Journal of the Operations Research
Society of Japan
Vol. 20, No. 2, June, 1977

© 1977 The Operations Research Society of Japan

TWO-MACHINE SCHEDULING UNDER ARBITRARY PRECEDENCE CONSTRAINTS

TADASHI KURISU
Osaka University

(Received July 12, 1976; Final March 10, 1977)

Abstract Suppose jobs, in the Johnson's two-machine n-job flow-shop scheduling problem, are grouped into disjoint subsets within which a job order that may not be preempted is specified. Furthermore, suppose that a precedence relation between these subsets is given such that the processing of a subset must be completed, on each machine, before the processing of another subset begins on the machine. This paper considers a problem to find a sequence in which jobs are to be processed on the machines in order to minimize the total elapsed time, under such general precedence constraints, from the start of the first job on machine I until the end of the last job on machine II. An efficient algorithm to obtain an optimal sequence is given and a simple example is shown.

1. Introduction

A meaningful study of a common production system necessitates intimate knowledge of each component. The scheduling is one such component, and its detailed study is therefore an essential element of the study of the complete system. Furthermore, the investigation of the scheduling model which is rather simple than the practical situation may shed light on a more complicated case. Two-machine flow-shop problem considered by Johnson [2] is one of the most important models in the scheduling theory. In Johnson's formulation, we are given two machines I and II and a set J = {1, 2, ... , n} of n jobs. Also given are the processing times A_i and B_i for each job i on machines I and II, respectively. Each job must be completed on machine I before it can be put on machine II and only one job can be processed at one time on a machine. Johnson gave a simple decision rule for obtaining a schedule so as to minimize the total elapsed time. Johnson's paper is important, not only for its own content, but also for the influence it has had on subsequent work.

In the Johnson model and many other scheduling models (see, for example, Conway et al. [1]), it is assumed that every permutation of n jobs is feasible. In most of the practical situations, however, certain orderings are prohibited either by technological constraints or by externally imposed policy. In various forms and contexts, such a case is encountered many thousands of times daily in enterprises throughout the world:

(*a*) If setup times are highly dependent on sequence, then one would group jobs with similar setups, sequence within these groups for minimum change-over time, and arrange the groups to optimize some measure.

(*b*) If due-date is associated with each job, then the jobs with earlier due-date should be processed before the jobs with later due-date.

(*c*) If there are jobs which should be re-processed after once they have been processed, then the first processing must be completed before the second one starts. (in scheduling computer jobs, for example, a file generated as output from a job might be required as input for another job.)

The existence of such restrictions reduces the number of feasible schedules, but this does not mean than an optimal schedule can be found more readily.

Recently, one-machine scheduling problems with precedence constraints have been considered by some researchers. For example, Lawler [4] developed an algorithm to obtain a sequence which minimizes the maximum cost subject to precedence constraints. Furthermore, Sidney [5] gave an algorithm to produce a sequence minimizing the mean completion time subject to precedence constraints. While, no previous theoretical results bearing directly on the two-machine problem with precedence constraints have been obtained. However, a procedure for the special case was developed by the author [3], in which the original jobs have been decomposed into disjoint parallel chains.

The object of this paper is to develop an algorithm producing an optimal schedule, for the two-machine n-job flow-shop problem, which minimizes the total elapsed time subject to more general precedence constraints.

2. Description of Problem

Consider a job-shop consisting of two machines I and II, and a set $J = \{1, 2, \ldots, n\}$ of n jobs to be processed on these machines. We are given processing times A_i and B_i, for each job i, on machines I and II, respectively. Each machine can handle at most one job at a time and the processing of each job must be finished on machine I before it can be processed on machine II.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

Let us now introduce the concept of strings. A string $(r, s, \ldots, t)$ is an ordered set of jobs which must be processed in a fixed job order $r, s, \ldots, t$ without preemption on both machines. Of course, there may be idle times, on machine II, between jobs in a string. However, once the first job in a string has started on a machine, then all the jobs in the string must be processed to be completed on the machine without starting jobs which do not belong to the string. We assume that the original n jobs have been grouped into m disjoint strings $I_1, I_2, \ldots, I_m$ and we set $X = \{I_1, I_2, \ldots I_m\}$. For a string $I_i = (r, r+1, \ldots, t)$, we set

$$a(I_i) = \max_{r \leq k \leq t} \{ \sum_{j=r}^{k} A_j - \sum_{j=r}^{k-1} B_j \}$$

and

$$b(I_i) = \max_{r \leq k \leq t} \{ \sum_{j=k}^{t} B_j - \sum_{j=k+1}^{t} A_j \}.$$

If only the jobs in I_i is processed, then the total elapsed time is represented as follows:

$$\max_{r \leq k \leq t} \{ \sum_{j=r}^{k} A_j + \sum_{j=k}^{t} B_j \}.$$

Hence, $a(I_i)$ and $b(I_i)$ denote the total idle times of the string I_i on machines II and I, respectively. It is evident that

$$a(I_i) - b(I_i) = \sum_{j \in I_i} (A_j - B_j). \tag{1}$$

We further assume that a partial ordering between strings is given by a binary relationship called precedence. If for some reason the processing of the string I_i must be completed on each machine before the processing of I_j begins on the machine, then I_i is said tc precede I_j and is written $I_i > I_j$. A special case of this relationship exists when there are no intervening strings. If $I_i > I_j$, and if there is no strings, I_k, such that $I_i > I_k > I_j$, then I_i is said to directly precede I_j and is written $I_i >> I_j$. It is convenient to display these relationships on a precedence graph such as G* shown in Fig. 2.1. The nodes of the graph represent the strings and the arrows represent "directly-precedes" relationships between the strings.

