Jounal of the Operations Research
Society of Japan
Vol. 22, No. 1 March, 1979

SINGLE-MACHINE SEQUENCING WITH PRECEDENCE CONSTRAINTS AND DEFERRAL COSTS

Tadashi Kurisu
Osaka University

(Received April 13, 1978; Revised August 30, 1978)

Abstract This paper deals with the problem of sequencing n jobs on a single machine. Associated with each job is its processing time and its deferral rate. In addition, a general precedence relation exists among jobs. The problem is to find the job sequence which is consistent with the precedence relation so as to minimize the sum of the deferral cost. This problem has been considered by Sidney and he has presented a decomposition algorithm. In this paper, it is shown that Sidney's algorithm is not effective for some cases. A new algorithm is presented and it is proved that the algorithm produces an optimal sequence. An illustrative example is shown in the final section.

1. Introduction

We consider the problem of scheduling a set of jobs $J = \{1, 2, \ldots, n\}$ to be processed on a single machine. Associated with job i is its processing time p_i and its deferral rate w_i (> 0). For a given permutation Π of n jobs, the flow time of job i, denoted $F_i(\Pi)$, is the processing time p_i of job i plus the processing times p_k of all jobs k that are ordered before i in the permutation Π. The problem is to find a permutation Π of the jobs in J, such that the total cost, given by $TC(\Pi) = \sum_{i=1}^{n} w_i F_i(\Pi)$ is minimized. If every permutation of n jobs is feasible, then the optimal sequence is determined by computing the ratio p_i/w_i for each job i and then ordering the jobs in the order of non-decreasing ratio. This result is due originally to Smith [6].

In this paper, we consider the above problem when some of the permutations of jobs are prohibited either by technological constraints or by externally imposed policy. The problem now is to find the feasible ordering with the lowest cost. This problem was considered by many researchers. For example, some special cases were solved by Conway, Maxwell and Miller [3], Horn [4], Sidney [5] and Adolphson [1]. However, the general case has not

been solved satisfactorily as will be seen in the next section.

2. Description of Problem

Consider a job-shop consisting of a single machine and a set J = {1, 2, ... , n} of n jobs to be processed on the machine. Associated with job i is its processing time p_i and its deferral rate w_i (> 0). The machine is continuously available from time 0 until all jobs have been completed.

An ordered set of jobs (r, s, ... , t) is said to be a string if and only if the jobs in the set must be processed according to the fixed order r, s, ... , t without interruption between jobs. We assume that the original n jobs have been grouped into m disjoint strings I_1, I_2, ... , I_m and we set X = {I_1, I_2, ... , I_m}. 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 before the processing of the string I_j begins, then I_i is said to precede I_j and is written $I_i > I_j$. If $I_i > I_j$ and there is no string, I_k, such that $I_i > I_k > I_j$, then I_i is said to directly precede I_j and is written $I_i >> I_j$. These precedence relationships are represented by a directed acyclic graph G in which an arrow from node I_i to node I_j indicates that I_i must precede I_j in any feasible ordering.

The problem we consider is to obtain a sequence minimizing the total cost subject to given precedence constraints represented by a precedence graph G = (X, U), where U denotes the set of arrows. This general problem was considered by Sidney [5]. (Note: in his model, each string consists of a job.) He presented the following decomposition algorithm and it is proved that a permutation is optimal if and only if it can be generated by the algorithm:

Decomposition algorithm

Step 1. Consider all initial sets among the unscheduled jobs and find the largest initial set S* that minimizes the quantity $\sum_{i \in S} p_i / \sum_{i \in S} w_i$.

Step 2. Find an optimal order for the jobs in S* and add the jobs in S* to the sequence.

Step 3. Remove the jobs in S* from consideration and return to Step 1 until all jobs are scheduled.

Sidney does not give any method for finding an optimal order for the jobs in S*. As Baker [2] pointed out, the general problem is not truly solved until efficient mechanisms have been designed for the following two steps:

1. Identification of S* (step 1).

2. Sequencing of jobs in S* (Step 2).
These are fairly difficult to perform efficiently in the general case as will be seen in Example 1.

