PARALLEL SEQUENCING AND ASSEMBLY LINE PROBLEMS

T. C. Hu

IBM Research Center, Yorktown, New York

(Received May 5, 1961)

This paper deals with a new sequencing problem in which n jobs with ordering restrictions have to be done by men of equal ability. Assume every man can do any of the n jobs. The two questions considered in this paper are: (1) How to arrange a schedule that requires the minimum number of men so that all jobs are completed within a prescribed time T, and (2) if m men are available, arrange a schedule that completes all jobs at the earliest time.

MANY scheduling (or sequencing) problems can be formulated as follows. Given n jobs with known times to perform each job and with technological ordering restrictions among the jobs, two questions that are often posed are the following.

1. Assume that all jobs must be completed by time T, arrange a schedule that requires the minimum number of men. (It is assumed that men are of equal ability and every man can do any of the n jobs.)

2. If m men are available, arrange a schedule that completes all jobs at the earliest time.

Both problems are proposed in reference 1. Here we consider the simplified version of the above problems that assumes that all jobs require equal time and that when a man finishes a job he can immediately start on another. Lower bounds on shortest time and minimum number of men to complete the jobs are obtained for arbitrary ordering restrictions. When the ordering restrictions form a tree, which is the case in an assembly line problem, a condition is obtained for determining the minimum number of men required to finish the jobs within a prescribed time, and a very simple algorithm is found that will finish all jobs at the earliest time with a given number of men.

Here we use the title 'parallel sequencing,' since the problem is to arrange the n jobs in several sequences that start simultaneously. This is in distinction to the class of sequencing problems solved by JOHNSON[2] in which n jobs are to be put into a single sequence.

Let $N_i (i=1, 2, \cdots, n)$ be n jobs that have to be done with technological ordering restrictions, e.g., a hole in a part must be drilled before it is threaded, or, in a computation in which certain mathematical terms must be computed before other terms. Let each job require one unit of time. The whole ordering restriction can be represented by a graph G consisting of n nodes representing jobs and directed arcs representing ordering restrictions. In order that the jobs represented by the graph G can actually be carried out, obviously, there should be no cycles formed by directed arcs of G. Otherwise, the graph G is arbitrary in general cases. In Fig. 1 an arbitrary graph is shown.

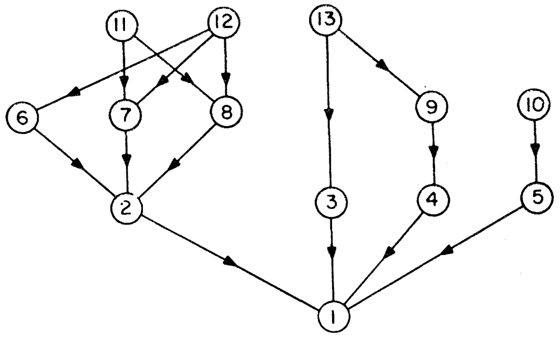

Figure 1

In Fig. 1, this would mean, for example, N_{10} must precede N_5; N_6, N_7, and N_8 must all precede N_2; N_{11} must precede both N_7 and N_8; and there are no ordering restrictions among N_8, N_3, and N_5.

We shall write: $N_i > N_j$ if N_i must precede N_j, $N_i \sim N_j$ if there is no ordering restriction between the two. Note that $N_i > N_j$ and $N_j > N_k$ imply $N_i > N_k$ while $N_i \sim N_j$ and $N_j \sim N_k$ do not imply $N_i \sim N_k$, i.e., N_i form a partially ordered set. We shall call a node N_k a final node in a graph G if there does not exist a node N_i in G such that $N_k > N_i$. In Fig. 1, for example, N_1 is a final node. Although a graph G may have more than one final node, we shall see, later, that the assumption of only one final node does not lose any generality. Hence, we shall assume that the graph G has only one final node in the following.