The problem we consider is to obtain a schedule minimizing the total elapsed time from the start of the first job on machine I until the end of the last job on machine II subject to given precedence constraints which are represented by a precedence graph G = (X, U), where U denotes the set of arrows.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

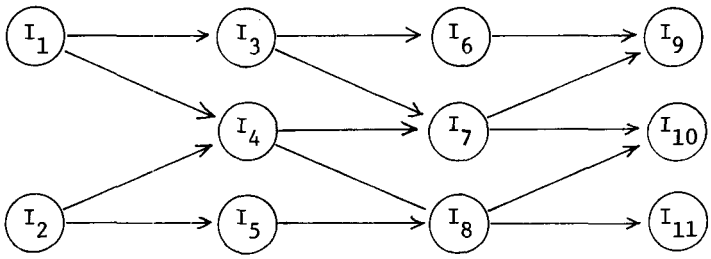

Fig. 2.1. Precedence graph G*

3. Theorems and Proofs

In this section, we give several theorems as a basis of the algorithm to obtain a schedule which minimizes the total elapsed time subject to precedence constraints.

Lemma 1. For minimizing the total elapsed time subject to precedence constraints, it is sufficient to consider only schedules in which the same string order occurs on machines I and II.

Proof: If a feasible schedule Π' does not have the same string order on machines I and II, then somewhere in the schedule for machine I there must be a string I_j that is sequenced directly before a string I_k, where I_j follows I_k possibly with intervening strings on machine II. Obviously, the positions of these two strings can be reversed on machine I without requiring an increase in the starting time of any string on machine II, and hence, the total elapsed time is not increased by the exchange. Since I_k is sequenced before I_j on machine II, this exchange does not imply the infeasibility of the schedule. Thus, it follows that there exists a feasible schedule Π —— with the total elapsed time no greater than that of Π' —— in which the strings are sequenced in the same string order on both machines. This terminates our proof.

We denote by $T(\Pi)$ the total elapsed time for a sequence Π. Furthermore, for strings $I_i = (r, s, \ldots, t)$ and $I_j = (u, v, \ldots, w)$, we denote the string $(r, s, \ldots, t, u, v, \ldots, w)$ by (I_i, I_j). It is easily verified that

$$a\{(I_i, I_j)\} = \max\{a(I_i),\ a(I_i) + a(I_j) - b(I_i)\}$$

and

$$b\{(I_i, I_j)\} = \max\{b(I_i) + b(I_j) - a(I_j),\ b(I_j)\}.$$

Theorem 1. Let I_1^*, I_2^*, I_3^* and I_4^* be strings. If

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

$$\min\{a(I_2^*),\ b(I_3^*)\} \leqq \min\{a(I_3^*),\ b(I_2^*)\},$$

then

$$T\{(I_1^*,\ I_2^*,\ I_3^*,\ I_4^*)\} \leqq T\{(I_1^*,\ I_3^*,\ I_2^*,\ I_4^*)\},$$

where either I_1^* or I_4^* may be empty.

Proof: For simplicity, we set $I_1^* = (1,\ 2,\ \dots,\ u)$, $I_2^* = (u+1,\ u+2,\ \dots,\ v)$, $I_3^* = (v+1,\ v+2,\ \dots,\ w)$ and $I_4^* = (w+1,\ w+2,\ \dots,\ n)$, where $u < v < w < n$. Then $T\{(I_1^*,\ I_2^*,\ I_3^*,\ I_4^*)\}$ and $T\{(I_1^*,\ I_3^*,\ I_2^*,\ I_4^*)\}$ can be represented as follows:

$$T\{(I_1^*,\ I_2^*,\ I_3^*,\ I_4^*)\} = \max_{1\leqq k\leqq n} F_k$$

$$T\{(I_1^*,\ I_3^*,\ I_2^*,\ I_4^*)\} = \max_{1\leqq k\leqq n} F_k',$$

where

$$F_k = \sum_{i=1}^{k} A_i + \sum_{i=k}^{n} B_i$$

and

$$F_k' = \begin{cases} F_k, & \text{for } 1 \leqq k \leqq u \text{ or } w+1 \leqq k \leqq n, \\ \sum_{i=1}^{u} A_i + \sum_{i=v+1}^{w} A_i + \sum_{i=u+1}^{k} A_i + \sum_{i=k}^{v} B_i + \sum_{i=w+1}^{n} B_i, & \text{for } u+1 \leqq k \leqq v, \\ \sum_{i=1}^{u} A_i + \sum_{i=v+1}^{k} A_i + \sum_{i=k}^{w} B_i + \sum_{i=u+1}^{v} B_i + \sum_{i=w+1}^{n} B_i, & \text{for } v+1 \leqq k \leqq w. \end{cases}$$

Hence, if

$$\max_{u+1\leqq k\leqq w} F_k \leqq \max_{u+1\leqq k\leqq w} F_k', \tag{2}$$

then we get

$$T\{(I_1^*,\ I_2^*,\ I_3^*,\ I_4^*)\} \leqq T\{(I_1^*,\ I_3^*,\ I_2^*,\ I_4^*)\}.$$

