AEC Computing and Applied Mathematics Center
Institute of Mathematical Sciences
New York University

TID-4500
15th Ed.

NYO-9488
PHYSICS

AN ALGORITHM FOR CONSTRUCTING FEASIBLE SCHEDULES AND COMPUTING THEIR SCHEDULE TIMES.

by

Jack Heller and George Logemann

November 15, 1960

462 002

ABSTRACT

An algorithm for the generation of feasible schedules and the computation of the completion times of the job operations of a feasible schedule is presented. Using this algorithm, the distribution of schedule times over the set of feasible schedules - or a subset of feasible solutions - was determined for technological orderings that could occur in a general machine shop. These distributions are found to be approximately normal. Biasing techniques corresponding to "first come first serve," random choice of jobs ready at each machine and combinations of these two extremes were used to compute distributions of schedule times.

In all the experiments "first come first serve" appears the best in the sense that convergence to the minimum is fastest and the sample minimum is the smallest.

TABLE OF CONTENTS

AN ALGORITHM FOR CONSTRUCTING FEASIBLE SCHEDULES AND COMPUTING THEIR SCHEDULE TIMES.

1. Introduction.

Lacking a practical algorithm to solve sequencing problems[1], for example the problem of finding a minimum schedule for the processing of job operations through a given set of machines, one must rely heavily upon simulation and sampling techniques.

A scheduling problem can be stated as a problem in linear graph theory[7], in terms of which the precedence relations of the schedule are depicted in a linear graph. Thus our algorithm is based on properties of linear graphs, as opposed to the properties of Gantt diagrams which motivate Giffler and Thompson[4]. More specifically, our algorithm embodies a recursive method of condensing the graphs corresponding to technological orderings into one feasible schedule-graph. Further, in the linear graph model of a schedule, the completion time becomes a recursive function defined on the graph. Thus our algorithm gives a method to compute quite easily the completion time as well as generate a schedule. In addition to the intrinsic recursivity, which makes the scheme easily programmable for automatic computers, our algorithm can construct schedules restricted by a large class of side conditions, corresponding to ground rules, e.g. urgency requirements, due date, minimum inventory, ...

In particular, numerical simulation experiments were performed

462 005

using our method, in which feasible schedules were constructed for technological orderings requiring many returns of a particular job to a particular machine. (Schedules of this type arise classically in general[1] machine shops and also in the recent discussions of multiprogramming[2].) In these experiments, feasible schedules were generated at random, with the possibility of a variable first-come-first-served bias. In all cases tried, pure first-come-first-serve gave not only the smallest observed schedule time, but took the least number of samples to converge to this minimum. The program for an IBM 704 can generate samples at the processing rate of about a thousand job operation nodes per second for the most complex technological orderings [c.f. Fig.10]. This time estimate is about half that tentatively reported by Giffler and Thompson[4]. It is feasible to search for more optimal side conditions or to solve practical problems using sampling techniques in conjunction with the following algorithm.

2. Precedence Relations, Directed Linear Graphs, and Schedule Time.

Convenient pictorial representations of the precedence relations of technological and schedule orderings are given by linear graphs[7]. We will denote each node of these linear graphs by a triple of integers (m j i): Here i represents the $i^{\underline{th}}$ return of job j to machine station m[1]. The directed branches of these linear graphs will indicate the precedence relations of the jobs through the machines.

Thus, for example, in Figure 1 at machine station 2 for the first time, (211), can only be started after job 1 at machine

[1] In the present discussion we will limit each machine station to contain one machine.

station 1 for the first time, (111), and job 1 at machine station 3 for the first time, (311), are completed. We say that (211) is possibly held up by (111) and (311); or, in the terminology of graph theory, (111) is covered by (211) and (311) is covered by (211); or (211) covers (111) and (311); or by introducing a natural notation: (111)$\rightarrow$(211) and (311)$\rightarrow$(211).

Using the above concept, it is the coverings that determine the starting time of a particular job on a particular machine for the i^{th} time. If we call T(m j i) the completion time of job j at machine station m for the i^{th} time, i.e. node (m j i), and if we call t_{mji} the processing time of node (m j i), the completion time of node (m j i) is

$$(2.1) \qquad T(m\ j\ i) = \max_{(m'j'i')\rightarrow(m\ j\ i)} T(m'j'i') + t_{mji},$$

because node (m j i) can be processed only after all nodes covering node (m j i) are completed and each node is started as soon as possible subject only to ordering constraints.

For a given technological ordering we can compute recursively the completion time of each node (m j i). In technological ordering $\mathcal{J}_1$, Figure 1, we observe that nodes (111) and (311) do not cover any nodes. Hence we give them a starting time of zero. Since the first term on the right of (2.1), that using the max operator, is the starting time of node (m j i), the completion time of nodes (111) and (311) is

$$T(111) = t_{111}$$

$$T(311) = t_{311} .$$

The node (211) covers (111) and (311) whose completion times are known. We thus have from (2.1)

$$T(211) = \max \begin{Bmatrix} T(111) \\ T(311) \end{Bmatrix} + t_{211}.$$

And finally

$$T(112) = T(211) + t_{112}.$$

When we have a schedule of a set of jobs, we can compute the completion times in the same manner. However, as is well known, every schedule will not be feasible[1,7]; i.e. some orderings of the jobs through each machine station may violate the given technological orderings. A schedule that is feasible is one whose schedule graph S has order relations consistent with the given technological order relations. In Figure 2 we exhibit one such graph, S_o, and note that the order relations of the "circle" nodes (job $\mathcal{J}_1$) and the "square" nodes (job $\mathcal{J}_2$) are consistent with those given in Figure 1.

The schedule time for a given schedule graph[2] S is

(2.2) $$T(S) = \max_{(m\ j\ i)} T(m\ j\ i).$$

---

[2] For the remainder of the paper all schedule graphs are assumed to be feasible unless explicitly stated otherwise.