A node N_j is called a starting node in the graph if there does not exist a node N_i such that $N_i > N_j$. In Fig. 1, for example, N_{11}, N_{12}, N_{13}, and N_{10} are starting nodes. If N_{13} is finished, then N_3 and N_9 become starting nodes. In the process of assigning men to jobs, a node finished can be regarded as removed from the graph. A node N_j is called a current starting node if there is no node N_i in the current graph such that $N_i > N_j$.

A path in G is a sequence of directed arcs that represent ordering restrictions among the nodes in the path. The length of a path is the number of directed arcs in it. For example, the path from N_{10} to N_1 is of length 2 and the two paths from N_{13} to N_1 are of length 2 and length 3, respectively.

LABELLING PROCESS AND LOWER BOUNDS

IN WHAT follows, when we speak of labelling a node, we mean to assign a number to a node. All the nodes in the graph G are labelled in the following way.

A node N_i is labelled with $\alpha_i = x_i + 1$ if x_i is the length of the longest path from N_i to the final node in G. The final node by the above rule would then receive the label 1. In Fig. 2, all numbers that appear in the nodes are the α_i's of the nodes which appear in Fig. 1. The process of labelling is equivalent to finding the longest path from a node to the final node in G.

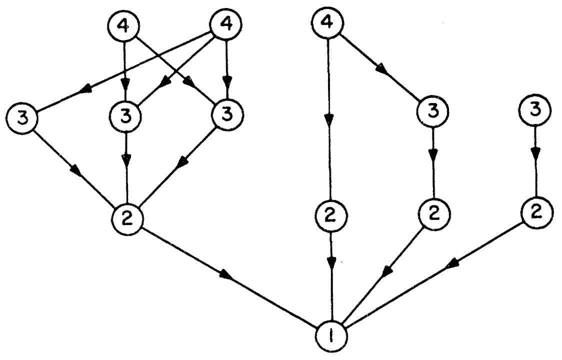

Figure 2

The labelling process can be done very quickly by starting with the final node and tracing backwards. Label all nodes 2 that connect with the final node by one arc, etc. If a node can receive more than one number, label it with the largest number it can receive.

Let $p(\alpha)$ be the number of nodes N_i with $\alpha_i = \alpha$. In Fig. 2, for example, we have $p(1) = 1$, $p(2) = 4$, $p(3) = 5$, and $p(4) = 3$. Let $s(\alpha)$ be the number of starting nodes with $\alpha_i = \alpha$. In Fig. 2, for example, $s(1) = 0$, $s(2) = 0$, $s(3) = 1$, and $s(4) = 3$. We shall write $\max_i \alpha_i = \boldsymbol{\alpha}$. In Fig. 2, $\boldsymbol{\alpha} = 4$.

Assume that we start from time $t = 0$. Nodes that are finished are considered to be removed from the graph. As time goes on, the current graph will change. When we deal with a certain current graph, we shall let $p_t(\alpha)$, $s_t(\alpha)$, and $\boldsymbol{\alpha}_t$ be the quantities defined above, in the current graph. The subscript t is used to indicate that t units of time have passed when the current graph is obtained. These notations $p_t(\alpha)$, $s_t(\alpha)$, $\boldsymbol{\alpha}_t$ are meaningful only when the current graph is known or a definite process of removing nodes is known.

Let T_0 be the shortest time to complete all jobs in the graph. Let T_t be the shortest time to complete all jobs in a current graph which is obtained after t units of time. Then for a current graph with α_t, regardless of how many men are available, it is clear that

$$T_t \geqq \alpha_t. \tag{1a}$$

In particular, we have
$$T_0 \geqq \alpha. \tag{1b}$$

Now, we shall study an inequality that is useful in getting lower bounds on the number of men required and also on the prescribed time to complete all jobs. Let $\alpha+c$ be the prescribed time where c is a nonnegative integer and let γ be a positive integer.