While, (2) is equivalent to

$$\max\left\{\begin{array}{l} \max_{u+1\leqq k\leqq v}\left[\sum_{i=u+1}^{k} A_i + \sum_{i=k}^{v} B_i + \sum_{i=v+1}^{w} B_i\right] \\ \max_{v+1\leqq k\leqq w}\left[\sum_{i=u+1}^{v} A_i + \sum_{i=v+1}^{k} A_i + \sum_{i=k}^{w} B_i\right] \end{array}\right\} \tag{3}$$

$$\max\left\{\begin{array}{l} \max_{v+1\leqq k\leqq w}\left[\sum_{i=v+1}^{k} A_i + \sum_{i=k}^{w} B_i + \sum_{i=u+1}^{v} B_i\right] \\ \max_{u+1\leqq k\leqq v}\left[\sum_{i=v+1}^{w} A_i + \sum_{i=u+1}^{k} A_i + \sum_{i=k}^{v} B_i\right] \end{array}\right\}.$$

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

By subtracting $\sum_{i=u+1}^{w} B_i$ from each term in (3), it becomes

$$\max\left\{\begin{array}{c}\max\limits_{u+1\leq k\leq v}\left[\sum\limits_{i=u+1}^{k} A_i - \sum\limits_{i=u+1}^{k-1} B_i\right] \\ \max\limits_{v+1\leq k\leq w}\left[\sum\limits_{i=v+1}^{k} A_i - \sum\limits_{i=v+1}^{k-1} B_i\right] + \sum\limits_{i=u+1}^{v}(A_i - B_i)\end{array}\right\}$$

$$\leq \max\left\{\begin{array}{c}\max\limits_{v+1\leq k\leq w}\left[\sum\limits_{i=v+1}^{k} A_i - \sum\limits_{i=v+1}^{k-1} B_i\right] \\ \max\limits_{u+1\leq k\leq v}\left[\sum\limits_{i=u+1}^{k} A_i - \sum\limits_{i=u+1}^{k-1} B_i\right] + \sum\limits_{i=v+1}^{w}(A_i - B_i)\end{array}\right\}$$

or simply

$$\max\{a(I_2^*),\ a(I_3^*) + \sum_{i=u+1}^{v}(A_i - B_i)\} \tag{4}$$

$$\leq \max\{a(I_3^*),\ a(I_2^*) + \sum_{i=v+1}^{w}(A_i - B_i)\}.$$

By means of (1), (4) can be represented as

$$\max\{a(I_2^*),\ a(I_2^*) + a(I_3^*) - b(I_2^*)\} \leq \max\{a(I_3^*),\ a(I_2^*) + a(I_3^*) - b(I_3^*)\}$$

or

$$\min\{a(I_2^*),\ b(I_3^*)\} \leq \min\{a(I_3^*),\ b(I_2^*)\}. \tag{5}$$

Hence,

$$\max_{u+1\leq k\leq w} F_k \leq \max_{u+1\leq k\leq w} F'_k$$

if and only if (5) holds. This terminates our proof.

It is easily seen that Thoeorem 1 is an extension of the well-known result by Johnson. For a precedence graph G = (X, U), we set

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

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

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

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

P(G) and Q(G) denote the sets of strings which may be performed first and last, respectively, for the problem with the precedence graph G.

Theorem 2. If there is a string I_i in P(G) such that

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

$$a(I_i) \leqq b(I_i)$$

and

$$a(I_i) \leqq a(I_j) \qquad \text{for all } I_j \text{ in } P(G),$$

then there exists a sequence minimizing the total elapsed time, subject to the precedence constraints represented by the precedence graph G, in which the string I_i is ordered in the first position.

Proof: Consider any sequence Π' with a string $I_j \neq I_i$ first. Such a sequence can be represented as follows:

$$\Pi' = (I_j, V, I_i, W),$$

where V and W denote the portions of the sequence occupied by the strings other than I_i and I_j. Suppose we modify this sequence to obtain:

$$\Pi = (I_i, I_j, V, W).$$

If the given precedence constraints are observed by sequence Π', then they are also observed by sequence Π, since neither I_i nor I_j is required to succeed any other string. Since

$$a(I_i) \leqq \min\{a(I_j), \ b(I_i)\},$$

we get

$$a(I_i) \leqq a(I_j) \leqq \max\{a(I_j), \ a(I_j) + a(V) - b(I_j)\} = a\{(I_j, V)\},$$

and so,

$$\min[a(I_i), \ b\{(I_j, V)\}] \leqq \min[a\{(I_j, V)\}, \ b(I_i)].$$

Thus, it follows, from Theorem 1, that $T(\Pi) \leqq T(\Pi')$. This terminates our proof.

The following theorem can be proved in an analogous way.

Theorem 2'. If there is a string I_i in Q(G) such that

$$b(I_i) \leqq a(I_i)$$

and

$$b(I_i) \leqq b(I_j) \qquad \text{for all } I_j \text{ in } Q(G),$$

then there exists a sequence minimizing the total elapsed time, subject to the precedence constraints represented by the precedence graph G, in which the string I_i is ordered in the last position.