The coverings in the graph S are used to evaluate each T(m j i).

3. <u>The Algorithm</u>.

By the term algorithm we mean a formal set of logical and numerical rules for the computation of some desired numerical function, viz. the completion times of a feasible schedule, the schedule time of a feasible schedule, ... The meaning of algorithm implies that the pictorial representations of both linear graphs and Gantt diagrams are not directly suitable for use in computer programs. Therefore, we introduce the notion of a "table of coverings," a list which contains the essential information about the precedence relations of the linear graph.

To describe the algorithm, let us first consider an example, namely, the construction of schedule S_0 (Fig. 2) from the technological orderings $\mathcal{J}_1$ and $\mathcal{J}_2$ (Fig. 1). Following this example illustrating the features of the algorithm, we will give the explicit rules.

To begin, we set up a table of coverings, such as Figure 3. The columns have the following meaning:

column 1: node designation (m j i).
column 2: nodes covering given node in technological ordering $\mathcal{J}_j$.
column 3: number of nodes covered by node in technological ordering $\mathcal{J}_j$.
column 4: processing time t(m j i).
column 5: index of schedule order for node on machine m.
column 6: working storage location for generating starting time of node.
column 7: finish time T(m j i).

462 009

Each line of the table corresponds to one node in the technological orderings. For convenience, the nodes are divided into three groups corresponding to machines 1, 2, and 3. The processing times $t(m\ j\ i)$ are purely arbitrary. Note that, by using the table of coverings, we can reconstruct the original graphs. For, consider the first line, corresponding to node (111). In column 2 it is stated that (211) covers (111), hence there is an arrow <u>from</u> (111) to (211) in the linear graph. That there is no arrow <u>to</u> (111) is noted in column 3. Continuing with each node we could reconstruct the linear graphs of Figure 1 from the table of Figure 3. Thus the table of coverings is in some sense equivalent to the linear graph.

It becomes our purpose thereby to construct a table corresponding to a feasible schedule. Certainly the covering table of a schedule must contain all the entries in the table of coverings. Moreover, when we construct a schedule such as S_0 from orderings $\mathcal{J}_1$ and $\mathcal{J}_2$, we are simply inserting other node triples into column 2. But a triple $(m\ j\ i)$ thus inserted into a row will have the same m as in column 1 of that row; since one object of the algorithm is to find the scheduled ordering of nodes on machine m, for each node we will generate, in column 5, integers 1, 2, 3... indicating that this is the first or second or third, etc., node on machine m[3].

In addition, we must compute, for the <u>augmented</u> table, the completion times $T(m\ j\ i)$. To do so we shall first generate the starting time for node $(m\ j\ i)$ in column 6 and then add to this quantity the processing time $t(m\ j\ i)$. Recalling that the starting

---

3 This is actually a choice of notation and is not actually necessary.

462 010

time of (m j i) is given by the maximum of T(m'j'i') on the set of (m'j'i') covered by (m j i), we shall generate the starting time for many nodes simultaneously: that is, as soon as T(m'j'i') has been determined[4], we shall examine Q_6(m"j'i") for all nodes (m"j'i") entered in Q_2(m'j'i'). We know node (m"j'i") cannot be processed until (m'j'i') is finished, thus Q_6(m"j'i") must be <u>at least</u> equal to T(m'j'i"). Therefore if Q_6(m'j'i') is less than T(m"j'i"), we must replace Q_6(m"j'i") with T(m'j'i').

However, this procedure only accounts for technological ordering delay before processing (m j i). In addition, we must also take into consideration a possible delay from the schedule; that is, in the case that the machine m is occupied. This procedure is accomplished by comparing T(m $j_0 i_0$) to Q_6(m j i) where (m $j_0 i_0$) is currently the last node scheduled on machine m.

The process to be described will therefore include rules for simultaneously scheduling nodes and computing their starting and finishing times. We will make use of this principle: A node (m j i) may be scheduled if <u>and only if</u> all nodes (m'j i') have been scheduled, where (m j i) covers (m'j i'), (m'j i')$\rightarrow$(m j i). Note also that the number of such nodes is given in Q_3(m j i).

To start the scheduling of the example, we observe that certain nodes (those with $Q_3 = 0$) cover no other nodes and therefore may

---

[4] In what follows we will refer many times to "the quantity in column n for node (m j i)." This will be called by the symbol Q_n(m j i). Also, t(m j i) = Q_4(m j i), T(m j i) = Q_7(m j i).

462 011

be scheduled. From these nodes: (111), (221), (311), and (321) select[5] node (111). Since (111) may be scheduled on machine 1, we set $Q_5(111) = 1$. Now that (111) is scheduled, we flag $Q_3(111)$ with some convenient negative number, say -1. Note that as the starting time of (111) is 0, its finish time is $T_{(111)} = t_{(111)} = 1.0$; this number is stored in $Q_6(111)$. Since we now know the node covered by (111) in the technological ordering, namely (211), cannot start before time 1.0, then we may compute the maximum of $T(111)$ and $Q_6(211)$, which is 1.0, to be stored in $Q_6(211)$. Also we subtract 1 from $Q_3(211)$ obtaining a remainder of 1, which states that there is yet one node to be scheduled before we can schedule (211). The updated table is now given in Figure 4, where we also show the current linear graph and Gantt diagram.

Now we have the possibility of scheduling one of the nodes (221), (311), and (321). Suppose we select (311); machine 3 has not been used, so set $Q_5(311) = 1$; set $Q_3(311) = -1$, for (311) is now scheduled. Then, since (311) covers no nodes in the schedule, we find $T(311) = t(311) = 2.0$. Again (211) is the only node covering (311); thus we replace $Q_6(211)$ by $\max(Q_6(211), T(311)) = \max(1.0, 2.0) = 2.0$. By subtracting 1 from $Q_3(211)$, we show that we have entered in the starting time expression another possible contribution. The remainder of zero indicates that (211) may now be scheduled. See Figure 5.

