MANAGEMENT SCIENCE
Vol. 17, No. 12, August, 1971
Printed in U.S.A.

AN ALGORITHM FOR OPTIMAL PROJECT SCHEDULING UNDER MULTIPLE RESOURCE CONSTRAINTS*

EDWARD W. DAVIS† AND GEORGE E. HEIDORN‡

An algorithm is described which permits the computation of optimal (minimum-duration) solutions for the resource-constrained project network scheduling problem under conditions of multiple resource requirements per job (activity). The approach is a form of bounded enumeration and employs techniques originally developed for the solution of the assembly line balancing problem. Initial computational results are given, along with an example of the type problems solved. Among the advantages of the procedure are that resource requirements can vary over job duration, and various assumptions concerning job continuity are allowable, with no extra computational effort.

1. Introduction

This paper is concerned with the project network scheduling problem of minimizing project duration under resource constraints. This problem arises when there are limited amounts of the resources required by the activities of the project. Because the amounts available are not always sufficient to satisfy the demands of concurrent activities, sequencing decisions are required, with a resultant increase in project duration beyond that determined by the "Critical Path" (the resource nonconstrained duration). The objective is to make the required sequencing decisions in such fashion that the increase in project duration is minimized, subject to the given resource and precedence constraints.

Davis [3] surveys the work on this problem (and other versions of it) through 1965, and comments on the absence of analytical (i.e., exact) solution procedures. Since then, Mueller-Mehrbach [12] and Johnson [9] have separately presented Branch-and-Bound solution procedures which permit the calculation of optimum solutions for this problem. These procedures are, however, limited to the simplest version involving requirements of only one resource type per job, no job interruptions, and constant resource requirements over every time period of job duration.

The particular case we are concerned with is the so-called "multiresource" problem, in which more than one resource type may be required by every job in the project and the availability of each type is limited. Except for trivial-sized examples, the multiresource case has so far resisted all attempts at analytical formulation in a manner conducive to rapid computation. Integer linear programming formulations have existed for some years, but this approach has proved to be both unreliable and computationally expensive, for network scheduling problems of even very modest size [1], [2], [11]. Alternatively, there abound a wide variety of approximate solution procedures whose "goodness" relative to the optimum solution have never been determined.

In this paper we describe a procedure for determining the optimal solution to problems involving requirements of several resource types per job. Among the advantages of this approach are that the resource requirements can vary over job duration, and

* Received October 1968; revised May 1970, August 1970, November 1970.
† Harvard University.
‡ Naval Postgraduate School, Monterey, California.

various assumptions concerning job continuity are allowed, with no extra computational effort.

First the formulation of the multiple-resource problem is given under the restrictive assumptions commonly imposed, and the procedure is presented for this version. Later we discuss how these restrictions may be relaxed to handle more general cases.

2. Formulation of the Problem

The problem may be stated as follows: a finite set of jobs is given, each of fixed *integer* duration, each requiring a fixed amount of one or more different resources, and subject to a set of precedence relations which specify permissible job orderings. Jobs may not be interrupted once started, and there are specified fixed limits on the availability of each resource. The objective is to minimize project duration, the time required to complete all jobs, subject to the given constraints.

Formulation of the problem for computational solution is facilitated by representing each original job of the project as a series of unit-duration *tasks*, equal in number to the job's duration, as shown in Figures 1 and 2.

To observe the constraint that jobs once begun may not be interrupted before completion, we specify that each task in a series representing an original job must *immediately* follow its predecessor, if one exists. For example, the relationships imposed on the tasks of Figure 2 from the jobs of Figure 1 are: $A1 \lessdot A2 \lessdot A3$, $A3 < B1$, $B1 \lessdot B2$, where "$\lessdot$" is defined as "must *immediately* precede" and $<$ denotes precedence in the normal sense.

We make the following definitions:

T_j = task j,

A = the set of all tasks in the project,

$$(T_1, T_2, \cdots),$$

Z_i = the set of tasks assigned on day i,

n = project duration, and let the requirements of each resource by task T_j be denoted by the *vector*

$$\bar{r}(T_j) = (\hat{r}_{1j}, \hat{r}_{2j}, \cdots, \hat{r}_{mj}), \tag{2.1}$$

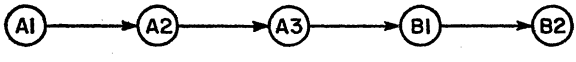
where $\hat{r}_{kj}$ = units of resource k required by task j, $k = 1, \cdots, m$. Let the resource requirements associated with each subset of tasks Z_i be denoted by the *vector*

$$\bar{R}_{Z_i} = \sum_{j \in Z_i} \bar{r}(T_j) = (r_{1i}, r_{2i}, \cdots, r_{mi}), \tag{2.2}$$

where r_{ki} = units of resource type k required for completion of the tasks in Z_i.

3 DAYS 2 DAYS

FIGURE 1. Original Jobs.



ALL DURATIONS = 1 DAY

FIGURE 2. Task Representation of Original Jobs.

Note that $\sum_{i=1}^{n} \bar{R}_{t_i} = \bar{R}_A$, the vector sum of project total resource requirement. Also, let the specified amounts of each resource be denoted by the *resource limits vector*

$$\bar{R}_L = (l_1, l_2, \cdots, l_m), \tag{2.3}$$

where l_j = number of units of resource type j available on each day of project duration.

