find subchains I_j and I_{j+1} which satisfy $I_j \prec I_{j+1}$. Go to Step 3. If there are no such chains, then the current elementary division is an optimal division.

Step 3. Make up an elementary division $\{(I_1), (I_2), \ldots, (I_{j-1}), (I_j, I_{j+1}), (I_{j+2}), \ldots, (I_q)\}$. Return to Step 3.

The following theorem gives a method to obtain an optimal sequence for a chain problem.

Theorem 4. If all the chains in a chain problem are divided into optimal divisions, then the optimal sequences for the string problem given by regarding all the subchains in the optimal divisions as strings are optimal sequences for the original chain problem.

Proof. Let S* be an optimal sequence for the string problem regarding all the subchains in the optimal divisions of chains $C_1, C_2, \ldots, C_w$ as strings and let S be an optimal sequence for the chain problem. By Theorem 3, there is a sequence S_1, with $T(S) \geqq T(S_1)$, which is feasible for the chain problem and in which all the subchains in the optimal division of C_1 are processed successively. Similarly, there is a feasible sequence S_2 for the chain problem in which all the subchains in the optimal divisions of C_1 and C_2 are processed successively such that $T(S_1) \geqq T(S_2)$. Thus, it follows that there is a sequence S_w, with $T(S) \geqq T(S_w)$, which is feasible for the chain problem and in which all the subchains in the optimal divisions of chains C_1, $C_2, \ldots, C_w$ are processed successively. Hence, S_w is feasible for the string problem so that $T(S) \geqq T(S_w) \geqq T(S^*)$. On the other hand, since S* is feasible for the chain problem, we get $T(S) \leqq T(S^*)$. Therefore, we obtain $T(S) = T(S^*)$, and hence, S* is optimal for the chain problem. This terminates our proof.

5. Examples

In this section, we give two breaf examples which illustrate some of

the results in the preceding sections.

Example 1. Consider three strings $I_1 = (1, 2, 3, 4, 5, 6)$, $I_2 = (7, 8, 9, 10, 11)$ and $I_3 = (12, 13, 14)$. For each job i, the processing times A_i and B_i on machines I and II are given in Table 1. The total idle times for I_1 are

Table 1. The processing times of the jobs

i	1	2	3	4	5	6	7
A_i	6	3	7	5	1	6	1
B_i	3	5	2	6	7	5	3
i	8	9	10	11	12	13	14
A_i	3	7	5	8	3	4	6
B_i	2	6	1	4	5	7	2

11 on both machines, i.e.,

$$a_1 = \max_{1 \leq u \leq 6} \{ \sum_{i=1}^{u} A_i - \sum_{i=1}^{u-1} B_i \} = 11$$

and

$$b_1 = \max_{1 \leq u \leq 6} \{ \sum_{i=u}^{6} B_i - \sum_{i=u+1}^{6} A_i \} = 11.$$

The total idle times for I_2 on machines II and I are 12 and 4, respectively, i.e.,

$$a_2 = 12 \qquad \text{and} \qquad b_2 = 4.$$

The total idle times for I_3 on machines II and I are 3 and 4, respectively, i.e.,

$$a_3 = 3 \qquad \text{and} \qquad b_3 = 4.$$

For the string problem, Theorem 1 gives an optimal sequence $\{(I_3), (I_1), (I_2)\}$. The total elapsed time for the sequence is 69 time units.

Example 2. In Example 1, let us suppose that I_1, I_2 and I_3 are chains. It is easily seen that $\{(1, 2, 3, 4, 5), (6)\}$, $\{(7), (8, 9), (10, 11)\}$ and $\{(12), (13), (14)\}$ are optimal divisions of the chains I_1, I_2 and I_3, respec-

tively. For each of these subchains, Table 2 gives the total idle times a and b, on machines II and I, respectively. We read, from Table 2, that {(7), (12), (13), (1, 2, 3, 4, 5), (8, 9), (6), (10, 11), (14)} is an optimal sequence for the chain problem. The total elapsed time for the sequence is 67 time units.

Table 2. The total idle times of the subchains

subchain	(1,2,3,4,5)	(6)	(7)	(8,9)	(10,11)	(12)	(13)	(14)
a	11	6	1	8	12	3	4	6
b	12	5	3	6	4	5	7	2

Acknowledgement

The author would like to express his sincere thanks to Prof. T. Nishida and Prof. M. Sakaguchi for their continuing guidances and encouragement. He wish also to thank Dr. K. Ohno for his helpful comments and suggestions.

References

[1] Bellman, R., "Mathematical Aspects of Scheduling Theory," *J. Soc. Ind. and Appl. Math.*, Vol. 4 (1956), pp. 168-205.

[2] Conway, R., W. Maxwell and L. Miller, *Theory of Scheduling*, Addison-Wesley, 1967.

[3] Johnson, S. M., "Optimal Two-and Three-Stage Production Schedules with Setup Times Included," *Nav. Res. Log. Quart.*, Vol. 1 (1954), pp. 61-68.

[4] ——— , "Discussion: Sequencing n jobs on Two Machines with Arbitrary Time Lags," *Management Science*, Vol. 5 (1959), pp. 299-302.

[5] Mitten, L. G., "Sequencing n Jobs on Two Machines with Arbitrary Time Lags," *Management Science*, Vol. 5 (1959), pp. 293-298.

[6] ——— , "A Scheduling Problem," *J. Ind. Eng.*, Vol. 10 (1959), pp. 131-135.

Tadashi Kurisu
Department of Applied Physics
Facluty of Engineering
Osaka University
Yamada-Kami, Suita
Osaka, 565, Japan