---

5 One may use a rule involving any of the quantities T_{mji}, t_{mji}, etc. The rule in this example is, "Select nodes at random."

462' 012

Thus we may now schedule one of the nodes (211), (221), and (321). Let us select (221). After scheduling this node, we obtain the table in Figure 6. From $Q_3(221)$, $Q_5(221)$, and $Q_7(221)$ we know that (221) has now been scheduled first on machine 2 with finishing time $T(211) = 3.0$. This node, however, is covered by two nodes: thus both $Q_3(121)$ and $Q_3(322)$ have been reduced by 1; both $Q_6(121)$ and $Q_6(322)$ have been set to the maximum of their previous values and $T(211)$.

If we now schedule node (211), we meet a new problem not yet discussed: machine 2 is now in use, as the lowest integer in $Q_5(2ji)$ is 1. Hence (211) will be the second node scheduled, and one must set $Q_6(211) = \max(Q_6(211), T(221))$, for (221) is the node scheduled first on machine 2. Then, as before, $T(221) = t(221) + Q_6(221) = 4.0$; finally we adjust the entries for (112), finding $Q_5(112) = 0$ and $Q_6(112) = 4.0$. C.f. Figure 7.

If we now continue, processing the remaining nodes in the order (121), (321), (112), (322), and (323), we will obtain Figure 8, the finished form of the table corresponding to S_0. The schedule time of S_0 is 7.0, the maximum of the entries $T(mji)$. By comparing the final Gantt diagram with the linear graphs in Figures 1 and 2, we see that the schedule is feasible and preserves the initial technological orderings; moreover, we have the correct starting and finishing times for all the nodes.

Summarizing, the steps of the algorithm are:

1.) Initialize the table of coverings, in particular by setting, for all (m j i), $Q_3(m\ j\ i)$ to the number of nodes covered by (m j i) in the technological

462 013

orderings and by setting $Q_5(m\ j\ i) = Q_6(m\ j\ i) = 0$.

ii.) Select one of the nodes $(m_o j_o i_o)$ with $Q_3(m_o j_o i_o) = 0$.
Let $k = \max\limits_{(j,i)} Q_5(m_o j\ i)$.

iii.) a.) If $k = 0$, set $Q_5(m_o j_o i_o) = 1$.

b.) If $k \neq 0$, set $Q_5(m_o u_o i_o) = k + 1$ and replace $Q_6(m_o j_o i_o)$ by $\max(Q_6(m_o j_o i_o), T(m_o j\ i))$ where $Q_5(m_o j\ i) = k$.

iv.) Set $Q_5(m_o j_o i_o) = -1$.

v.) Compute $T(m_o j_o i_o) = Q_6(m_o j_o i_o) + t(m_o j_o i_o)$

vi.) For each node $(m\ j_o i)$ in $Q_2(m_o j_o i_o)$, replace $Q_6(m\ j_o i)$ by $\max(Q_6(m\ j_o i), T(m_o j_o i_o))$ and subtract 1 from $Q_3(m\ j_o i)$. If there are no such nodes, ignore this step.

vii.) Repeat ii.)...vi.) until all entries in column 3 are -1. The maximum of the numbers in column 7 is the total schedule time.

Let us conclude this section with a few general comments. We observe that the processing times may depend, in some applications, upon the starting times and in general upon any function of all the nodes previously scheduled. Our algorithm can handle these cases with no modifications.

On the other extreme, we might point out that to generate a feasible graph one never has to consider the processing times at all, for feasibility is a combinatorial <u>property</u> of graphs and the times are a <u>function</u> defined on the graph. Simply employ the algorithm without using columns 4,6, and 7. Still from another point of view, if one wishes to evaluate some recursive function on an already existing linear graph he may use a very similar technique to that described in the algorithm.

Finally, in many cases the rules for selecting nodes to be processed are logical functions of the processing times. In fact, it seems that one very promising field for further research is the discovery of rules which only allow a subset of all possible schedules to be generated, but in such a way that the chance of obtaining a minimal schedule is increased. Giffler and Thompson[4] have discovered one such rule in their concept of "left-shifting." In the following section we discuss another possibility.

4. Numerical Experiments.

Using the algorithm described in section 3, we have randomly sampled feasible schedules for various sets of technological orderings. The purpose of our experiments was to see if a possible bias procedure could be determined so that the schedules randomly generated would have for the most part schedule times near the minimum schedule time.

Formally, in any scheduling problem, we are attempting to find the minimum schedule time over the set of feasible schedules. If we call $\mathcal{F}$ the set of feasible schedules, the minimum schedule time is

$$T_{min} = \min_{S \in \mathcal{F}} \max_{(mji) \in S} T(m\ j\ i) \tag{4.1}$$

where $T(m\ j\ i)$ is given by (2.1). If we choose a subset of

162 015

feasible schedules $\mathcal{S}$, i. e. if $S \subset \mathcal{F}$,

(4.2) $$T_{min} \leq \min_{S \in \mathcal{S}} \max_{(m j i) \in S} T(m\ j\ i) \leq T_{max},$$

where T_{max} is the maximum schedule time over all feasible schedules $\mathcal{F}$. A biased procedure chooses some subset $\mathcal{S}$ of feasible schedules. It is desired that this biased subset $\mathcal{S}$ contains T_{min} with a relatively high probability.

Suppose we define a bias in the following manner: when we are about to perform step ii) of the algorithm, we must select one node from P, the set of nodes which now may be scheduled. Let us consider two possible ways of making this choice:

a) Choose a node at random. (I.e., if P has n nodes, select a node in such a way that each node has probability $\frac{1}{n}$ of being chosen.)

b) Let Q be a subset of P on which the starting time is a minimum. Then randomly choose a node from Q.

Observe that rule (b.) corresponds to a "first come, first served" type of rule.