Then, the problem can be formulated as: find a collection of subsets of A, i.e. $(Z_1, Z_2, \cdots, Z_n)$, which *minimizes* n subject to the following conditions:

$$\bigcup_{i=1}^{n} Z_i = A, \tag{2.4}$$

$$Z_1 \cap Z_j = \varnothing, \qquad \forall\, i \neq j, \tag{2.5}$$

$$\bar{R}_{Z_i} \leqq \bar{R}_L \tag{2.6}$$

$$\text{If } T_1 < T_2 \text{ and } T_1 \in Z_i, T_2 \in Z_j, \text{ then } i \leqq j. \tag{2.7}$$

$$\text{If } T_1 \in Z_i \text{ and } T_1 \overset{\bullet}{\ll} T_2, \text{ then } T_2 \in Z_{i+1}. \tag{2.8}$$

The above formulation is the project scheduling counterpart of the formulation of the assembly line balancing problem given in [5].

3. Transformation to a Shortest Route Problem

In a manner similar to that used in attacking the assembly line balancing problem [5], the conditions (2.4)—(2.8) above can be transformed into a problem of finding a shortest route between two specified nodes of a finite directed network.

In the new network (called the A-network), nodes represent subsets of tasks, and arcs connect subsets which could be completed on adjacent days. The minimization of project duration, then, is accomplished by finding a path from start to finish in the A-network which contains a minimal number of arcs.

The procedure will be illustrated by application to the sample problem shown in Figure 3, consisting of 2 dummy jobs and 5 real jobs which each require the indicated amounts of 3 different resource types. The corresponding task representation of the sample problem is shown in Figure 4. Note that nodes represent jobs and arcs indicate precedence.

4. Generation of Feasible Subsets

A subset of A, $S_j = (T_1, T_2, \cdots, T_m)$ is said to be *feasible* if $T_j \in S_j$ and $T_i < T_j$ imply that $T_i \in S_j$.[1] Thus, a feasible subset is one which, whenever it contains a given task, also contains all predecessors of that task. The tasks in a feasible subset may be performed in *some* sequence, consistent with the precedence relations, without the prior performance of any other task not in that subset.

This definition of a feasible subset follows that given in [7]. Both [7] and [5] give similar procedures for generating feasible subsets which can be applied to our problem; the procedure to be described here is based upon the latter.

First, however, note that although resource demands are not involved in the definition of a feasible subset, there is a *resource requirements vector* $\bar{R}_{s_i}$, defined similarly to (2.2), associated with every feasible subset, which is obtained by forming the sum of the resource requirements vectors of each task in subset S_i.

[1] Note that from here on the subset subscript j indicates the jth feasible subset created. This differs from the notation of §2 where the subscript indicated day j.

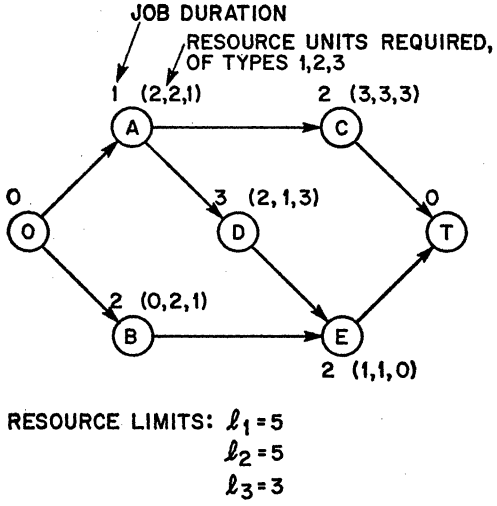


FIGURE 3. The Sample Problem.

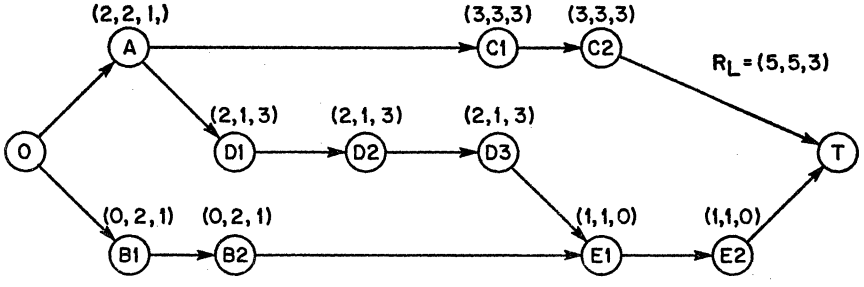


FIGURE 4. Task Representation of Sample Problem.

Now, let S_j, $j = 0, 1, 2, \cdots, r$, be the entire collection of feasible subsets, with $S_0 = \varnothing$, the empty set, and $S_r = A$. Let an *immediate follower* of a feasible subset S_j be defined as a task which is an immediate successor of at least one of the tasks in S_j and is not preceded by any tasks not in S_j. Then the generation of feasible subsets for a given problem proceeds exactly as described in [5]. An example of this procedure for the sample network shown in Figure 4 is given in Table 1.

This method of generating feasible subsets has the properties that (1) every feasible subset is generated, (2) no feasible subsets are duplicated, and (3) all task combinations generated are feasible subsets (for proof, see Gutjahr [6]). Two properties of the procedure when applied to project network scheduling are (see Davis [4] for proof):

(A) A task appears for the first time in feasible subsets generated in stage K, where K is the task early start time (EST).

(B) The number of stages is equal to the number of time periods in the longest path through the task precedence network (i.e., the "critical path").