Let

$$\max_\gamma [1/(\gamma+c)\textstyle\sum_{j=1}^{j=\gamma} p(\alpha+1-j)] = 1/(\gamma^*+c)\sum_{j=1}^{j=\gamma^*} p(\alpha+1-j).$$

LEMMA 1. *If* (2) *is true,*

$$y < 1/(\gamma^*+c)\textstyle\sum_{j=1}^{j=\gamma^*} p(\alpha+1-j), \tag{2}$$

then it is impossible to complete all jobs with y men in $\alpha+c$ units of time.

Proof. The total number of nodes removed from the graph in t units of time with y men cannot exceed $y \cdot t$. Let $t=\gamma^*+c$. Then the total number of nodes removed must not exceed $y(\gamma^*+c)$ which by assumption (2) is less than $\sum_{j=1}^{j=\gamma^*} p(\alpha+1-j)$. Hence there exists at least one node with $\alpha_i \geqq \alpha+1-\gamma^*$ undone at $t=\gamma^*+c$. By (1a) we need at least

$$T_{\gamma^*+c} = \alpha+1-\gamma^*$$

units of time to finish the remaining graph. So we need a total of at least

$$\gamma^*+c+(\alpha+1-\gamma^*) = \alpha+c+1$$

units of time to finish the original graph.

In the next section, we shall use this result to give the minimum number of men that are required to complete all jobs within a prescribed time, in case the ordering restriction forms a tree. The result certainly is also of use in measuring the efficiency of an arbitrary proposed algorithm, although for an arbitrary graph, the lower bound on the number of men (if $\alpha+c$ is fixed first) or the lower bound on $\alpha+c$ (if y is fixed first) provided by (2) may be too low.

For the graph with two or more final nodes, we can introduce an artificial node that is preceded by all final nodes in the graph. If we label the artificial node with $\alpha_i=0$ and for other nodes N_i, let α_i be the length of the longest path from N_i to the artificial final node, all the above discussion holds true, so that we can assume only one final node in G.

OPTIMUM SEQUENCING

An assembly line is usually designed to produce a single product that consists of several elementary parts. A job N_i may either be making an elementary part or the job of putting several elementary parts together to make a more complicated part. The job of producing an elementary part must, of course, precede the job of putting things together for which the elementary part is needed. Since an elementary part can only be used in one complicated part, the ordering restrictions is then a tree.† We shall consider from now on, only the case that G is a tree.

In application, an assembly line consists of several subassembly lines in parallel. This model would either minimize the cycle time T for a given assembly line or the number of subassembly lines for a specified cycle time T. Or the final node of the tree may represent a certain mathematical expression where m is the number of parallel processors (i.e., arithmetic units) in a computer.

When the graph G is a tree, the length of the path from N_i to the final node is unique, and the labelling process is much simplified. Also, because of certain properties of a tree, we are able to answer the two questions in the introduction completely.

We shall first give the algorithm for finishing all jobs at the earliest time with a given number of men, illustrate it with an example, and then prove its validity. It is obvious that we can only assign men to current starting nodes in a graph.

Algorithm

Preliminary: Label all nodes with $\alpha_i = x_i + 1$ where x_i is the length of the path from N_i to the final node in the tree.

Algorithm: *If the total number of starting nodes is less than or equal to m, where m is the number of men available, then so are all starting nodes.*

If the total number of starting nodes is greater than m, choose m starting nodes with values of α_i not less than those not chosen. In the case of a tie, the choice is arbitrary.

The rule is then repeated for the remaining graph.

In Fig. 3, for example, the α_i of nodes N_i appear in the circles. If we have three men, then, according to the algorithm, we should choose three nodes among N_{16}, N_{17}, N_{18}, and N_{19}. Here we arbitrarily chose N_{17}, N_{18},

† Tree means a tree with one final node throughout the paper.

and N_{19} and enclose them in a dotted curve. The successive steps are self-explanatory. The whole graph is finished in eight units of time, which is clearly minimal.