Our biasing corresponds to a third case containing both of the above:

c) Use methods (a.) and (b.) with probabilities 1-p and p respectively. For convenience we shall call p the "bias factor;" p = 0 corresponds to purely random selection; p = 1 implies a continual application of "first come, first serve."

It was hoped that by a judicious choice of p we could sample

462 016

from a subset of schedules which would contain relatively many minimum schedules. It is well known that a subset of schedules may not contain the minimum schedule: in the Johnson[3] 2-job 4-machine example the "first come, first serve" subset of schedules does not contain the minimum. However, this example is extreme in the sense that there are relatively few schedules not of the "first come, first serve" type that are smaller than the minimum "first come, first serve" schedule. In the assembly line case[5] the total number of schedules giving the minimum "first come, first serve" schedule value relative to the total number of feasible schedules is very much smaller than the number of minimum "first come, first serve" schedules relative to the total number of "first come, first serve" schedules.

In our experiments we have used biasings to determine the probability distributions of the schedule times over the set of schedules. The technological orderings of one set of numerical experiments are given in Figure 9; the corresponding processing times of each operation are given above each node. In Figure 10 we give the distributions of schedule times determined by random sampling. Since the processing times of each operation are integers, the schedule times are integers; however, we have drawn the normal distributions determined by the sample expected schedule time and sample variance of the schedule time as continuous functions.

We see that the biasing of 1, "first come, first serve," gives an expected schedule time smaller than the other expected schedule times; we see that the biasing of 1 gives a variance of the schedule time smaller than the other variances. In this

462 017

example it is better to sample from the bias of 1 set of schedules with bias p = 1 for the probability of finding a minimum seems greatest. In all our experiments, we have noticed that the "first come, first serve" set gives the minimum. For the assembly line case an explanation in probability has been given[5].

We have also noticed that as the technological orderings become more complex, the "first come, first serve" case converges to the minimum sample in less trials. (An example of a typical, complex technological ordering is given in Figure 11.) It would seem that reversals of operations rarely (if at all) produce schedules whose times are smaller than the "first come, first serve" minimum.

APPENDIX

Perhaps we may indicate a few justifications for the above process, to show all possible feasible schedules may be generated [depending on the selection rule in step (ii) [6]] and that the times calculated are correct. The terminology is that of linear graph theory. In particular, these proofs hold for feasible (finite, transitive) graphs, and also for partially-ordered systems, since the latter can be reduced to the former by suitably redefining the non-coverings (cf.ref. [7]). Proofs of the following remarks are merely indicated; formal rigor may be obtained by those who wish to consult [7].

<u>Remark 1</u>: Given any feasible graph S_0, then S_0 may be generated with suitable rule ii). <u>Proof</u>: As S_0 is feasible, either S_0 is empty or there must be an initial node $(m_1 j_1 i_1)$ in S_0 (one which covers no other node), otherwise S_0 has a loop. Let S_1 be the graph derived from S_0 by removing node $(m_1 j_1 i_1)$. S_1 is feasible, otherwise S_0 would not be, either. Again, either S_1 is void or contains an initial node $(m_2 j_2 i_2)$. Thus we obtain a sequence of graphs $S_1, \ldots, S_k$ where S_k is void and such that S_{v+1} is obtained from S_v by removing node $(m_v j_v i_v)$. Suppose rule ii) is a random rule; i.e., at any stage of the computation, any node $(m\ j\ i)$ with $Q_3(m\ j\ i) = 0$ has a non-zero probability of being chosen. If the nodes are selected in order $(m_1 j_1 i_1), \ldots, (m_k j_k i_k)$, then S_0 will be generated.

<u>Remark 2</u>: If $J_1, \ldots, J_n$ are a finite number of feasible graphs, then the result of the process is a feasible graph S_0.

---

[6] C.f. footnote 5, p. 11.

462 019

Proof. It is impossible to generate a loop with the algorithm, for consider the stage of the process in which we select a node (m j i). This node is not covered in any feasible technological ordering J_i by some node $(m_0 j_0 i_0)$ which has been previously scheduled, for $(m_0 j_0 i_0)$ could only have been scheduled after all of the nodes which it does cover in J_i had also been previously scheduled. Also, it is impossible that node $(m_0 j_0 i_0)$ precede and follow (m j i) on the same machine. Thus S_0 must be feasible.

Remark 3: Each node in each graph J_i must occur in S_0.

Proof: For, each J_v is feasible and therefore has at least one initial element $(m_v j_v i_v)$. Further, if we remove the nodes from J_v as they are scheduled, when $(m_v j_v i_v)$ has been scheduled, there will be at least one more initial node $(m_{v'} j_{v'} i_{v'})$ or else J_v is void. Only when all the J_v are thus void will the process end. Thus all nodes in all technological orderings J_v will have to be scheduled in S_0.

Remark 4: Thus the algorithm can generate all of the feasible schedules and only these schedules from a set of feasible technological orderings. **Proof:** For, all the schedules can be generated by remark 1, and by remarks 2 and 3, only feasible graphs that contain all nodes can be generated.

Remark 5: The processing times generated are those given by formula (2.1). **Proof:** All contributions to the starting term, are entered into the maximum term: for any (m'j'i') of the nodes over which the maximum of T(m'j'i') is taken is either covered by (m j i) in some technological ordering, or occupies machine m directly before (m j i) does.

462 020

Bibliography

[1] Churchman, C. W., Ackoff, R. L. and Arnoff, E. L., Introduction to Operations Research, John Wiley and Sons, Inc. (1957).

[2] Codd, E. F., "Multiprogram Scheduling" Comm. ACM v. 3, pp. 347-350 and 413-418 (1960).

[3] Johnson, S. M., "Optimal two- and three-stage production schedules with set up time included," Naval Research Logistics Quarterly, v. 1, (1954).

[4] Giffler, B. and Thompson, G., "Algorithms for solving production-scheduling problems," J. Opers. Res. Am., v. 8, no. 4, pp. 487-503 (1960).