5. Creation of the A-Network

Using the feasible subsets, the A-network can be constructed. The procedure followed is analogous to that in [5]. Starting with the empty set S_0, directed arcs are drawn between nodes if precedence and resource constraints as given below are satis-

TABLE 1

Feasible Subsets Generated by the Sample Problem

Marked tasks	K	j	Tasks in Subset	$\bar{R}_{sj}$	UIF
	0	0	empty set	(0, 0, 0)	*A*, *B*1
A,	1	1	*A*	(2, 2, 1)	*C*1, *D*1
*B*1		2	*B*1	(0, 2, 1)	*B*2
		3	*A* *B*1	(2, 4, 2)	*B*2, *C*1, *D*1
*B*2,	2	4	*A* *C*1	(5, 5, 4)	*C*2
*C*1,		5	*A* *D*1	(4, 3, 4)	*D*2
*D*1		6	*A* *C*1 *D*1	(7, 6, 7)	*C*2, *D*2
		7	*B*1 *B*2	(0, 4, 2)	
		8	*A* *B*1 *B*2	(2, 6, 3)	
		9	*A* *B*1 *C*1	(5, 7, 5)	*C*2
		10	*A* *B*1 *D*1	(4, 5, 5)	*D*2
		11	*A* *B*1 *B*2 *C*1	(5, 9, 6)	*C*2
		12	*A* *B*1 *B*2 *D*1	(4, 7, 6)	*D*2
		13	*A* *B*1 *C*1 *D*1	(7, 8, 8)	*C*2, *D*2
		14	*A* *B*1 *B*2 *C*1 *D*1	(7, 10, 9)	*C*2, *D*2
*C*2,	3	15	*A* *C*1 *C*2	(8, 8, 7)	
*D*2		16	*A* *D*1 *D*2	(6, 4, 7)	*D*3
		17	*A* *C*1 *C*2 *D*1	(10, 9, 10)	
		18	*A* *C*1 *D*1 *D*2	(9, 7, 10)	*D*3
		19	*A* *C*1 *C*2 *D*1 *D*2	(12, 10, 13)	*D*3
		20	*A* *B*1 *C*1 *C*2	(8, 10, 8)	
		21	*A* *B*1 *D*1 *D*2	(6, 6, 8)	*D*3
		22	*A* *B*1 *B*2 *C*1 *C*2	(8, 12, 9)	
		23	*A* *B*1 *B*2 *D*1 *D*2	(6, 8, 9)	*D*3
		24	*A* *B*1 *C*1 *C*2 *D*1	(10, 11, 11)	
		25	*A* *B*1 *C*1 *D*1 *D*2	(9, 9, 11)	*D*3
		26	*A* *B*1 *C*1 *C*2 *D*1 *D*2	(12, 12, 14)	*D*3
		27	*A* *B*1 *B*2 *C*1 *C*2 *D*1	(10, 13, 12)	
		28	*A* *B*1 *B*2 *C*1 *D*1 *D*2	(9, 11, 12)	*D*3
		29	*A* *B*1 *B*2 *C*1 *C*2 *D*1 *D*2	(12, 14, 15)	*D*3
*D*3	4	30	*A* *D*1 *D*2 *D*3	(8, 5, 10)	
		31	*A* *C*1 *D*1 *D*2 *D*3	(11, 8, 13)	
		32	*A* *C*1 *C*2 *D*1 *D*2 *D*3	(14, 11, 16)	
		33	*A* *B*1 *D*1 *D*2 *D*3	(8, 7, 11)	
		34	*A* *B*1 *B*2 *D*1 *D*2 *D*3	(8, 9, 12)	*E*1
		35	*A* *B*1 *C*1 *D*1 *D*2 *D*3	(11, 10, 14)	
		36	*A* *B*1 *C*1 *C*2 *D*1 *D*2 *D*3	(14, 13, 17)	
		37	*A* *B*1 *B*2 *C*1 *D*1 *D*2 *D*3	(11, 12, 15)	*E*1
		38	*A* *B*1 *B*2 *C*1 *C*2 *D*1 *D*2 *D*3	(14, 15, 18)	*E*1
*E*1	5	39	*A* *B*1 *B*2 *D*1 *D*2 *D*3 *E*1	(9, 10, 12)	*E*2
		40	*A* *B*1 *B*3 *C*1 *D*1 *D*2 *D*3 *E*1	(12, 13, 15)	*E*2
		41	*A* *B*1 *B*2 *C*1 *C*2 *D*1 *D*2 *D*3 *E*1	(15, 16, 18)	*E*2
*E*2	6	42	*A* *B*1 *B*2 *D*1 *D*2 *D*3 *E*1 *E*2	(10, 11, 12)	
		43	*A* *B*1 *B*2 *C*1 *D*1 *D*2 *D*3 *E*1 *E*2	(13, 14, 15)	
		44	*A* *B*1 *B*2 *C*1 *C*2 *D*1 *D*2 *D*3 *E*1 *E*2	(16, 17, 18)	

fied, until the final node S_r is reached. No arcs enter S_0 and none leave S_r ; the result is a directed network from S_0 to S_r on the set of all feasible subsets.

A directed arc $\overrightarrow{S_iS_j}$ is created from feasible subset S_i to subset S_j if and only if the conditions given below are met.