Theorem 3. Let Π and Π' be sequences of the strings in $X - \{I_i\}$. If

$$T(\Pi) \leqq T(\Pi'),$$

then

$$T\{(I_i, \Pi)\} \leqq T\{(I_i, \Pi')\}$$

and

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

$$T\{(\Pi, I_i)\} \leqq T\{(\Pi', I_i)\}.$$

Proof: Let n_u be the number of jobs in the string I_u and let A_{uv}, and B_{uv} be the processing times of the v-th job in the string I_u on machines I and II, respectively. Furthermore, we denote by $I_{[k]}$ and $I_{(k)}$ the strings ordered in the k-th position of the sequences Π and Π', respectively. Then $T(\Pi)$ can be represented as follows:

$$T(\Pi) = \max_{1\leqq u\leqq m-1} \max_{1\leqq v\leqq n_{[u]}} \{\sum_{k=1}^{u-1} A^*_{[k]} + \sum_{l=1}^{v} A_{[u]l} + \sum_{l=v}^{n_{[u]}} B_{[u]l} + \sum_{k=u+1}^{m-1} B^*_{[k]}\},$$

where

$$A^*_u = \sum_{v=1}^{n_u} A_{uv}$$

and

$$B^*_u = \sum_{v=1}^{n_u} B_{uv}.$$

We further obtain

$$T\{(I_i, \Pi)\}$$

$$= \max \left\{ \begin{array}{c} \max_{1\leqq v\leqq n_i} [\sum_{l=1}^{v} A_{il} + \sum_{l=v}^{n_i} B_{il} + \sum_{k=1}^{m-1} B^*_{[k]}] \\ \max_{1\leqq u\leqq m-1} \max_{1\leqq v\leqq n_{[u]}} [A^*_i + \sum_{k=1}^{u-1} A^*_{[k]} + \sum_{l=1}^{v} A_{[u]l} + \sum_{l=v}^{n_{[u]}} B_{[u]l} + \sum_{k=u+1}^{m-1} B^*_{[k]}] \end{array} \right\}$$

$$= \max\{ \max_{1\leqq v\leqq n_i} [\sum_{l=1}^{v} A_{il} + \sum_{l=v}^{n_i} B_{il}] + \sum_{k=1}^{m-1} B^*_{[k]}, \quad A^*_i + T(\Pi)\}.$$

Similarly, we have

$$T\{(I_i, \Pi')\} = \max\{ \max_{1\leqq v\leqq n_i} [\sum_{l=1}^{v} A_{il} + \sum_{l=v}^{n_i} B_{il}] + \sum_{k=1}^{m-1} B^*_{(k)}, \quad A^*_i + T(\Pi')\}.$$

Since $\sum_{k=1}^{m-1} B^*_{[k]} = \sum_{k=1}^{m-1} B^*_{(k)}$ and $T(\Pi) \leqq T(\Pi')$, we get $T\{(I_i, \Pi)\} \leqq T\{(I_i, \Pi')\}$. The proof of the second inequality of the theorem is similar.

From Theorems 2 and 3, it follows that if $P(G)$ consists of only one string I_i or there is a string I_i in $P(G)$ such that

$$a(I_i) \leqq b(I_i)$$

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

and

$$a(I_i) \leqq a(I_j) \qquad \text{for all } I_j \text{ in } P(G),$$

then we can obtain an optimal sequence for the problem with the precedence graph G = (X, U) by ordering the string I_i in the first position and by selecting an optimal sequence for the problem with the precedence graph omitted, from G, the node I_i and the arrows starting from I_i. Similarly, if Q(G) consists of only one string I_i or there is a string I_i in Q(G) such that

$$b(I_i) \leqq a(I_i)$$

and

$$b(I_i) \leqq b(I_j) \qquad \text{for all } I_j \text{ in } Q(G),$$

then we can obtain an optimal sequence for the problem with the precedence graph G = (X, U) by ordering I_i in the last position and by selecting an optimal sequence for the problem with the precedence graph omitted, from G, the node I_i and the arrows terminating at I_i.

Theorem 4. Let I_i be a string in X - P(G) such that

$$a(I_i) \leqq \min\{a(I_j),\ b(I_j)\} \qquad \text{for all } I_j \text{ in X.}$$

Then there exists an optimal sequence —— minimizing the total elapsed time subject to the precedence constraints represented by a precedence graph G = (X, U) —— in which a string in $P(I_i, G)$ is ordered directly before I_i.

Proof: Let Π' be any feasible sequence and let I_j be the string ordered last among the strings in $P(I_i, G)$ under the sequence Π', i.e.,

$$\Pi' = (V,\ I_j,\ W,\ I_i,\ Z),$$

where V, W and Z represent the portions of the sequence occupied by the strings other than I_i and I_j, and V contains all the strings in $P(I_i, G) - \{I_j\}$. Suppose we modify this sequence Π' to obtain:

$$\Pi = (V,\ I_j,\ I_i,\ W,\ Z).$$

The given precedence constraints are observed by Π, since they are observed by Π' and the strings in W may be sequenced after the string I_i. Regarding W = $(K_1, K_2, \ldots, K_s)$, where each K_l is a string in X, as a string, we have

$$a(W) = \max_{1 \leqq u \leqq s} \{ \sum_{l=1}^{u} a(K_l) - \sum_{l=1}^{u-1} b(K_l) \}$$

and

$$b(W) = \max_{1 \leqq u \leqq s} \{ \sum_{l=u}^{s} b(K_l) - \sum_{l=u+1}^{s} a(K_l) \}.$$

By assumption

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

$$a(I_i) \leqq \min\{a(K_l),\ b(K_l)\} \qquad \text{for } l = 1, 2, \ldots, s$$

and

$$a(I_i) \leqq b(I_i),$$

and hence, we get

$$a(I_i) \leqq \min\{a(W),\ b(I_i)\}.$$

Thus, from Theorem 1, it follows that the total elapsed time is no greater for sequence Π than for Π'. This terminates our proof.