[5] Heller, J., "Combinatorial, probabilistic and statistical aspects of an M x J scheduling problem," A.E.C. Research and Development Report NYO-2540 (1959).

[6] Heller, J., "Some numerical experiments for an M x J flow shop and its decision theoretic aspects," J. Opers. Res. Am., v. 8, no. 2, pp. 178-184 (1960).

[7] Heller, J., "Some problems in linear graph theory that arise in the analysis of the sequencing of jobs through machines." A.E.C. Research and Development Report NYO-9487 (1960).

462 021

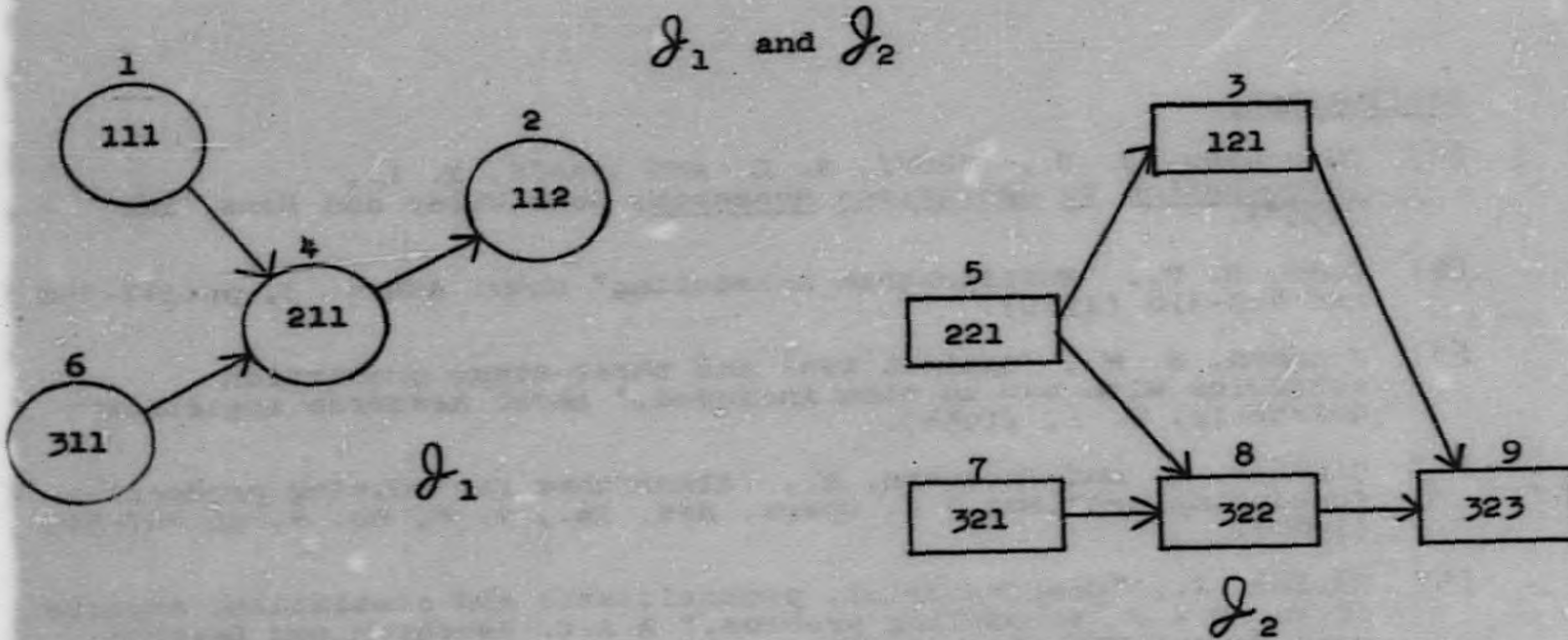


Figure 1.

Linear graphs picturing technological orderings.

S_0

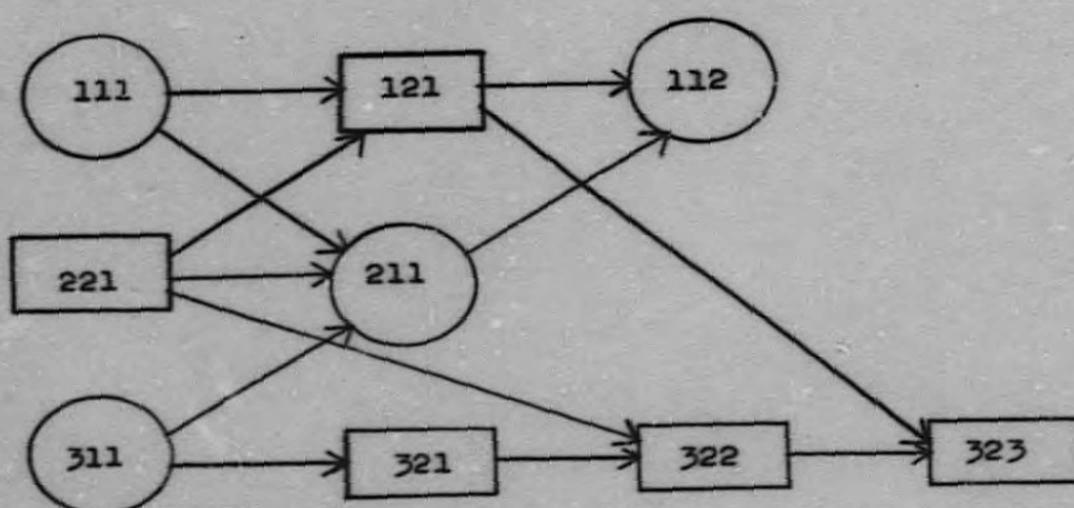


Figure 2.

A feasible schedule S_0 constructed from the technological orderings of Figure 1.