Example 1. Suppose that there are nine strings with the precedence graph G_1 as indicated in Fig. 1. We assume that each string I_i consists of a job i

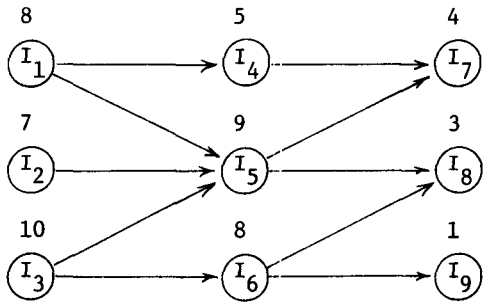

Fig. 1. Precedence Graph G_1

Table 1. Initial set S and $\rho(S) = \sum_{i \in S} p_i / \sum_{i \in S} w_i$

S	ρ(S)	S	ρ(S)
{1}	8.0	{2, 3, 6, 9}	6.5
{2}	7.0	{1, 2, 3, 4, 5}	7.8
{3}	10.0	{1, 2, 3, 4, 6}	7.6
{1, 2}	7.5	{1, 2, 3, 5, 6}	8.4
{1, 3}	9.0	{1, 2, 3, 6, 9}	6.8
{1, 4}	6.5	{1, 3, 4, 6, 9}	6.4
{2, 3}	8.5	{1, 2, 3, 4, 5, 6}	7.83
{3, 6}	9.0	{1, 2, 3, 4, 5, 7}	7.17
{1, 2, 3}	8.33	{1, 2, 3, 4, 6, 9}	6.5
{1, 2, 4}	6.67	{1, 2, 3, 5, 6, 8}	7.5
{1, 3, 4}	7.67	{1, 2, 3, 5, 6, 9}	7.17
{1, 3, 6}	8.67	{1, 2, 3, 4, 5, 6, 7}	7.29
{2, 3, 6}	8.33	{1, 2, 3, 4, 5, 6, 8}	7.14
{3, 6, 9}	6.33	{1, 2, 3, 4, 5, 6, 9}	6.86
{1, 2, 3, 4}	7.5	{1, 2, 3, 5, 6, 8, 9}	6.57
{1, 2, 3, 5}	8.5	{1, 2, 3, 4, 5, 6, 7, 8}	6.75
{1, 2, 3, 6}	8.25	{1, 2, 3, 4, 5, 6, 7, 9}	6.5
{1, 3, 4, 6}	7.75	{1, 2, 3, 4, 5, 6, 8, 9}	6.38
{1, 3, 6, 9}	6.75	{1, 2, 3, 4, 5, 6, 7, 8, 9}	6.11

and that deferral rate of each job is equal to one. Processing time of job i is shown, in Fig. 1, above the description of the string $I_i = \{i\}$.

Step 1. There are 38 initial sets as shown in Table 1. From the table, we read that $S^* = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ minimizes $\sum_{i \in S} p_i / \sum_{i \in S} w_i$.

Step 2. Finding an optimal sequence for the jobs in S^* is just the original problem. Thus, the algorithm presented by Sidney is not applicable to this problem.

Furthermore, Adolphson [1] considered the same problem and gave an algorithm. His algorithm terminates when either the optimal sequence is found or when none of the transformations can be applied (in which case, the optimal sequence can not be found). An example of the latter case is shown in his paper.

3. Theorems and a New Algorithm

In this section, we give several theorems and a new algorithm to obtain a sequence minimizing the total cost subject to precedence constraints represented by a precedence graph $G = (X, U)$.

We set

$$p(I_i) = \sum_{j \in I_i} p_j,$$

$$w(I_i) = \sum_{j \in I_i} w_j$$

and

$$\rho(I_i) = p(I_i)/w(I_i).$$

Theorem 1. (Sidney [5], Lemma 1) Let Π and Π' be permutations of strings in $X - \{I_i\}$. If

$$TC(\Pi) \leqq TC(\Pi'),$$

then

$$TC\{(I_i, \Pi)\} \leqq TC\{(I_i, \Pi')\}$$

and

$$TC\{(\Pi, I_i)\} \leqq TC\{(\Pi', I_i)\}.$$

Lemma 1. (Sidney [5], Lemma 2) Let $\Pi = (I_1, I_2, I_3, I_4)$ and $\Pi' = (I_1, I_3, I_2, I_4)$ be feasible sequences, where I_1 or I_4 or both may be empty. Then $TC(\Pi) \leqq TC(\Pi')$ if and only if $\rho(I_2) \leqq \rho(I_3)$.