By an entirely analogous argument, we obtain the following theorem:

Theorem 4'. Let I_i be a string in $X - Q(G)$ such that

$$b(I_i) \leqq \min\{a(I_j),\ b(I_j)\} \qquad \text{for all } I_j \text{ in } X.$$

Then there exists an optimal sequence —— minimizing the total elapsed time subject to the precedence constraints represented by a precedence graph $G = (X, U)$ —— in which a string I_j in $Q(I_i, G)$ is ordered directly after I_i.

4. An Algorithm

In this section, we give an algorithm to obtain a sequence minimizing the total elapsed time subject to the precedence constraints which are represented by a precedence graph G.

Suppose it is decided that the strings I_i and I_j with $I_j \in Q(I_i, G)$ —— or equivalently $I_i \in P(I_j, G)$ —— are processed successively on the machines, i.e., I_i and I_j should be regarded as to constitute a string (I_i, I_j). Then for satisfying the precedence relations represented by G, the strings which must be processed before the strings I_i and/or I_j must be processed before the string (I_i, I_j), and the strings which must be processed after the strings I_i and/or I_j must be processed after the string (I_i, I_j). Hence, the precedence graph $G = (X, U)$ must be altered by the following procedure†:

(i) Change X into $X - \{I_i\} - \{I_j\} + \{(I_i, I_j)\}$.

(ii) Change each arrow from I_k to I_i into an arrow from I_k to (I_i, I_j).

(iii) Change each arrow from I_j to I_k into an arrow from (I_i, I_j) to I_k.

(iv) Change each arrow from I_k to I_j with $I_k \neq I_i$ into an arrow from I_k to (I_i, I_j).

(v) Change each arrow from I_i to I_k with $I_k \neq I_j$ into an arrow from (I_i, I_j) to I_k.

---

† (i) to (v) mean to obtain a condensation of a graph G.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

(vi) If there are arrows which are implied by a set of arrows after (ii) to (v), omit the implied precedence arrows.

We denote the resultant graph by $G|\{I_i, I_j\}$. To illustrate the procedure, we assume that the strings I_4 and I_7 in the precedence graph G^* as was indicated in Fig. 2.1 constitute a string (I_4, I_7). Then the string I_3 must be processed before the string (I_4, I_7) since I_3 must be processed before the string I_7, and the string I_8 must be processed after the string (I_4, I_7) since I_8 must be processed after the string I_4. There is an arrow from I_1 to I_4 in G^*. However, the arrow from I_1 to (I_4, I_7) must be eliminated in $G^*|\{I_4, I_7\}$ since there are arrows from I_1 to I_3 and from I_3 to (I_4, I_7). Similarly, the arrow from (I_4, I_7) to I_{10} must be omitted since there are arrows from (I_4, I_7) to I_8 and from I_8 to I_{10}. Thus, we obtain $G^*|\{I_4, I_7\}$ as indicated in Fig. 4.1.

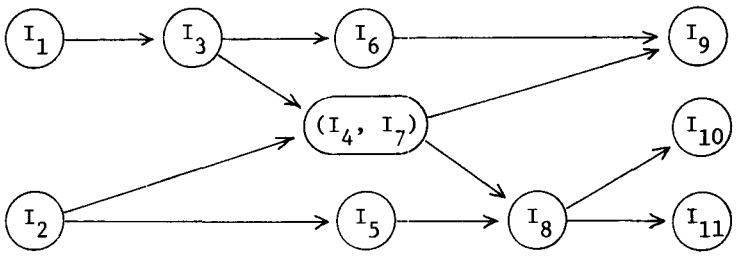

Fig. 4.1. Precedence graph $G^*|\{I_4, I_7\}$

We now define a set $C(G)$ for a precedence graph $G = (X, U)$ by the following algorithm:

Step 1. Set

$$B = A = \phi,$$
$$I_i^* = I_i \qquad \text{for } i = 1, 2, \ldots, m,$$
$$a(I_i^*) = a(I_i) \qquad \text{for } i = 1, 2, \ldots, m,$$
$$b(I_i^*) = b(I_i) \qquad \text{for } i = 1, 2, \ldots, m,$$
$$X^* = \{I_1^*, I_2^*, \ldots, I_m^*\}$$

and

$$G^* = (X^*, U).$$

Go to Step 2.

Step 2. If nodes in the current precedence graph G^* are totally ordered, then stop: $\Pi = (B, C, A)$ is an element in $C(G)$, where C denotes the order of nodes in G^*. Otherwise, go to Step 3.

Step 3. If there is only one node I_i^* in $P(G^*)$ or there is a node I_i^* in $P(G^*)$

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

such that

$a(I_i^*) \leqq b(I_i^*)$

and

$a(I_i^*) \leqq a(I_j^*)$ for all I_j^* in $P(G^*)$,

then set $B = (B, I_i^*)$ and $X^* = X^* - \{I_i^*\}$. Moreover, omit, from G^*, the arrows starting from I_i^* and let the resultant graph G^*. Return to Step 2. If there are no such strings in $P(G^*)$, then go to Step 4.

Step 4. If there is only one node I_i^* in $Q(G^*)$ or there is a node I_i^* in $Q(G^*)$ such that

$b(I_i^*) \leqq a(I_i^*)$

and

$b(I_i^*) \leqq b(I_j^*)$ for all I_j^* in $Q(G^*)$,

then set $A = (I_i^*, A)$ and $X^* = X^* - \{I_i^*\}$. Moreover, omit the arrows terminating at I_i^* in G^* and let the resultant graph G^*. Return to Step 2. If there are no such strings in $Q(G^*)$, then go to Step 5.