462 022

Initialized Table of Coverings

Q_1	Q_2	Q_3	t	Q_5	Q_6	T
111	211	0	1.0			
112		1	1.0			
121	323	1	3.0			
211	112	2	1.0			
221	121,322	0	3.0			
311	211	0	2.0			
321	322	0	1.0			
322	323	2	1.0			
323		2	1.0			

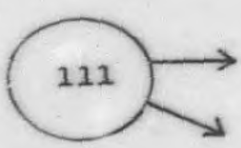
Figure 3

462 023

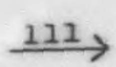
After Scheduling Node (111)

Q_1	Q_2	Q_3	t	Q_5	Q_6	T
111	211	-1	1.0	1		1.0
112		1	1.0			
121	323	1	3.0			
211	112	1	1.0		1.0	
221	121,322	0	3.0			
311	211	0	2.0			
321	322	0	1.0			
322	323	2	1.0			
323		2	1.0			

a. table



b. linear graph



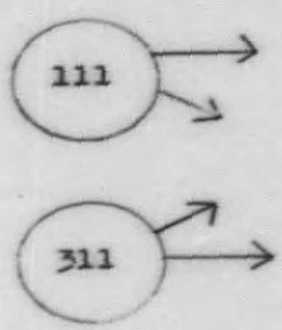
c. Gantt diagram

Figure 4

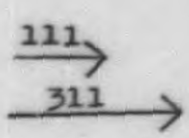
After Scheduling Node (311)

Q_1	Q_2	Q_3	t	Q_5	Q_6	T
111	211	-1	1.0	1		1.0
112		1	1.0			
121	323	1	3.0			
211	112	0	1.0		2.0	
221	121,322	0	3.0			
311	211	-1	2.0	1		2.0
321	322	0	1.0			
322	323	2	1.0			
323		2	1.0			

a. table



b. linear graph



c. Gantt diagram

Figure 5

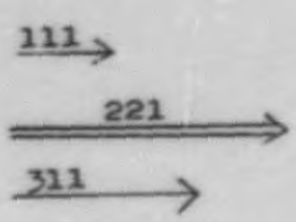
462 025

After Scheduling Node (221)

Q_1	Q_2	Q_3	t	Q_5	Q_6	T
111	211	-1	1.0	1		1.0
112		1	1.0			
121	323	0	3.0		3.0	
211	112	0	1.0		2.0	
221	121,322	-1	3.0	1		3.0
311	211	-1	2.0	1		2.0
321	322	0	1.0			
322	323	1	1.0		3.0	
323		2	1.0			

a. table

b. linear graph



c. Gantt diagram

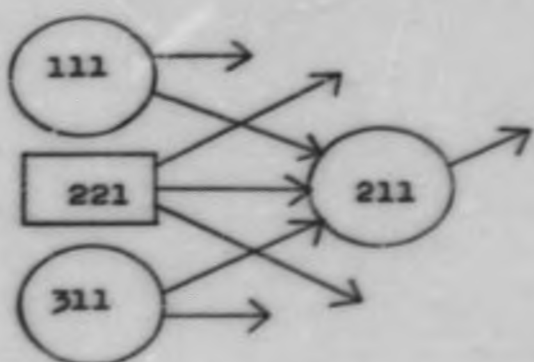
Figure 6

462 026

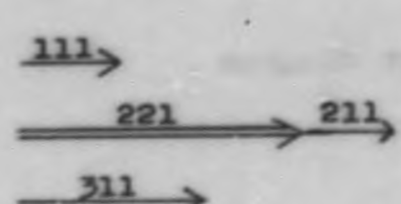
After Scheduling (211)

Q_1	Q_2	Q_3	t	Q_5	Q_6	T
111	211	-1	1.0	1		1.0
112		0	1.0		4.0	
121	323	0	3.0		3.0	
211	112	-1	1.0	2	3.0	4.0
221	121,322	-1	3.0	1		3.0
311	211	-1	2.0	1		2.0
321	322	0	1.0			
322	323	1	1.0		3.0	
323		2	1.0			

a. table



b. linear graph



c. Gantt diagram

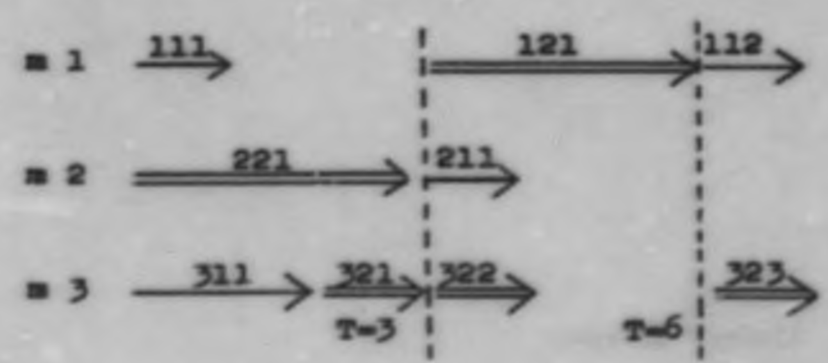
Figure 7

462 027

Final Table of Coverings and Gantt Diagram
for S_0, the Schedule in Figure 2.