First, to observe the precedence constraints of the original problem, (a) S_i must be a proper subset of S_j (i.e. $S_i \subset S_j$) and (b) all predecessor tasks of a given task in subset S_j must be contained in subset S_i ; i.e., if $T_s \in S_j$ and $T_r < T_s$, then $T_r \in S_i$.

Second, to observe the constraint that jobs in the original network be performed continuously once started, additional "must follow" constraints must be imposed on task sequence. Thus, in constructing an arc $\overrightarrow{S_iS_j}$ from S_i to S_j , if $T_r \in S_i$ and $T_r \ll T_s$, then $T_s \in S_j$. This constraint has a two-way implication; that is, if $T_s \in S_j$ and $T_r \ll T_s$, then $T_r \in S_i$. In the case $T_r < T_s$, it is *not* necessary that $T_s \in S_j$.

Finally, it is required that $\bar{R}_{s_j} - \bar{R}_{s_i} \leqq \bar{R}_L$, where $\bar{R}_L$ is the *resource limits vector*, defined by (2.3).

Arcs constructed by observing the above criteria represent the assignment of tasks on particular schedule days. That is, the rth arc $\overrightarrow{S_iS_j}$ in a path from S_0 corresponds to the assignment of tasks $(S_j - S_i)$ on the rth day of the original problem. The proof of the feasibility of each such assignment is similar to that given in [5].

As an illustration of the method, the A-network for the example problem is shown in Figure 5. Each numbered node in the figure corresponds to the same number feasible subset given in Table 1. Not all of the possible arcs are shown, but the network is sufficient to illustrate the concepts involved. Shaded nodes in Figure 5 identify the shortest path, with 7 arcs, corresponding to a 7-day schedule. Table 2 shows the daily task assignments for this path.

6. Two Additional Properties of the Construction Procedure

Two additional properties of the procedure which substantially reduce the effort required in searching for arc connections are (see Davis [4] for proof):

(C) Directed arcs $\overrightarrow{S_iS_j}$ can exist only for the case $i < j$; furthermore, when $S_i \in$ stage K and $S_j \in$ stage $K + 1$, $i \leqq i'$, where i' is the index of the subset used in creating S_j .

(D) Arcs can exist only between feasible subsets in the same and adjacent stages.

Because of these properties, from a given node S_j of stage K, the search for possible arc connections can be limited to nodes S_p , $p > j$, in stage K and to nodes S_r , $r \geqq k$, in stage $K + 1$, where k is the index of the first new feasible subset in stage $K + 1$ created from S_j .

7. Techniques for Feasible Subset Elimination

The procedure as presented so far has one major drawback: the number of feasible subsets generated can be very large for even small problems, depending upon the duration of the original jobs and complexity (interconnections) of the network. For example, if the sample problem used here (Figure 4) is modified by increasing the duration of jobs A and B to 6 days each and adding one additional job F of 6 days' duration, the number of feasible subsets generated increases from 44 to 591.[2] Another 6-job network with exactly the same precedence relations but different job durations could generate several thousand feasible subsets.

One obvious means of improving the procedure, then, would be to apply techniques

[2] The exact number of subsets generated for any network by the method described in this paper can be determined by a simple counting procedure due to Held, Karp and Shareshian [8].

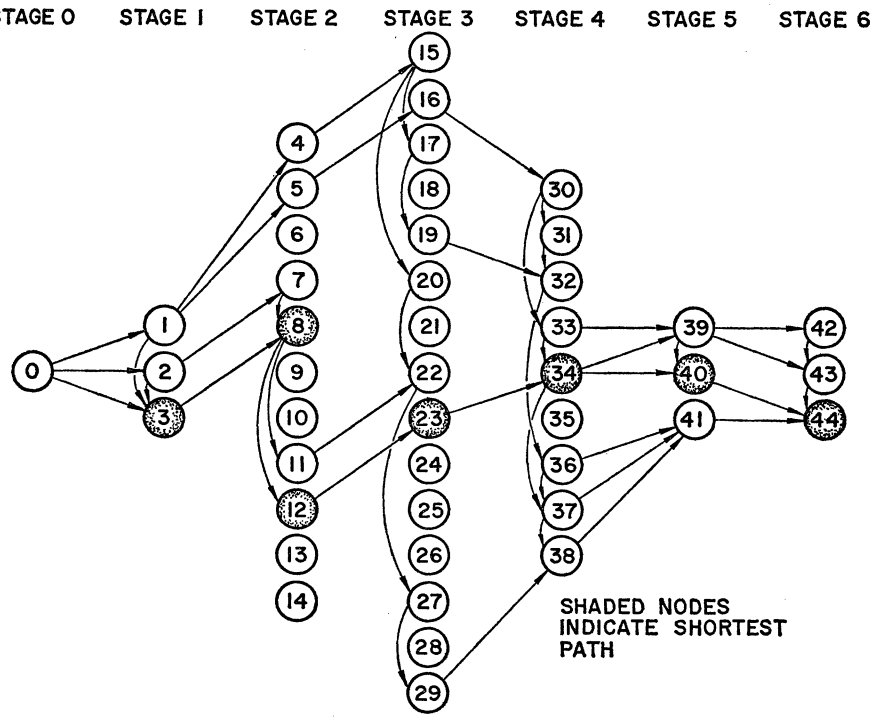


FIGURE 5. A-Network for the Sample Problem.

reducing the total number of feasible subsets generated, while still guaranteeing the production of an optimal solution if one exists. The rest of this paper will be devoted to the discussion of such techniques.