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

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

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

and

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

Theorem 2. If

$$\rho(I_i) \leqq \rho(I_j) \qquad \text{for all } I_j \text{ in } X,$$

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

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

$$\Pi' = (V_1, I_j, V_2, I_i, V_3),$$

where V_1, V_2 and V_3 represent the portions of the sequence occupied by the strings other than I_i and I_j, and V_1 contains all the strings which belong to $P(I_i, G) - \{I_j\}$. If V_2 is empty, nothing remains to prove. If V_2 is not empty, we modify this sequence Π' to obtain:

$$\Pi = (V_1, I_j, I_i, V_2, V_3).$$

The given precedence constraints are observed by Π, since they are observed by Π' and the strings in V_2 may be ordered after the string I_i. By assumption,

$$\rho(I_i) \leqq \rho(I_k) = \frac{p(I_k)}{w(I_k)} \qquad \text{for all } I_k \text{ in } V_2,$$

and thus,

$$\rho(I_i) \leqq \frac{\sum_{I_k \in V_2} p(I_k)}{\sum_{I_k \in V_2} w(I_k)} = \rho(V_2).$$

Therefore, from Lemma 1, we have $TC(\Pi) \leqq TC(\Pi')$. Thus, Π is also an optimal sequence. This completes our proof.

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

Theorem 3. If

$$\rho(I_i) \geqq \rho(I_j) \qquad \text{for all } I_j \text{ in } X,$$

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

Let $I_i = (r, s, \ldots, t)$ and $I_j = (u, v, \ldots, w)$ be strings. We denote a new string $(r, s, \ldots, t, u, v, \ldots, w)$ by (I_i, I_j). Suppose it is decided

that the jobs in I_j are ordered directly after the jobs in I_i where $I_j \in Q(I_i, G)$ —— or equivalently $I_i \in P(I_j, G)$ ——, then I_i and I_j are regarded as to constitute a string (I_i, I_j). In this case, 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 $X - \{I_i\} - \{I_j\} + \{(I_i, I_j)\}$ and remove the arrow from I_i to 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.

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

We denote the resultant graph by $G|\{I_i, I_j\}$.

We now generate a set C(G) from a precedence graph $G = (X, U)$ by the following algorithm. In Theorem 4, we prove that an optimal sequence —— which minimizes the total cost subject to precedence constraints represented by a precedence graph G —— exists in C(G).

Algorithm.

Step 0. Set

$$B = A = \phi,$$

$$I_i^* = I_i, \qquad \text{for } i = 1, 2, \ldots, m,$$

$$p(I_i^*) = \sum_{j \in I_i^*} p_j, \qquad \text{for } i = 1, 2, \ldots, m,$$

$$w(I_i^*) = \sum_{j \in I_i^*} w_j, \qquad \text{for } i = 1, 2, \ldots, m,$$

$$\rho(I_i^*) = p(I_i^*)/w(I_i^*), \qquad \text{for } i = 1, 2, \ldots, m,$$

$$X^* = \{I_1^*, I_2^*, \ldots, I_m^*\}$$

and

$G^* = (X^*, U^*)$, where U^* denotes the set of arrows between I_i^* and I_j^* corresponding to the arrows between I_i and I_j in U.

Go to Step 1.

Step 1. If there is no arrow in the current precedence graph G^*, then obtain

a sequence Π' by ordering the strings in X* according to the nondecreasing $\rho(I_i^*)$ and let $\Pi = (B, \Pi', A)$ be an element in C(G) and delete G*. If another G* remains yet, repeat Step 1 for the G*, otherwise stop the algorithm. If the current precedence graph G* has some arrows, go to Step 2.