Step 5. Obtain the minimum value of the $a(I_i^*)$'s and $b(I_i^*)$'s in the current precedence graph G^*. If it is $a(I_i^*)$, then go to Step 6. If it is $b(I_i^*)$, then go to Step 7.

Step 6. Fix a node I_j^* in $P(I_i^*, G^*)$ and make up a new node (I_j^*, I_i^*). Moreover, set

$G_j = G^*|\{I_j^*, I_i^*\}$,

$a\{(I_j^*, I_i^*)\} = a(I_j^*)$

and

$b\{(I_j^*, I_i^*)\} = b(I_j^*) + b(I_i^*) - a(I_i^*)$.

Put $G^* = G_j$ for each I_j^* in $P(I_i^*, G^*)$ and return to Step 2.

Step 7. Fix a node I_j^* in $Q(I_i^*, G^*)$ and make up a new node (I_i^*, I_j^*). Moreover, set

$G_j = G^*|\{I_i^*, I_j^*\}$,

$a\{(I_i^*, I_j^*)\} = a(I_i^*) + a(I_j^*) - b(I_i^*)$

and

$b\{(I_i^*, I_j^*)\} = b(I_j^*)$.

Put $G^* = G_j$ for each I_j^* in $Q(I_i^*, G^*)$ and return to Step 2.

We call each sequence in $C(G)$ a candidate sequence for a precedence graph G. It is apparent, from theorems in the previous section, that there is an optimal sequence in $C(G)$. Hence, we can obtain an optimal sequence by calculating the total elapsed time for each candidate sequence.

Remarks

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

1. In the above algorithm, (I_j^*, I_i^*) and (I_i^*, I_j^*) in Steps 6 and 7, respectively, are treated as a string. Thus, in the later steps, each node may contain two or more original strings. However, we consider these strings as to constitute a string.
2. If, in Step 5, two or more $a(I_i^*)$'s and/or $b(I_i^*)$'s take the minimum value, then any one among them may be selected. While, it is more effective to select the node, among them, with the fewest number of arrows terminating at I_i^* in the case $a(I_i^*)$ is minimum and starting from I_i^* in the case $b(I_i^*)$ is minimum. If some different nodes are chosen in Step 5, then we may get different sets of candidate sequences. However, both sets contain optimal sequences. Thus, we may obtain a set C(G) of candidate sequences produced by the algorithm and need not obtain all the candidate sequences which belong to some C(G).
3. In Step 6, I_j^* is sequenced last among the strings in $P(I_i^*, G^*)$. Hence, strings I_j^* and I_k^* in $P(I_i^*, G^*)$ generate different candidate sequences. Similarly, in Step 7, strings I_j^* and I_k^* in $Q(I_i^*, G^*)$ generate different candidate sequences. It may occur, however, that the sequences $(I_1^*, I_2^*, I_3^*, I_4^*)$ and $(I_1^*, I_3^*, I_2^*, I_4^*)$ are elements in a C(G). In such a case, we can eliminate from consideration some of the candidate sequences by means of Theorem 1.
4. If each string consists of a job and every permutation of jobs is feasible, then our algorithm terminates within Steps 1, 2, 3 and 4, and thus, we get a set C(G) with one element. This is just the algorithm proposed by Johnson [2].
5. If the original precedence graph consists of parallel chains such that the strings in each chain are totally ordered, then $P(I_i^*, G^*)$ and $Q(I_i^*, G^*)$ contain at most one node, and hence, it suffices to get one G_j^* in Steps 6 and 7. Thus, we have C(G) with one candidate sequence, and so, we can obtain an optimal sequence directly.

5. An Example

In this section, we illustrate the procedure given in Section 4 with a simple example. Suppose that there are nine strings with the precedence graph G_1 as indicated in Fig. 5.1. We assume that each string I_i consists of a job i and that the processing times A_i and B_i are given in Table 1. We obtain $C(G_1)$ by the algorithm mentioned in the previous section as follows:

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

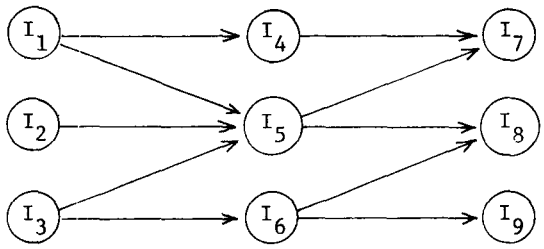

Fig. 5.1. Precedence graph G_1

Table 1. Processing times of the jobs

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

Step 1. We set

$$B = A = \phi,$$
$$I_i = \{i\}, \qquad \text{for } i = 1, 2, \ldots, 9,$$
$$a(I_i) = A_i, \qquad \text{for } i = 1, 2, \ldots, 9,$$
$$b(I_i) = B_i, \qquad \text{for } i = 1, 2, \ldots, 9,$$
$$X^* = \{I_1, I_2, \ldots, I_9\}$$

and

$$G^* = G_1.$$

(In this example, we denote a string composed of several original strings by the set of strings which make up the string.)

Step 4. Since

$$b(I_7) \leqq \min\{a(I_7),\ b(I_8),\ b(I_9)\},$$

we set $A = (I_7)$. Omitting, from G_1, the node I_7 and the arrows from I_4 to I_7 and from I_5 to I_7, we get the precedence graph G_2 which is shown in Fig. 5.2. (Note: in Figs. 5.2 to 5.11, the total idle times $a(I_i^*)$ and $b(I_i^*)$ on machines II and I are shown above and below, respectively, the description of the string I_i^*.)

