Computer techniques, now employed by management in the planning, scheduling and control of projects, generally rely on the formulation of project "networks" as input to the computer programs.

This paper discusses a computer procedure for improving the input by obtaining networks with certain minimal properties.

With this improvement in input, the overall efficiency in using existing programs can be increased.

Computer construction of minimal project networks

by Bernard Dimsdale

PERT[1,2], LESS[3], CPM[4,5] and other techniques applied to the planning, scheduling and control of projects involve the construction of project representations called *networks* which pictorialize the relationships among the component activities of the particular project under consideration.

For example, in fabricating a structure the activities may involve: building forms, erecting steel, pouring concrete, procuring materials and equipment, etc. Whether "pouring concrete" is a single activity which occurs just once, or a single activity which is repeated as construction proceeds, or a complex of component activities involving ordering sand, delivery, cement mixing, etc. depends on the specific nature of the project at hand and is of no interest here. The point is that project planning involves specifying the component activities of the project, whatever they may be, and specifying the order in which they occur.

Once the list of activities is obtained and the order in which they are to occur has been specified, an *activity* network may be drawn in which the *nodes* designate *activities* and the *branches* (arrows) connecting nodes indicate that the activity represented by initial node precedes the activity represented by terminal node.

To illustrate, assume that a project involves activities A, B, C and D, and suppose that A must precede both C and D and that B must precede D. Figure 1 shows the corresponding activity network and a table containing the same information.

In the table, **A**, **P** and **F** denote *activity*, *preceding activity* and *following activity*, respectively.

Scheduling, on the other hand, involves assigning time for the beginning and the end of each component activity. For this purpose, a different kind of network is useful. Here the *nodes* represent *points in time* and the *branches* represent *activities*, so that an activity starts at the time assigned to its initial node and ends at the time assigned to its terminal node. This kind of network is called an *event* network or a *project* network. It is to this kind of network that the above techniques are usually applied for scheduling purposes.

Corresponding to the project introduced in the above example, we may assign t_1 and t_2, t_2 and t_3, t_4 and t_5, t_5 and t_6 as the beginning and completion times for activities A, C, B, D, respectively. Thus we have:

$$t_1 < t_2,\ t_2 < t_3,\ t_4 < t_5 \text{ and } t_5 < t_6,$$

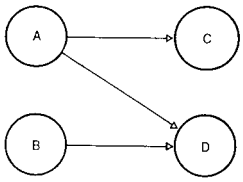
and also

$$t_2 \leq t_5,$$

since activity A precedes activity D. If we insist that each branch of an event network represent an activity, it will be apparent that there is no method of drawing an event network which includes the information, $t_2 \leq t_5$. To overcome this difficulty, we permit the introduction of branches, whose only function is to represent the passage of time (possibly zero) and say that such branches represent *dummy* activities. With this agreement, we may draw the event network shown on the left side of Figure 2 involving the dummy activity E, thereby including the information, $t_2 \leq t_5$. Although the agreement makes it possible to represent any project with an event network, this representation is never unique. For example in the present instance, the event network shown on the right in Figure 2, involving the dummy activities F and G, also correctly represents the project. It will be clear that there are an unlimited number of different sized event network representations for a given project. Of course, it is not always necessary to introduce dummies, but in the case of projects of the complexity encountered in practice, dummies are nearly always required.

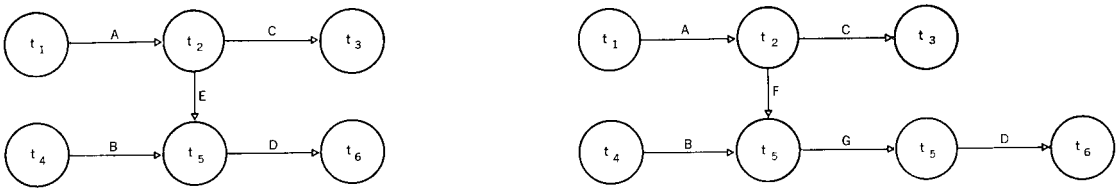
On the other hand, we will show by example that given an event network, it is always possible to find a corresponding unique

Figure 1 Activity network and its tabular form



A	P	F
A	—	$C\ D$
B	—	D
C	A	—
D	$A\ B$	—

Figure 2 Equivalent event networks



activity network (free of dummy activities) describing the same project. Thus we have a semi-duality between activity and event networks.