Step 2. If there is only one string I_i^* in P(G*) or there is a string I_i^* in P(G*) such that

$$\rho(I_i^*) \leqq \rho(I_j^*) \qquad \text{for all } I_j^* \text{ in } X^*,$$

then set $B = (B, I_i^*)$. Moreover, remove, from G*, the string I_i^* and the arrows starting from I_i^* and let the resultant graph $G^* = (X^*, U^*)$. Return to Step 1. If there are no such strings in P(G*), then go to Step 3.

Step 3. If there is only one string I_i^* in Q(G*) or there is a string I_i^* in Q(G*) such that

$$\rho(I_i^*) \geqq \rho(I_j^*) \qquad \text{for all } I_j^* \text{ in } X^*,$$

then set $A = (I_i^*, A)$. Moreover, remove, from G*, the string I_i^* and the arrows terminating at I_i^* and let the resultant graph $G^* = (X^*, U^*)$. Return to Step 1. If there are no such strings in Q(G*), then go to Step 4.

Step 4. Find the strings I_i^* and I_j^* in X* such that

$$\rho(I_i^*) \leqq \rho(I_k^*) \leqq \rho(I_j^*) \qquad \text{for all } I_k^* \text{ in } X^*.$$

If $|P(I_i^*, G^*)| \leqq |Q(I_j^*, G^*)|$†, then go to Step 5. Otherwise, go to Step 6.

Step 5. Fix a node I_k^* in $P(I_i^*, G^*)$ and make up a new node (I_k^*, I_i^*). Moreover, set

$$G_k = G^* | \{I_k^*, I_i^*\},$$
$$p\{(I_k^*, I_i^*)\} = p(I_k^*) + p(I_i^*),$$
$$w\{(I_k^*, I_i^*)\} = w(I_k^*) + w(I_i^*)$$

and

$$\rho\{(I_k^*, I_i^*)\} = p\{(I_k^*, I_i^*)\}/w\{(I_k^*, I_i^*)\}.$$

For *each* I_k^* in $P(I_i^*, G^*)$, set $G^* = G_k$ and return to Step 1.

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

$$G_k = G^* | \{I_j^*, I_k^*\},$$
$$p\{(I_j^*, I_k^*)\} = p(I_j^*) + p(I_k^*),$$
$$w\{(I_j^*, I_k^*)\} = w(I_j^*) + w(I_k^*)$$

and

---

† $|S|$ denotes the cardinality of the set S.

$\rho\{(I_j^*, I_k^*)\} = p\{(I_j^*, I_k^*)\}/w\{(I_j^*, I_k^*)\}$.

for *each* I_k^* in $Q(I_j^*, G^*)$, set $G^* = G_k$ and return to Step 1.

Theorem 4. For the problem to minimize the total cost subject to precedence constraints represented by a precedence graph $G = (X, U)$, there exists an optimal sequence in the set $C(G)$.

Proof. Step 1 in Algorithm is justified since there is an optimal sequence in $C(G)$ from the results of Theorem 1 and Lemma 1. In Step 2, from Theorems 1 and 2, we can get an optimal sequence by removing the string I_i^* and the arrows starting from I_i^*. Similarly, Step 3 is justified from Theorems 1 and 3. In Step 5, each of the I_k^* in $P(I_i^*, G^*)$ is considered, and thus, we can get an optimal sequence by means of Theorem 2. Similarly, Step 6 is justified from Theorem 3.

It follows, from Theorem 4, that we can get an optimal sequence by calculating the total cost for each sequence in $C(G)$.

Remarks.

1. If there are two or more I_i^*'s and/or I_j^*'s satisfying the condition in Step 4, then any one among them may be selected. If another string is selected in Step 4, then we may get another set $C(G)$. However, every $C(G)$ contains an optimal sequence, and thus, it is sufficient to obtain one set $C(G)$ generated by the algorithm.
2. If the original precedence graph G is a collection of rooted trees, then $P(I_i^*, G^*)$ contains at most one node, and hence, we get $C(G)$ with one element. Thus, we can obtain an optimal sequence directly. In this case, our algorithm looks easier than the algorithm presented by Horn [4].

4. An Example

In this section, we illustrate the algorithm given in the previous section by solving Example 1.