Step 4. Since

$$b(I_4) \leqq \min\{a(I_4),\ b(I_8),\ b(I_9)\},$$

we set $A = (I_4, I_7)$. Omitting, from the current precedence graph G_2, the node I_4 and the arrow from I_1 to I_4, we get the precedence graph

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

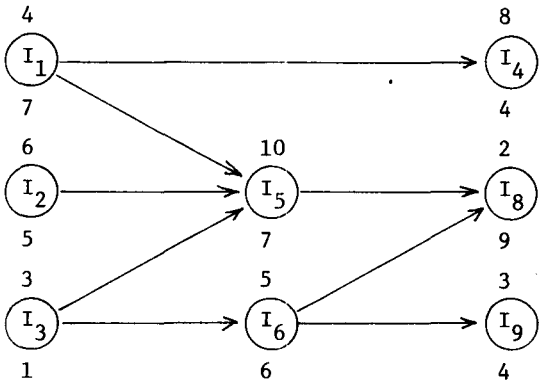

Fig. 5.2. Precedence graph G_2

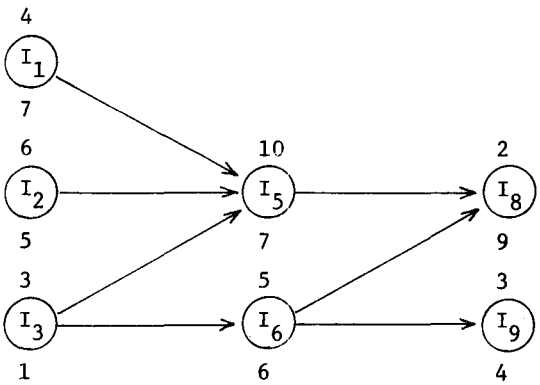

Fig. 5.3. Precedence graph G_3

G_3 as indicated in Fig. 5.3.

Step 5. The minimum value of the total idle times in G_3 is one which is attained by $b(I_3)$, and hence, we go to Step 7.

Step 7. Since $Q(I_3, G_3) = \{I_5, I_6\}$, we make up two precedence graphs $G_4 = G_3|\{I_3, I_5\}$ and $G_5 = G_3|\{I_3, I_6\}$ which are shown in Figs. 5.4 and 5.5, respectively.

First, we shall get candidate sequences from G_4.

Step 3. Since

$$a(I_1) \leqq \min\{a(I_2),\ b(I_1)\},$$

we set $B = (I_1)$. Omitting, from G_4, the node I_1 and the arrow from I_1 to (I_3, I_5), we get the precedence graph G_6 which is represented in Fig. 5.6.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

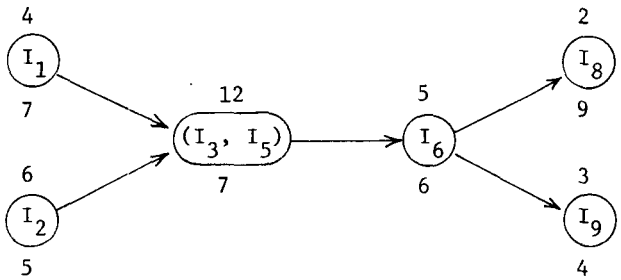

Fig. 5.4. Precedence graph $G_4 = G_3|\{I_3, I_5\}$

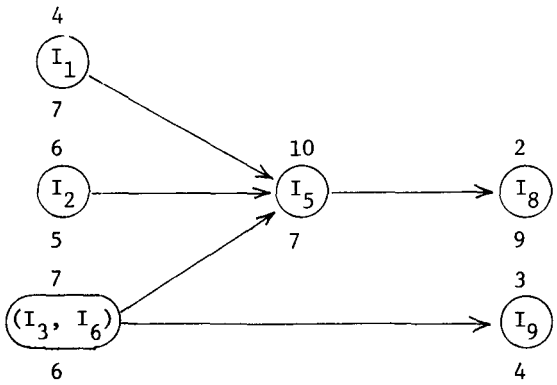

Fig. 5.5. Precedence graph $G_5 = G_3|\{I_3, I_6\}$

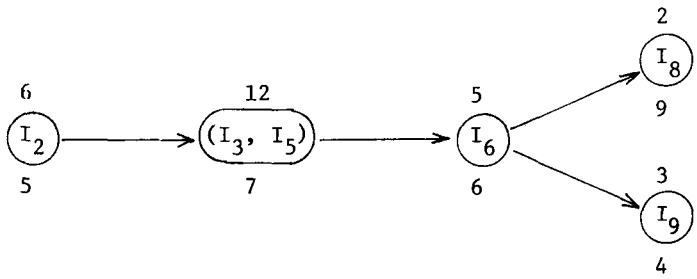

Fig. 5.6. Precedence graph G_6

Step 3. Repeating Step 3, we have $B = (I_1, I_2, I_3, I_5, I_6)$ and the precedence graph G_7 which is shown in Fig. 5.7.

Step 3. In the current precedence graph G_7, it is apparent that I_8 should be ordered before I_9. Thus, we get a candidate sequence $\Pi_1^* = (1, 2, 3, 5, 6, 8, 9, 4, 7)$ for G_1.

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

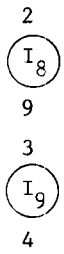

Fig. 5.7. Precedence graph G_7

Next, we shall get candidate sequences from G_5.