It is obvious that the efficiency of techniques based on event networks will be strongly influenced by the size of the network employed. Though the problem of finding the smallest network is transparent in the above example, it is not a trivial problem for projects of realistic size. In general, the number of nodes and branches utilized in an event network to represent a given project will vary widely, depending upon the experience and ingenuity of those constructing the network.

content of paper

The purpose of this paper is to describe a procedure which will permit, for a given project, computer construction of a corresponding event network with a minimal number of nodes. The basic theory is developed in another paper[6] where it is shown that, in general, minimization of the number of nodes does not simultaneously minimize the number of branches. In this same paper, a necessary and sufficient condition for simultaneous minimization is stated, and it appears likely that most practical projects would be consistent with the condition. If the condition is not satisfied, then event networks can be constructed with fewer dummies and more nodes, but these nodes will be incident to dummy branches exclusively. In this case, the networks with the minimum number of dummies may or may not have unique structure.

The procedure of constructing event networks to be described in this paper requires as data: project description by means of a list indicating for each activity all immediately following activities, and another list indicating for each activity all immediately preceding activities, with each of the lists ordered by levels and free of cycles and redundancy, as explained later. The basic precedence data required for the formation of the above lists is contained in tables of the type illustrated in Figure 1. It should be noted that the columns of this type of table are not independent—the second can be generated from the first and third, or the third from the first and second.

Sometimes the problem of finding a minimal event network to replace a given event network will occur, and on other occasions part of the data describing the project may be in event network form.

In the following section a method of converting from event networks to activity networks is given. The transformation of the precedence data to the required form (including checks for redundancy, cycles and other inconsistencies) is detailed in the next sections. Then, the method of construction of minimal event networks is given. The final sections are devoted to computer programming considerations.

conversion from event to activity network

The method of conversion will be illustrated by converting the event network given in Table 1, where **I** and **T** denote *initial* and *terminal* nodes, respectively, and the subscripted D's denote *dummy* activities.

Table 1

A	I	T	A	I	T
A	1	2	N	12	13
B	1	3	P	10	12
C	1	4	Q	2	10
D	2	5	D_1	6	11
E	2	6	D_2	4	11
F	4	8	D_3	1	6
G	4	6	D_4	8	10
H	3	7	D_5	3	5
I	4	7	D_6	5	11
J	6	9	D_7	9	12
K	7	9			
L	11	13			
M	10	1			

Table 2

N	P	F
1	M	$A\ B\ C\ D_3$
2	A	$D\ E\ Q$
3	B	$H\ D_6$
4	C	$F\ G\ I\ D_2$
5	$D\ D_5$	D_6
6	$E\ G\ D_3$	$J\ D_1$
7	$H\ I$	K
8	F	D_4
9	$J\ K$	D_7
10	$Q\ D_4$	$M\ P$
11	$D_1\ D_2\ D_6$	L
12	$P\ D_7$	N
13	$L\ N$	—

Table 3

A	P	F
D_1	$E\ G\ D_3$	L
D_2	C	L
D_3	M	$J\ D_1$
D_4	F	$M\ P$
D_5	B	D_6
D_6	$D\ D_5$	L
D_7	$J\ K$	N

Table 4

A	P	F
D_1	$E\ G\ M$	L
D_2	C	L
D_3	M	$J\ L$
D_4	F	$M\ P$
D_5	B	L
D_6	$D\ B$	L
D_7	$J\ K$	N

Table 5

N	P	F
1	M	$A\ B\ C$
		$J\ L$
2	A	$D\ E\ Q$
3	B	$H\ L$
4	C	$F\ G\ I\ L$
5	$D\ B$	L
6	$E\ G\ M$	$J\ L$
7	$H\ I$	K
8	F	$M\ P$
9	$J\ K$	N
10	$Q\ F$	$M\ P$
11	$E\ G\ M$	
	$C\ B\ D$	L
12	$P\ J\ K$	N
13	$L\ N$	—

Table 6

A	P	F
A	M	$D\ E\ Q$
B	M	$H\ L$
C	M	$F\ G\ I\ L$
D	A	L
E	A	$J\ L$
F	C	$M\ P$
G	C	$J\ L$
H	B	K
I	C	K
J	$M\ E\ G$	N
K	$H\ I$	N
L	$M\ B\ C$	
	$D\ E\ G$	—
M	$F\ Q$	$A\ B\ C$
		$J\ L$
N	$J\ K\ P$	—
P	$F\ Q$	N
Q	A	$M\ P$

The first step is to form Table 2, which lists the nodes (**N**) together with the preceding and following activities for each node.