Step 0. We set

$$I_i^* = I_i = \{i\}, \qquad \text{for } i = 1, 2, \ldots, 9,$$
$$p(I_i) = p_i, \qquad \text{for } i = 1, 2, \ldots, 9,$$
$$w(I_i) = w_i, \qquad \text{for } i = 1, 2, \ldots, 9,$$
$$\rho(I_i) = p(I_i)/w(I_i) \qquad \text{for } i = 1, 2, \ldots, 9$$

and

$$\rho(I_i^*) = \rho(I_i) \qquad \text{for } i = 1, 2, \ldots, 9.$$

Step 4. Since

$\rho(I_9) \leqq \rho(I_k^*) \leqq \rho(I_3)$ for all I_k^* in G_1

and

$|P(I_9, G_1)| < |Q(I_3, G_1)|$,

we go to Step 5.

Step 5. Since $P(I_9, G_1) = \{I_6\}$, it is sufficient to make up only one new string (I_6, I_9), and thus, we get a precedence graph $G_2 = G_1|\{I_6, I_9\}$ as shown in Fig. 2. (Note: in Figs. 2 to 12, $\rho(I_i^*)$ is shown above the description of the string I_i^*.)

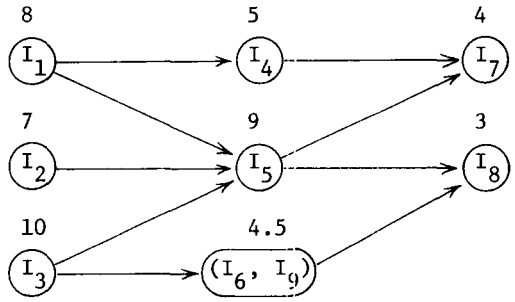

Fig. 2. Precedence Graph G_2

Step 4. Since

$\rho(I_8) \leqq \rho(I_k^*) \leqq \rho(I_3)$

for all I_k^* in the current precedence graph G_2 and

$|P(I_8, G_2)| = |Q(I_3, G_2)|$,

we go to Step 5.

Step 5. Since $P(I_8, G_2) = \{I_5, (I_6, I_9)\}$, we make up two precedence graphs $G_3 = G_2|\{I_5, I_8\}$ and $G_4 = G_2|\{(I_6, I_9), I_8\}$ as indicated in Figs. 3 and 4, respectively.

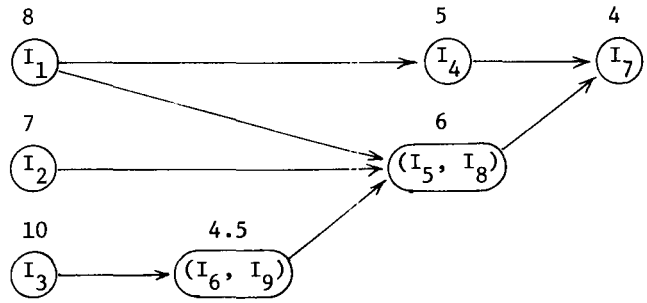

Fig. 3. Precedence Graph $G_3 = G_2|\{I_5, I_8\}$

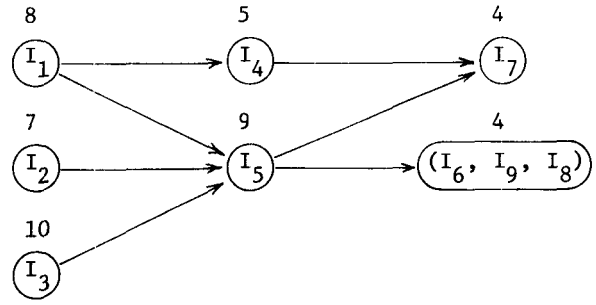

Fig. 4. Precedence Graph $G_4 = G_2|\{(I_6, I_9), I_8\}$

First, starting from G_3, we get sequences belonging to $C(G_1)$.

Step 3. Since $Q(G_3) = \{I_7\}$, we set $A = (I_7)$ and get a precedence graph G_5 as shown in Fig. 5.