The basic approach is to select a "target" duration (n') for the project and then, during the performance of the algorithm, to discard those feasible subsets which

(a) could not themselves possibly appear on a path of length n' or less through the A-network, and

(b) could not generate any other feasible subsets which would, either.

Given a target duration, two independent sets of criteria, one based on the precedence constraints and the other on the resource constraints, can be readily determined and then applied to each feasible subset when it is generated.

7.1. *Precedence-Based Elimination Criteria*

By considering only the precedence constraints, it is possible to specify for each task the latest day on which it may be performed if the target duration is to be met. This is precisely the late-start-time computation of the usual critical path analysis. We call this day, so determined, the late performance time (LPT) of the task. Table 3 shows the LPT for each task of the sample problem of Figure 4 for a target duration of 6 days. Also, we define D_t to be the set of tasks with LPT $\leqq t$. Table 4 shows the D_t corresponding to Table 3.

From the properties of the subset generation procedure, we know that the number of arcs to reach a feasible subset of stage K from the initial node of the A-network is at least K. Since arcs correspond to days, a feasible subset S_j in stage K must include all the tasks of D_K, or it could not possibly appear on a path of length n' or less.

TABLE 2

Arc Number, t	Node Reached	Corresponding Feasible Subset	Task Assignment On Day t
7	44	A $B1$ $B2$ $C1$ $C2$ $D1$ $D2$ $D3$ $E1$ $E2$	$(S_{44} - S_{40}) = C2, E2$
6	40	A $B1$ $B2$ $C1$ $D1$ $D2$ $D3$ $E1$	$(S_{40} - S_{24}) = C1, E1$
5	34	A $B1$ $B2$ $D1$ $D2$ $D3$	$(S_{24} - S_{23}) = D3$
4	23	A $B1$ $B2$ $D1$ $D2$	$(S_{23} - S_{12}) = D2$
3	12	A $B1$ $B2$ $D1$	$(S_{12} - S_{8}) = D1$
2	8	A $B1$ $B2$	$(S_{8} - S_{3}) = B2$
1	3	A $B1$	$(S_{3} - S_{0}) = A, B1$

TABLE 3

Task	A	$B1$	$B2$	$C1$	$C2$	$D1$	$D2$	$D3$	$E1$	$E2$
LPT	1	3	4	5	6	2	3	4	5	6

TABLE 4

t	1	2	3	4	5	6
D_t	A	A $D1$	A $B1$ $D1$ $D2$	A $B1$ $B2$ $D1$ $D2$ $D3$	A $B1$ $B2$ $C1$ $D1$ $D2$ $D3$ $E1$	A $B1$ $B2$ $C1$ $C2$ $D1$ $D2$ $D3$ $E1$ $E2$

Therefore, if $D_K \not\subset S_j$, S_j can be discarded. (It is proven in [4] that any subsets which would have been generated from this S_j are similarly of no interest.)

Actually, there is a better lower bound than K on the minimum number of arcs required to reach feasible subset S_j of stage K. It can be obtained as follows:

$$K^* = \min_{a \leqq j^* \leqq b} [N_{K-1}(j^*)] + 1, \tag{7.1.1}$$

K^* = better lower bound on the number of days required to execute the tasks comprising feasible subset S_j of stage K, where $N_{K-1}(j^*)$ = number of arcs on the shortest path from S_0 to feasible subset S_{j^*} of stage $K - 1$. a = node index of the 1st feasible subset in stage $K - 1$. b = node index of the parent feasible subset of S_j, in stage $K - 1$.

It is proven in [4] that this tighter bound still restricts elimination to those subsets which satisfy both conditions (a) and (b) given above.

7.2 *Resource-Based Elimination Criteria*

By considering *only* the resource constraints and knowing the total amounts of resources required to perform the entire project, it is possible to specify for each day the minimum amounts of each resource which must already be expended on the project up through that day, in order that the remaining resource requirements be small enough to be accommodated on the remaining days within the target duration. The vector $\bar{W}_t$ of minimum required resource usage through day t is defined as follows:

$$\bar{W}_t = [\omega_{t1}, \omega_{t2}, \cdots, \omega_{tj}] \tag{7.2.1}$$

where ω_{tj} = minimum required resource units usage of resource type j through day t, calculated as follows:

$$\omega_{tj} = \max [0, w_j - (n' - t)l_j]$$

and l_j, n' are as previously defined.

Appropriate values of $\bar{W}_t$ are used for feasible subset elimination in a manner similar to the LPT criteria. That is, at stage K of generation those subsets not satisfying the criterion $\bar{R}_S \geqq \bar{W}_{K^*}$ are discarded. Values of $\bar{W}_t$ for the sample problem with a target duration of 6 days are given in Table 5 below.

8. Choosing the Target-Duration

The two types of elimination criteria described above are complementary in application. Thus, at stage K of feasible subset generation we keep only those feasible subsets S_j for which $\bar{R}_{S_j} \geqq \bar{W}_{K^*}$ *and* $D_{K^*} \subset S_J$.

Use of the elimination criteria in Tables 4 and 5 will not produce a solution for the example problems, since a 6-day solution does not exist. Therefore, it is necessary to determine the elimination criteria for a target duration of 7 days and again attempt solution. The criteria for $n' = 7$ are given in Table 6. (Note how it compares with Tables 4 and 5.) In this example, we know that a 7-day solution will be obtained, but in general it would be necessary to make repeated trial solutions using elimination criteria based upon target-durations of successively increasing 1-day increments until a solution is obtained. An alternate approach is that of starting with the solution produced by a good heuristic and testing for the existence of a shorter-duration schedule.