Q_1	Q_2	Q_3	t	Q_5	Q_6	T
111	211	-1	1.0	1		1.0
112		-1	1.0	3	6.0	7.0
121	323	-1	3.0	2	3.0	6.0
211	112	-1	1.0	2	3.0	4.0
221	121,322	-1	3.0	1		3.0
311	211	-1	2.0	1		2.0
321	322	-1	1.0	2	2.0	3.0
322	323	-1	1.0	3	3.0	4.0
323		-1	1.0	4	6.0	7.0

a. table



b. Gantt diagram

Figure 8

462 028

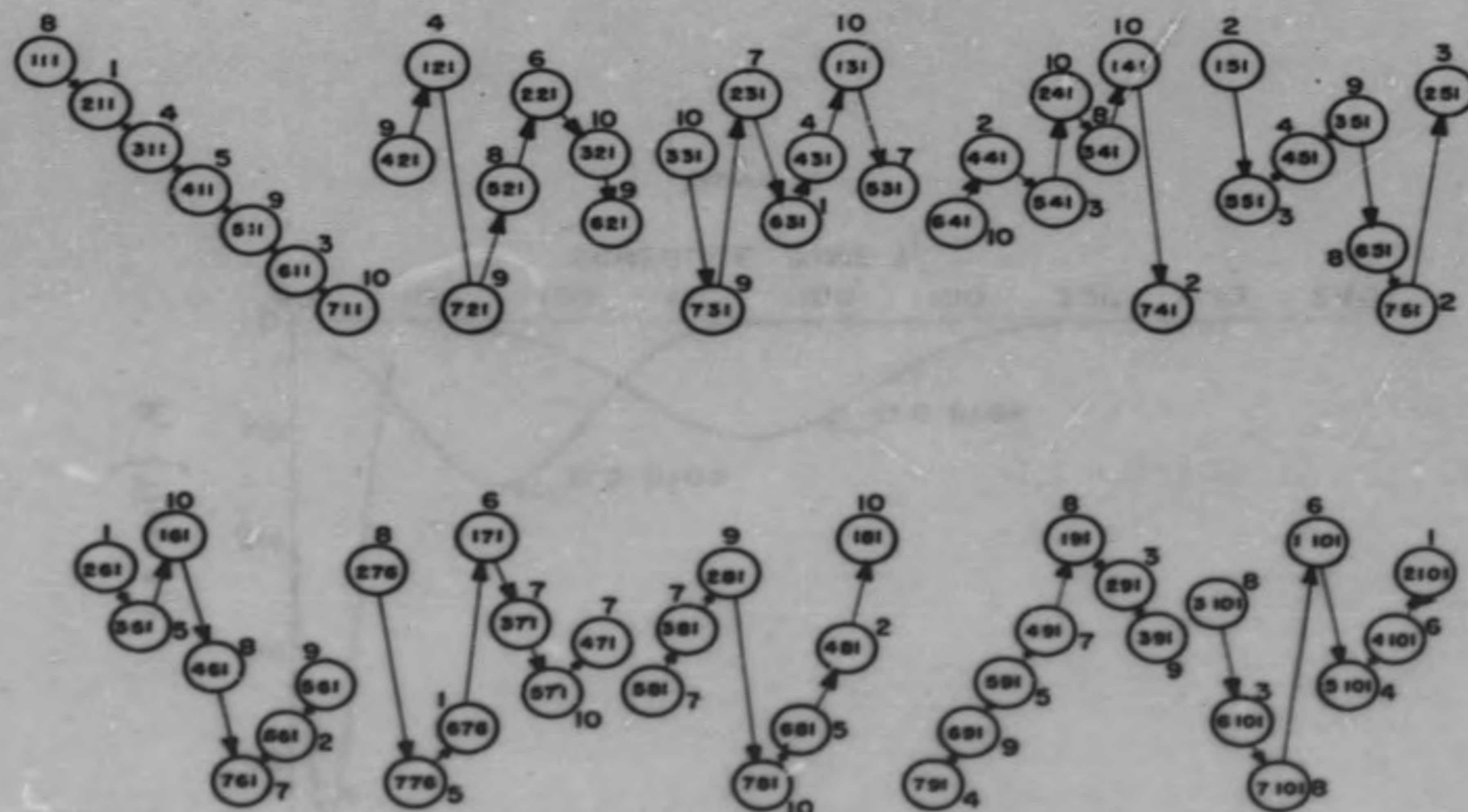


Figure 9

TECHNOLOGICAL ORDERINGS WITH THEIR OPERATION TIMES t_{mji} INDICATED NEXT TO EACH NODE.