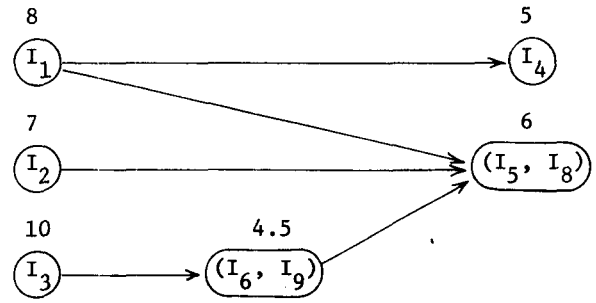

Fig. 5. Precedence Graph G_5

Step 4. Since

$$\rho\{(I_6, I_9)\} \leqq \rho(I_k^*) \leqq \rho(I_3)$$

for all I_k^* in the current precedence graph G_5 and

$$|Q(I_3, G_5)| = |P\{(I_6, I_9), G_5\}|,$$

we go to Step 5.

Step 5. Constituting a new string (I_3, I_6, I_9), we get a precedence graph $G_6 = G_5|\{I_3, (I_6, I_9)\}$ as indicated in Fig. 6.

Step 4. We have

$$\rho(I_4) \leqq \rho(I_k^*) \leqq \rho(I_1)$$

for all I_k^* in the current precedence graph G_6 and

$$|P(I_4, G_6)| < |Q(I_1, G_6)|,$$

and so we again go to Step 5.

Step 5. Since $P(I_4, G_6) = \{I_1\}$, we make up a new string (I_1, I_4) and get a precedence graph G_7 as shown in Fig. 7.

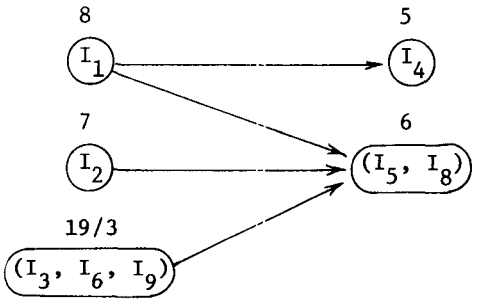

Fig. 6. Precedence Graph $G_6 = G_5|\{I_3, (I_6, I_9)\}$

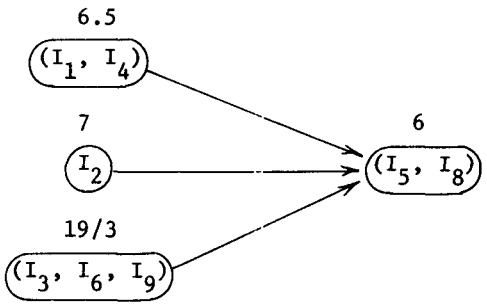

Fig. 7. Precedence Graph G_7

Step 3. Since $Q(G_7) = \{(I_5, I_8)\}$, we set $A = (I_5, I_8, I_7)$ and get a precedence graph G_8 as illustrated in Fig. 8.

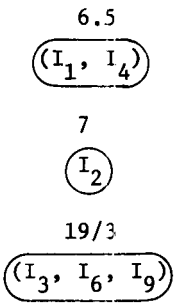

Fig. 8. Precedence Graph G_8

Step 1. It is easily seen that $(I_3, I_6, I_9, I_1, I_4, I_2)$ is an optimal sequence for the precedence graph G_8, and thus, $\Pi_1 = (3, 6, 9, 1, 4, 2, 5, 8, 7)$ is an element in $C(G_1)$.

Next, starting from G_4, we get sequences which belong to $C(G_1)$.

Step 4. Since

$$\rho\{(I_6, I_9, I_8)\} \leqq \rho(I_k^*) \leqq \rho(I_3)$$

for all I_k^* in the current precedence graph G_4 and

$$|P\{(I_6, I_9, I_8), G_4\}| = |Q(I_3, G_4)|,$$

we go to Step 5.

Step 5. We constitute a new string (I_5, I_6, I_9, I_8) and get a precedence graph G_9 as indicated in Fig. 9.