9. Further Reductions in Computational Effort

The elimination technique as described so far results in some feasible subsets being discarded because they satisfy *both* conditions (a) and (b) of §7. A further reduction in computational effort may be achieved by segregating those that are not discarded into two groups:

(1) those that could appear on a path of n' or less through the A-network, and

(2) those that could not.

The reason why members of group (2) cannot simply be discarded is that they may result in the generation of subsets in later stages which would belong to group (1).

TABLE 5

t	1	2	3	4	5	6
$\bar{W}_t$	(0, 0, 3)	(0, 0, 6)	(1, 2, 9)	(6, 7, 12)	(11, 12, 15)	(16, 17, 18)

TABLE 6

t:	1	2	3	4	5	6	7
$\bar{W}_t$:	(0, 0, 0)	(0, 0, 3)	(0, 0, 6)	(1, 2, 9)	(6, 7, 12)	(11, 12, 15)	(16, 17, 18)
D_t:	ϕ	A	A	A	A	A	A
				$B1$	$B1$ $B2$	$B1$ $B2$	$B1$ $B2$
			$D1$	$D1$ $D2$	$D1$ $D2$ $D3$	$C1$	$C1$ $C2$
						$D1$ $D2$ $D3$	$D1$ $D2$ $D3$
						$E1$	$E1$ $E2$

This segregation can be accomplished during the A-network construction by creating a list at each stage, L_K, composed of the indices of those feasible subsets which are members of group (1). These subsets are identified by the fact that they satisfy the criteria $\bar{R}_{S_j} \geqq W_{K'}$ and $D_{K'} \subset S_j$, where $K' = N_K(j)$ (the minimum number of days required to perform the tasks in subset S_j of stage K).

The search for arc connections into a newly generated node S_j of stage K can be restricted to those nodes of stage $K-1$ appearing on the list L_{K-1}, and any previously-generated nodes of stage K with indices already on the list L_K.

We present below a Dynamic Programming approach which utilizes these ideas to perform the shortest-path determination during construction of the A-network. In this approach only one arc into each node is retained during construction. $N_K(j)$ is computed as follows:

$$N_K(j) = \min \begin{bmatrix} \min_{j^* \in L_{K-1}} [\delta + N_{K-1}(j^*)] \\ \min_{j' \in L_K} [\delta + N_K(j')] \end{bmatrix}, \tag{9.1}$$

where $a \leqq j^* \leqq b$, $c \leqq j' \leqq j$, a, b, c, δ, S_i are defined as below: and the node index of subset S_j is added to list L_K (i.e. $j \in L_K$)

$$\text{IFF } \bar{R}_{S_j} > \bar{W}_{K'} \quad \textit{and} \quad D_{K'} \subset S_j$$

for $K' = N_K(j)$.

Definitions.

$$\begin{aligned} \delta = 1 \quad & \text{IFF } S_i \subset S_j \\ & \text{IFF } T_r \in S_i, \text{ where } T_r < T_s \text{ and } T_s \in S_j \\ & \text{IFF } T_s \in S_j, \text{ where } T_r \in S_i \text{ and } T_r \ll T_s \\ & \text{IFF } \bar{R}_{s_j} - \bar{R}_{s_i} \leqq \bar{R}_L \\ = \infty \quad & \text{otherwise,} \end{aligned}$$

and

a = index of 1st node in stage $K-1$,
b = index of the "parent" node in stage $K-1$,
c = index of the 1st node in stage K,
S_i represents the potential node from which the arc will originate.

The computational procedure goes as described earlier from node S_0, for each feasible subset surviving the first elimination tests (i.e. using values D_{K^*}, $\bar{W}_{K^*}$). Those nodes in stage 1 with $N_1(j) = 1$ will be identified as members of L_1, as well as any nodes with $N_1(j) > 1$ which satisfy the $\bar{W}_{K'}D_{K'}$ criteria for their respective values of $K' = N_1(j)$. Nodes with $N_1(j) > 1$ not satisfying both criteria are numbered in their order of generation and used for generating feasible subsets in stage 2, as before, but these indices are not added to list L_1. As each node of stage 2 surviving the first elimination is generated, the calculation of $N_2(j)$ is made searching only over those nodes of stage 1 indicated by list L_1 as well as any previously-generated modes of stage 2 already on list L_2, which is being created. These node indices of stage 2 for which $N_2(j) > 2$ are added to L_2 only if both K' tests are satisfied. The procedure continues in a similar manner for all successive stages until the final node is reached, at which point the shortest path is traced as before.

Because arc connections are now made only between feasible subsets of each stage

indicated by the respective lists L_K, the computation of K^*, the lower bound on $N_K(j)$ for any feasible subsets S_j of stage K, is changed as follows:

$$K^* = \min_{a \leq j^* \leq b;\ j^* \in L_{K-1}} [N_{K-1}(j^*)] + 1 \tag{9.2}$$

where a, b and L_{K-1} are defined as before.

There is still more feasible subset elimination which can be performed in addition to that already described, but *only* in regard to feasible subsets with node indices *not* contained in any list L_K. There are three categories of such feasible subsets which may be eliminated completely, without destroying optimality. They are:

(a) Feasible subsets with no unmarked immediate follower tasks.

(b) Feasible subsets at stage K "above" the first feasible subset identified by the list L_K.