Figure 9

462 030

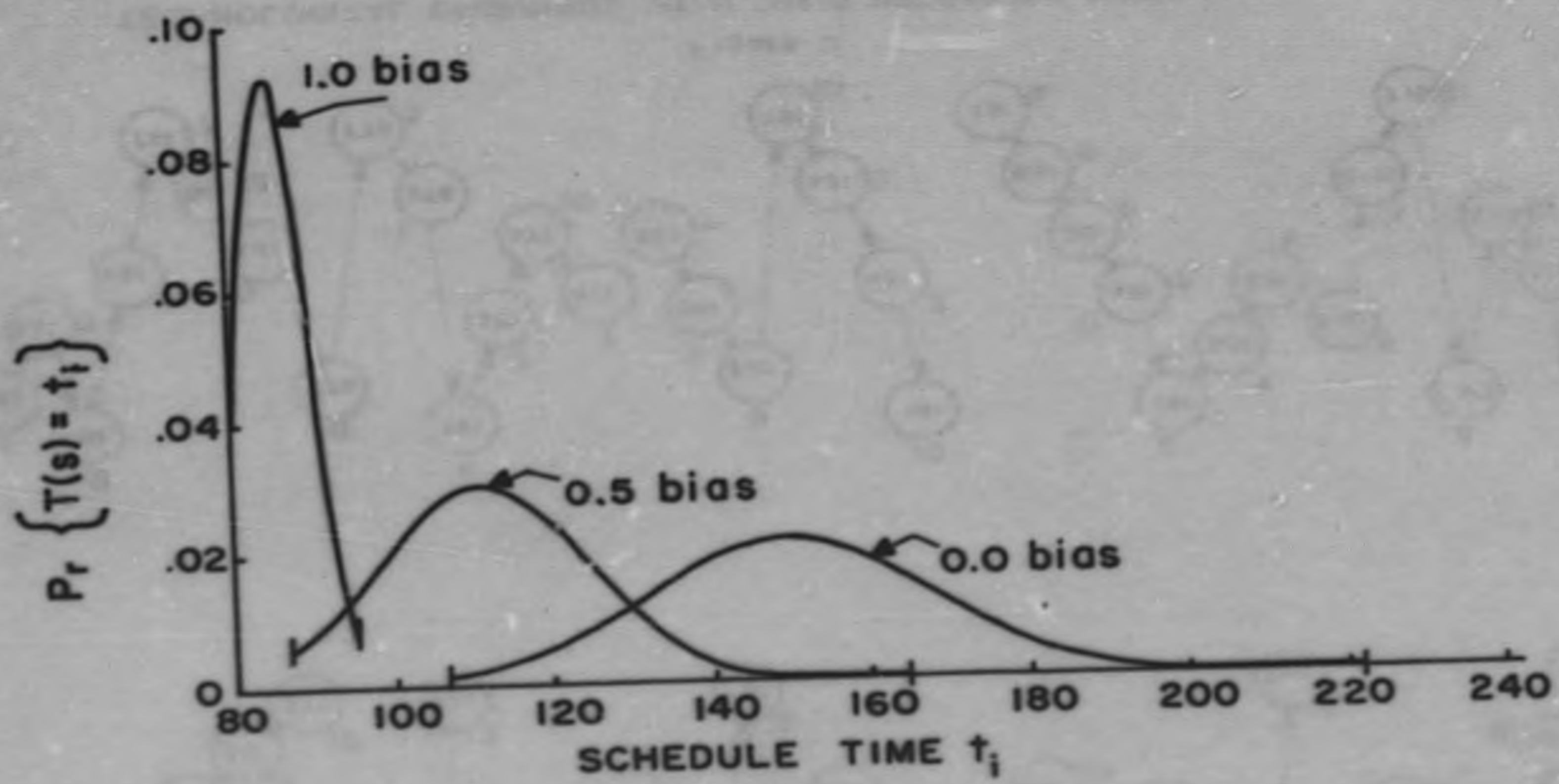


Figure 10

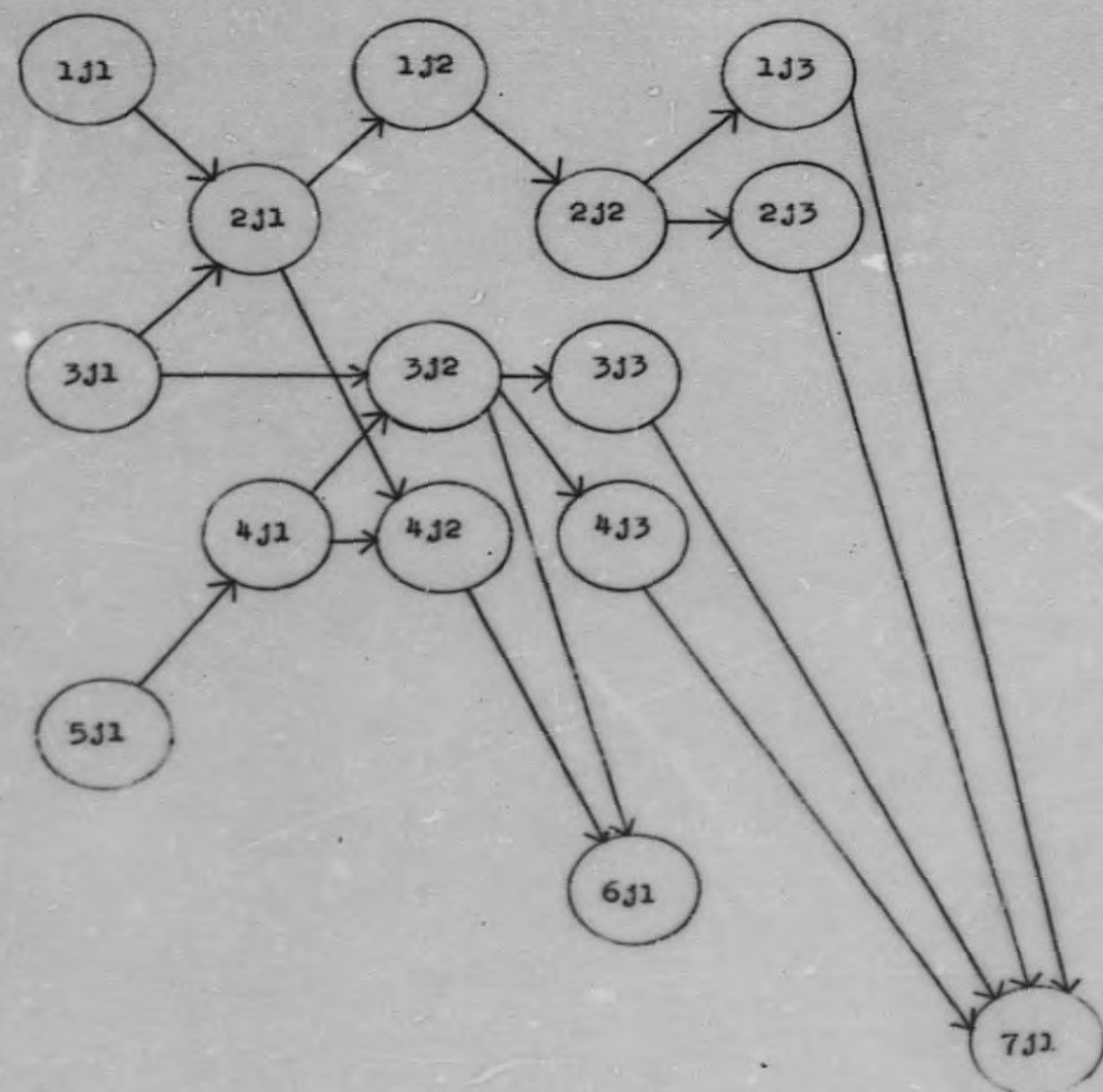


Figure 11

An example of a technological ordering in which the job can divide on and return to some of the machines.

462 031

END