The algorithm has the following mechanical anology. Use metal rings to represent nodes, and tie the rings together by pieces of string of unit length, to obtain a model of the tree diagram. Now, hold the final node

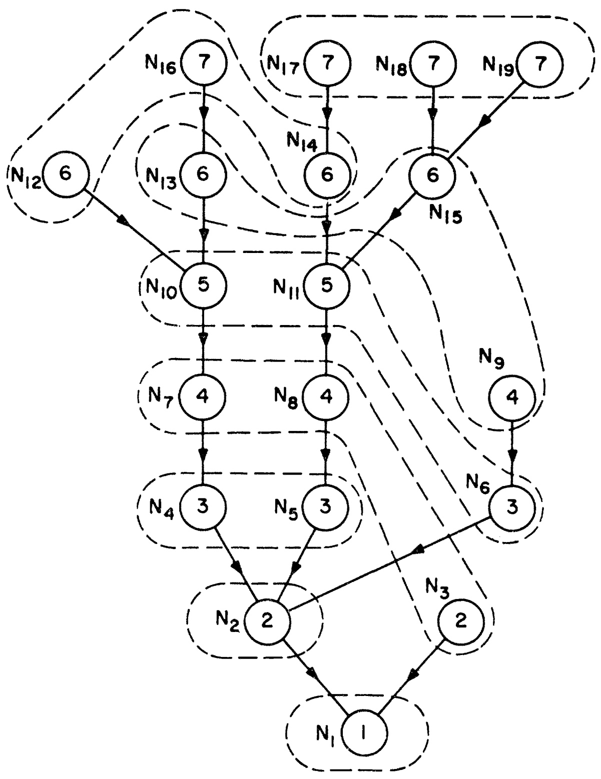

Figure 3

and let all the other rings go free. Then the algorithm is to cut off at most m end-rings at a time, with preference given to bottom end-rings if there are more than m available for cutting.

The algorithm can be described as 'cutting the longest queue.'

Although the algorithm is intuitively plausible, the proof that it completes all jobs at the earliest time is somewhat long.

First we consider the first question in the first section. If all jobs must be done within a prescribed time T, what is the minimum number of men required? We have seen from (1b) that the prescribed time must not be less than α. Let $T=\alpha+c$ where c is a nonnegative integer, and let m be the integer satisfying

$$m-1<1/(\gamma^*+c)\sum_{j=1}^{j=\gamma^*} p(\alpha+1-j)\leqq m. \tag{3}$$

We assert that m is the minimum number of men required. It is clearly necessary to have m men as we have proved in Lemma 1 that it is impossible to complete all jobs with $m-1$ men in $\alpha+c$ units of time. Now we shall prove that it is also sufficient.

Let $P(\gamma)$ be the set of nodes N_i with $\alpha_i\geqq\alpha+1-j(j=1,\cdots,\gamma)$. We shall define an integer γ' by the following conditions:

1. Using this algorithm, remove starting nodes m at a time, at time units $t=1,\cdots,c'$, until the number of starting nodes is less than m. Remove all the starting nodes in the current graph at $t=c'+1$. (c' is a nonnegative integer.)
2. All nodes in $P(\gamma)$ are removed after $c'+1$ units of time.
3. γ' is the largest of such integers.

In Fig. 3, for example, $\gamma'=5$. If the total number of starting nodes in the original graph is less than m, then we define $\gamma'=1$. In general, $1\leqq\gamma'\leqq\alpha$. (It should be noted that certain arbitrariness in the algorithm, i.e., removing any nodes with the same labelling in the case of a tie, will not change γ' or any other arguments discussed below.)

Let $S_t(\alpha)$ be the number of starting nodes (at time t) that have the labellings $\alpha_i\geqq\alpha$. Since the removal of a starting node either creates *one* or creates *no* starting node in a tree, $S_t(\alpha)$ is monotone decreasing as t increases for a fixed value of α.