(c) Feasible subsets of stage K which will generate no feasible subsets of stage $K+1$ capable of satisfying the W_{K^*}, D_{K^*} tests at stage $K+1$.

The appropriate proofs and a discussion of the elimination rule for each category above are given in [4].

10. Elimination Techniques Applied to the Sample Problem

By applying the techniques described above to the sample problem of Figure 4 using the elimination criteria of Table 6, the reduced set of feasible subsets shown in Table 7 is obtained. The new A-network and optimal task assignments are shown in Figure 6 and Table 8, where it may be seen that the same schedule as before is obtained.

TABLE 7

Reduced Set of Feasible Subsets for the Sample Problem, After Elimination.

Stage	Subset No.	Tasks in Subset	$\bar{R}_{sj}$	K^*
0	0	empty set ∅		
1	1	*A*	(2, 2, 1)	1
	2	*B*1	(0, 2, 1)	1
	3	*A* *B*1	(2, 4, 2)	1
2	4	*A* *C*1	(5, 5, 4)	2
	5	*A* *D*1	(4, 3, 4)	2
	6	*A* *C*1 *D*1	(7, 6, 7)	2
	7	*A* *B*1 *B*2	(2, 6, 3)	2
	8	*A* *B*1 *D*1	(4, 5, 5)	2
	9	*A* *B*1 *B*2 *D*1	(4, 7, 6)	2
	10	*A* *B*1 *C*1 *D*1	(7, 8, 8)	2
	11	*A* *B*1 *B*2 *C*1 *D*1	(7, 10, 9)	2
3	12	*A* *D*1 *D*2	(6, 4, 7)	3
	13	*A* *B*1 *D*1 *D*2	(6, 6, 8)	3
	14	*A* *B*1 *B*2 *D*1 *D*2	(6, 8, 9)	3
	15	*A* *B*1 *C*1 *D*1 *D*2	(9, 9, 11)	3
	16	*A* *B*1 *C*1 *C*2 *D*1 *D*2	(12, 12, 14)	3
	17	*A* *B*1 *B*2 *C*1 *C*2 *D*1 *D*2	(9, 11, 12)	3
	18	*A* *B*1 *B*2 *C*1 *C*2 *D*1 *D*2	(12, 14, 15)	3
4	19	*A* *B*1 *B*2 *D*1 *D*2 *D*3	(8, 9, 12)	4
	20	*A* *B*1 *B*2 *C*1 *D*1 *D*2 *D*3	(11, 12, 15)	4
	21	*A* *B*1 *B*2 *C*1 *C*2 *D*1 *D*2 *D*3	(14, 15, 18)	4
5	22	*A* *B*1 *B*2 *C*1 *D*1 *D*2 *D*3 *E*1	(12, 13, 15)	6
	23	*A* *B*1 *B*2 *C*1 *C*2 *D*1 *D*2 *D*3 *E*1	(15, 16, 18)	6
6	24	*A* *B*1 *B*2 *C*1 *C*2 *D*1 *D*2 *D*3 *E*1 *E*2	(16, 17, 18)	7

Table 7 shows that the total number of feasible subsets is reduced from 44 to 24. Furthermore, as indicated in Figure 6, only 12 of these 24 are candidates for arc connections (i.e., with node indices belonging to some list L_K). As a result, only 88 pairwise comparisons of feasible subsets are required in making arc connections and finding the shortest path, which contrasts with 415 pairwise comparisons required for the same problem with no elimination (Figure 5). Note also that the last column of Table 7 indicates the computational advantages of using K^* as a lower bound on $N_K(j)$ instead of K.

11. Heuristic Setting of Elimination Criteria

Instead of determining the elimination criteria from the properties of a critical path type solution, the criteria can be established heuristically to force early elimination of *apparently* nonoptimal alternatives. In this way, it is sometimes possible to produce quickly a solution of shorter duration than the starting solution. But because the resulting solution, if one is found, may not be optimal, it is necessary to solve again, using the usual elimination criteria. In addition to being the only means of approach to some problems, this line of attack can sometimes result in lower total computation time on problems which can be handled in the normal manner. Examples of this approach are given in [4].

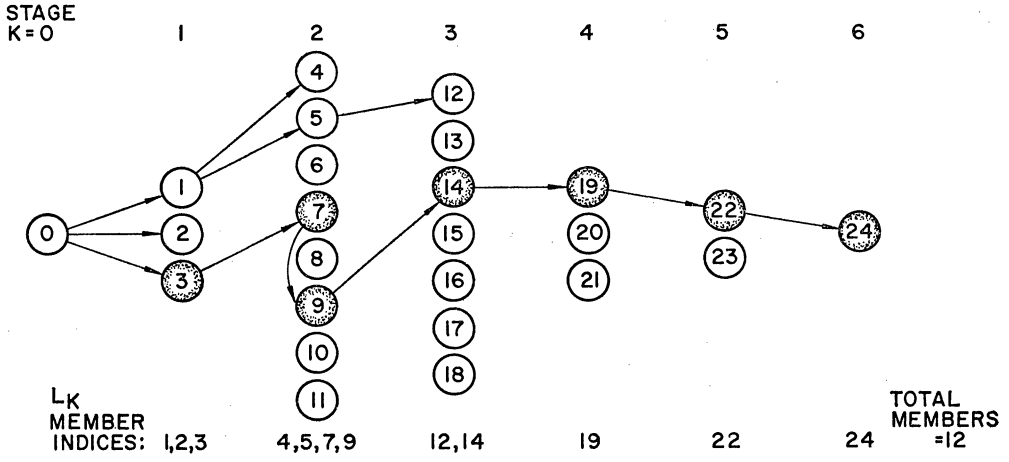


FIGURE 6. Reduced A-Network after Feasible Subset Elimination.