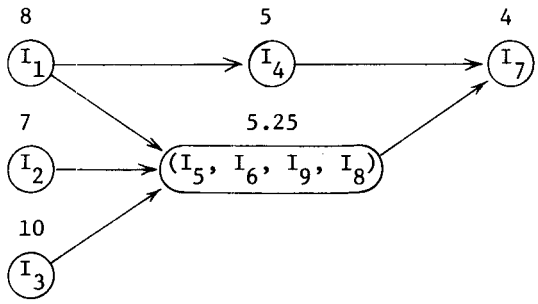

Fig. 9. Precedence Graph G_9

Step 3. As $Q(G_9) = \{I_7\}$, we set $A = (I_7)$ and get a precedence graph G_{10} as indicated in Fig. 10.

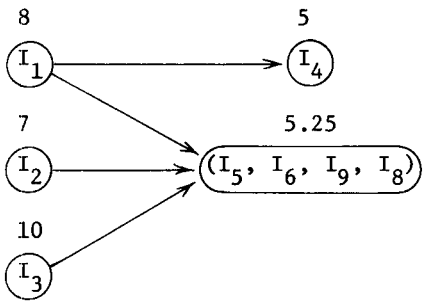

Fig. 10. Precedence Graph G_{10}

Step 4. Since

$$\rho(I_4) \leqq \rho(I_k^*) \leqq \rho(I_3) \qquad \text{for all } I_k^* \text{ in } G_{10}$$

and

$$|P(I_4, G_{10})| = |Q(I_3, G_{10})|,$$

we go to Step 5.

Step 5. Making up a new string (I_1, I_4), we get a precedence graph G_{11} as shown in Fig. 11.

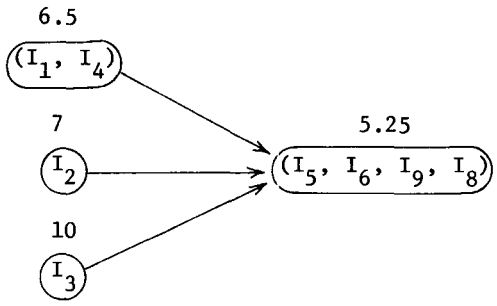

Fig. 11. Precedence Graph G_{11}

Step 3. Since there is only one string (I_5, I_6, I_9, I_8) in $Q(G_{11})$, we set $A = (I_5, I_6, I_9, I_8, I_7)$, and get a precedence graph G_{12} as shown in Fig. 12.

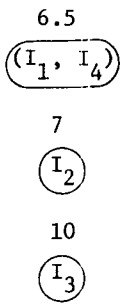

Fig. 12. Precedence Graph G_{12}

Step 1. Apparently, (I_1, I_4, I_2, I_3) is an optimal sequence for G_{12}, and so, $\Pi_2 = (1, 4, 2, 3, 5, 6, 9, 8, 7)$ is an element in $C(G_1)$.

Thus, we have $C(G_1) = \{\Pi_1, \Pi_2\}$. It is easily calculated that $TC(\Pi_1) = 299$ and $TC(\Pi_2) = 311$, and hence, Π_1 is an optimal sequence.

Acknowledgements

The author wishes to thank Professor M. Sakaguchi of Osaka University for his useful suggestions and valuable discussions. He also wishes to acknowledge the referees for their comments, which have greatly improved this paper.

References

[1] Adolphson, D. L.: Single Machine Job Sequencing with Precedence Constraints. *Siam Journal on Computing*, Vol. 6 (1977), pp. 40-54.

[2] Baker, K, R.: *Introduction to Sequencing and Scheduling*, John Wiley, New York, 1974.

[3] Conway, R. W., Maxwell, W. L., and Miller, L. W.: *Theory of Scheduling*, Addison-Wesley, Reading, Mass., 1967.

[4] Horn, W. A.: Single-Machine Job Sequencing with Treelike Precedence Ordering and Linear Delay Penalties. *Siam Journal on Applied Mathematics*, Vol. 23 (1972), pp. 189-202.

[5] Sidney, J. B.: Decomposition Algorithms for Single-Machine Sequencing with Precedence Relations and Deferral Costs. *Operations Research*, Vol. 23 (1975), pp. 283-298.

[6] Smith, W. E.: Various Optimizers for Single-Stage Production. *Naval Research Logistics Quarterly*, Vol. 3 (1956), pp. 59-66.

Tadashi KURISU:
Department of Applied Mathematics
Faculty of Engineering Science
Osaka University
Toyonaka, Osaka, Japan