LEMMA 2. *If it takes $c'+1$ units of time to remove nodes of $P(\gamma')$, then for any time $t\geqq c'+1$ all current graphs obtained by the algorithm will have $P_t(\alpha_t)<m$, i.e., all nodes with the largest labelling in a current graph will be less than m in number.*

Proof. Assume for a time $t'\geqq c'+1$, we have $p_{t'}(\alpha_{t'})\geqq m$; then all those nodes with the labelling $\alpha_{t'}$ must be starting nodes. So we have $S_{t'}(\alpha_{t'})>m$. From the above discussion we have $S_t(\alpha_{t'})\geqq m$ for all $t\leqq t'$. Therefore, there have always been enough starting nodes to make it possible to remove nodes 'm' at a time. As all of these removed nodes have labellings of not less than $\alpha_{t'}=\alpha+1-\gamma$, this contradicts the fact that γ' is the largest of such integers.

Note that there are at most $(c'+\lambda)m$ nodes in $P(\gamma')$ where $0<\lambda<1$.

THEOREM. *If the number of men m satisfies the condition* (3), *and if the graph of ordering restrictions is a tree, then all jobs can be completed within $\alpha+c$ units of time.*

Proof. By the definition of γ^*, if (3) is true, then

$$1/(\gamma'+c)\sum_{j=1}^{j=\gamma'} p(\alpha+1-j)\leqq m \tag{4}$$

or

$$\sum_{j=1}^{j=\gamma'} p(\alpha+1-j)\leqq(\gamma'+c)m. \tag{5}$$

Since $$\textstyle\sum_{j=1}^{j=\gamma'} p(\alpha+1-j) \geqq (c'+\lambda)m,$$
then from (5), we have
$$c'+\lambda \leqq c+\gamma'$$
or
$$c' \leqq c+\gamma'-1+(1-\lambda). \tag{6}$$

As c' and $(c+\gamma'-1)$ are both integers and $0<1-\lambda<1$, we have
$$c' \leqq c+\gamma'-1. \tag{7}$$

As we can remove all nodes of $P(\gamma')$ in $c'+1$ units of time, it follows from (7) that we can remove all nodes of $P(\gamma')$ in $(c+\gamma'-1)+1=c+\gamma'$ units of time. By Lemma 2, we can remove the remaining nodes in $\alpha-\gamma'$ units of time. So we need totally *at most* $c+\gamma'+(\alpha-\gamma')=\alpha+c$ units of time. Therefore, it is sufficient to have m men.

Now we consider the second problem: if m is the number of men available, what is an algorithm to complete all jobs at the earliest time? We assert that the algorithm (cutting the longest-queue) completes all jobs at the earliest time. This can be seen as follows. Let m be given first, then either
$$\max_\gamma[(1/\gamma)\textstyle\sum_{j=1}^{j=\gamma} p(\alpha+1-j)] \leqq m \tag{8}$$
or we can choose a smallest positive integer $c \geqq 1$ for which
$$\begin{aligned}\max_\gamma[1/(\gamma+c)\textstyle\sum_{j=1}^{j=\gamma} p(\alpha+1-j)] &\leqq m \\ &< \max_\gamma[1/(\gamma+c-1)\textstyle\sum_{j=1}^{j=\gamma} p(\alpha+1-j)].\end{aligned} \tag{9}$$

If (8) is true, then by the Theorem, we can finish all jobs within α units of time and from (1b) we see that α is minimal. If (9) is true, then from the Theorem, we can finish all jobs within $\alpha+c$ units of time, and from Lemma 1, we see that it is impossible to finish all jobs with m men in $\alpha+c-1$ units of time; hence the algorithm finishes all jobs at the earliest time.

If a job requires three periods of time to perform, then we can consider the job as three nodes each requiring one unit of time and connected by two directed arcs in series. The result of Lemma 1 still holds, but the results in this section are no longer true.

THE AUTHOR is indebted to DR. N. C. HSU and DR. T. J. RIVLIN for reading the first draft of the paper and making valuable suggestions.

REFERENCES

1. L. R. FORD, JR. AND D. R. FULKERSON, *Flows in Networks*, Chap. 2, Sec. 9 forthcoming, Princeton Univ. Press, 1961.
2. S. M. JOHNSON, "Optimal Two- and Three-Stage Production Schedules with Setup Times Included," *Naval Res. Log. Quart.* **1,** 61–68 (1954).