Step 3. Since

$$a(I_1) \leqq \min[b(I_1),\ a(I_2),\ a\{(I_3,\ I_6)\}],$$

we set $B = (I_1)$, and get the precedence graph G_8 which is represented in Fig. 5.8.

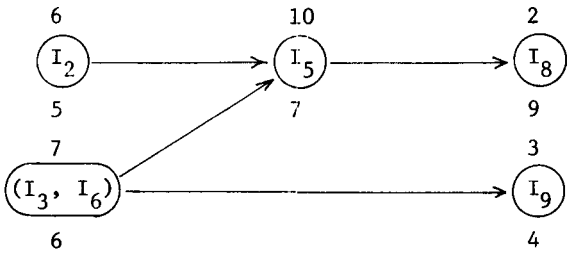

Fig. 5.8. Precedence graph G_8

Step 5. Since $a(I_8)$ takes the minimum value of the total idle times in G_8, we go to Step 6.

Step 6. Combining the strings I_5 and I_8, we get the precedence graph G_9 which is shown in Fig. 5.9.

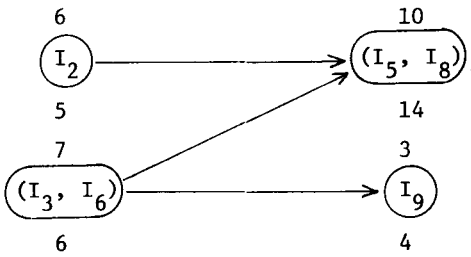

Fig. 5.9. Precedence graph G_9

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

Step 5. Since $a(I_9)$ takes the minimum value of the total idle times in the current precedence graph G_9, we again go to Step 6.

Step 6. Constituting a string (I_3, I_6, I_9), we get the precedence graph G_{10} as indicated in Fig. 5.10.

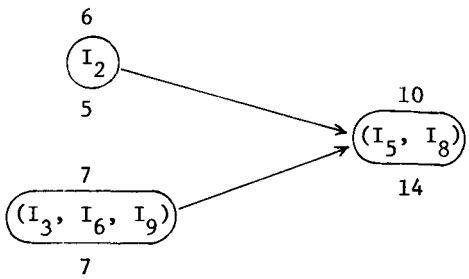

Fig. 5.10. Precedence graph G_{10}

Step 4. Since $Q(G_{10}) = \{(I_5, I_8)\}$, we set $A = (I_5, I_8, I_4, I_7)$ and get the precedence graph G_{11} as indicated in Fig. 5.11.

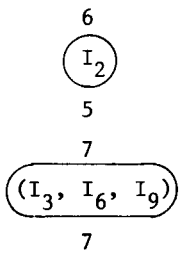

Fig. 5.11. Precedence graph G_{11}

Step 3. Since

$$\min[a\{(I_3, I_6, I_9)\},\ b(I_2)] \leqq \min[a(I_2),\ b\{(I_3, I_6, I_9)\}],$$

(I_3, I_6, I_9) should be processed before I_2, and hence, we get a candidate sequence $\Pi_2^* = (1, 3, 6, 9, 2, 5, 8, 4, 7)$ for the precedence graph G_1. Thus, we obtain $C(G_1) = \{\Pi_1^*, \Pi_2^*\}$.

It is easily calculated that $T(\Pi_1^*) = 56$ and $T(\Pi_2^*) = 54$, and hence, Π_2^* is an optimal sequence for the problem with the precedence graph G_1 and the processing times given in Table 1.

At the first glance, it seems that there are unusually few number of candidate sequences considering that there are 720 feasible sequences —— with

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.

the same string order on machines I and II —— for the precedence graph G_1. However, there are not so many candidate sequences as one might suppose. We solved 50 problems with the precedence graph G_1. In these problems, we assumed each string to be consisted of a job and we generated processing times A_i and B_i from an uniformly distributed random number over (0.0, 10.0). We got 1.66 as the mean number of the candidate sequences in each $C(G_1)$. The maximum number of the candidate sequences was seven and we got only one candidate sequence in 32 problems. Thus, we directly obtained an optimal schedule in more than 60% cases.

Acknowledgements

The author would like to express his sincere thanks to Professor T. Nishida and M. Sakaguchi for their continuing guidances and encouragement. He wishes to thank Assistant Professor R. Manabe and the referees for their helpful comments and suggestions.

References

1. Conway, R. W., Maxwell, W. L., and Miller, L. W.: *Theory of Scheduling*, Addison-Wesley, Reading Mass., 1967.
2. Johnson, S. M.: Optimal Two- and Three-Stage Production Schedules with Setup Times Included. *Naval Research Logistics Quarterly*, Vol. 1 (1954), pp. 61-68.
3. Kurisu, T.: Two-Machine Scheduling under Required Precedence among Jobs. *Journal of the Operations Research Society of Japan*, Vol. 20 (1976), pp. 1-13.
4. Lawler, E. L.: Optimal Sequencing of a Single Machine Subject to Precedence Constraints. *Management Science*, Vol. 19 (1973), pp. 544-546.
5. Sidney, J. B.: Decomposition Algorithms for Single-Machine Sequencing with Precedence Relations and Deferral Costs. *Operations Research*, Vol. 23 (1975), pp. 283-298.

Tadashi KURISU: Department of Applied Physics, Faculty of Engineering
Osaka University, Suita
Osaka, 565, Japan

Copyright © by ORSJ. Unauthorized reproduction of this article is prohibited.