TABLE 8

Task assignments corresponding to path shown in Figure 6.

Day	Task Assignment
7	$(S_{24} - S_{22}) = C2, E2$
6	$(S_{22} - S_{19}) = C1, E1$
5	$(S_{19} - S_{14}) = D3$
4	$(S_{14} - S_{9}) = D2$
3	$(S_{9} - S_{7}) = D1$
2	$(S_{7} - S_{3}) = B2$
1	$(S_{3} - S_{0}) = A, B1$

12. Applications of the Procedure to More General Cases

Variations in Job Continuity

The assumption that jobs must be performed continuously once started is not a necessary requirement of the algorithm. Instead, this assumption was imposed to conform with those generally associated with the version of the project network scheduling problem most often discussed in the literature. If it can be assumed that jobs may be interrupted and restarted later at no cost, then such job "splitting" can easily be handled, at little or no increase in computational requirements for a given problem.

Nonconstant Resource Requirements

Another assumption which may be relaxed is that of constant job resource requirements. Instead of a uniform resource usage rate over the duration of the job the resource requirements may vary from day to day, without affecting the computing time.

13. Computer Program

The algorithm described has been programmed in MAD for the IBM 7094 computer to provide a mechanism for testing the computational feasibility of the procedure. It was designed to handle small problems involving no more than about 25–35 jobs (a maximum of 105 tasks) and 3 different resource types per job.[3]

The primary operating constraint of the program is the number of feasible subsets which may be stored at each stage after elimination, and this number is a function of the number of unit-duration tasks in the problem. For example, when the number of tasks is between 71 and 105, a total of 906 feasible subsets per stage is allowed. If at some stage the allowable number of feasible subsets is exceeded, an optimal solution cannot be guaranteed because when this occurs, the program selects the "most promising" feasible subset from the previous stage (in effect, "freezing" part of the schedule) and then continues based on that. In this way a target-duration-or-shorter schedule can often be obtained, but it is not necessarily optimal.

For initial testing of the algorithm's performance, 65 artificially-constructed problems have been attempted, each containing between 50 and 95 unit-duration tasks (30 original jobs) and involving 3 different resource types per job and project. Optimal solutions were found for 48 of these, and for the remaining 17 an approximate solution was found which was at least one day shorter than the initial solution provided the program.[4] The solution time for the 48 optimally-solved problems varied from 2 seconds to almost 3 minutes, with an average time of .94 minutes and a variance of $(.34)^2$ minutes. The computation time for the 17 approximate solutions ranged from 25 seconds to 1.5 minutes, with an average time of .88 minutes and variance of $(.04)^2$ minutes.

References

1. Balinski, M. L., "Integer Programming: Methods, Uses, Computation," *Management Science* (November 1965).
2. Brand, J. D., Meyer, W. L. and Schaffer, L. R., "The Resource Scheduling Problem in Construction," Civil Engineering Studies, Report No. 5, Dept. of Civil Engineering, University of Illinois, Urbana, 1964.

[3] Since the original submission of this paper, a new program has been written for the IBM 360 which will handle up to 220 tasks and 5 different resource types per job. Tests are now under way with the new program.

[4] The initial solution for each problem was found using the "RSM" heuristic of Brand, Meyer and Shaffer [2].

3. DAVIS, E. W., "Resource Allocation in Project Network Models—A Survey," *Journal of Industrial Engineering* (April 1966).
4. ——, "An Exact Algorithm for the Multiple Constrained-Resource Project Scheduling Problem," unpublished Ph.D. thesis, Dept. of Administrative Sciences, Yale University, New Haven, May, 1968.
5. GUTJAHR, A. L. AND NEMHAUSER, G. L., "An Algorithm for the Line-Balancing Problem," *Management Science* (November 1964).
6. ——, "An Algorithm for the Assembly-Line Balancing Problem," unpublished M.S. thesis, Johns Hopkins University, Baltimore, 1963.
7. HELD, M. AND KARP, R. M., "A Dynamic Programming Approach to Sequencing Problems," *Journal of the Society for Industrial and Applied Mathematics* (March 1962).
8. ——, —— AND SHARESHIAN, W. S., "Assembly Line Balancing: Dynamic Programming with Precedence Constraints," *Operations Research* (May–June 1963).
9. JOHNSON, T. J. R., "An Algorithm for the Resource-Constrained Project Scheduling Problem," unpublished Ph.D. thesis, School of Management, M.I.T., August, 1967.
10. KELLEY, J. E., *Industrial Scheduling*, Chapter 21, (Ed. Muth and Thompson), Prentice-Hall, 1963.
11. MOODIE, C. L. AND MANDEVILLE, D. E., "Project Resource Balancing by Assembly Line Balancing Techniques," *Journal of Industrial Engineering* (July 1966).
12. MUELLER-MERBACH, H., "Ein Verfahren Zur Planung des Optimalen Betriebsmitteleinsatzes bei der Terminierung von Grossprojekten," *Zeitschrift fuer Wirtschaftliche Fertigung*, Heft 2 and Heft 3 (February and March 1967).