After forming Table 2, the next step is to eliminate its dummy activities. To accomplish this, a series of tables listing the dummies together with each of their preceding and following activities is constructed. The first such table, Table 3, is written by consulting Tables 1 and 2 as follows: for the first row, from Table 1, D_1 has initial node 6 which from Table 2 is preceded by activities E, G and D_3; similarly D_1 has terminal node 11 which is followed by activity L; etc. The second table in the series, Table 4, is obtained from Table 3 by replacing the dummy activities appearing in the second and third columns (again consulting Tables 1 and 2). This replacement process is continued until a table with no dummies in the second and third columns is obtained (as in Table 4, so that in the present case the process is complete), or until a dummy occurs in the second or third column which is identical with a dummy in the first column and same row. In the latter case, there is an error in the data and no progress can be made until it is corrected. Assuming, as here, that this does not happen, the last table in the series (Table 4) is used in replacing the dummy activities in Table 2 to obtain Table 5. At this point, the required tabular form of the activity network (Table 6) can be produced since, for example, the first row of Table 5 says that: "M is followed by A, B, D, J, L"; "A is preceded by M"; "B is preceded by M"; etc.

Assuming that the input data describing the project is in activity precedence form, there are four cases:

Case 1. The data may be given as a list of activities, together with the preceding and following activities for each item in the list.

Case 2. Only preceding activities may be given for each activity.

Case 3. Only following activities may be given for each activity.

Case 4. Mixtures of Cases 1, 2 and 3 are provided.

transformation of precedence input data

Table 7

A	P	F	Case
A	—	B C D	1
B	—	E G	3
C	A	E I G H	1
D	—	F G	3
E	B C	—	2
F	—	I	3
G	—	J	3
H	—	J	3
I	B C E F	K	1
J	G H	K	1
K	I J	—	1

Table 8

A	P	F
A	—	C
B	A	E I
C	A	E I
D	A	—
E	B C	I
F	D	I
G	B C D	J
H	C	J
I	C F	K
J	G H	K
K	I J	—

Table 9

A	P	F
A	—	B C D
B	A	E G
C	A	E I G H
D	A	F G
E	B C	I
F	D	I
G	B C D	J
H	C	J
I	C E F	K
J	G H	K
K	I J	—

Table 10

A	P	F
A	D	C
B	J	F H
C	A F L	—
D	—	A E K
E	D	F L
F	B E	C H
G	H	—
H	B F	G I
I	H K	—
J	—	B K
K	D J	I
L	E	C

Table 11

IL	A	P	F
0	D	—	A E K
0	J	—	B K

Table 12

A	P
A	—
B	—
C	A F L
E	—
F	B E
G	H
H	B F
I	H K
K	—
L	E

Table 13

IL	A	P	F
0	D	—	A E K
0	J	—	B K
1	A	D	C
1	B	J	F H
1	E	D	F L
1	K	D J	I

Table 14

A	P
C	F L
F	—
G	H
H	F
I	H
L	—

Assuming for the moment that Case 2 applies, it is not difficult to complete the tabular form of the activity network; for if activity A is preceded by activity B, then activity B is followed by activity A. Of course, it is possible that some listed activity is preceded by an activity which is not listed as such, in which case the data is in error. It is also possible that an activity is indicated as preceded by itself, which again is an error. Precisely similar remarks apply to Case 3. In Case 1 the same two kinds of errors may be present. In addition, the list of following activities may not be consistent with the list of preceding activities, in which case discrepancies exist.

In Case 4, the data should be placed in a single table (with each entry designated by Case for later error checking). Assume, for example, that Table 7 has been formed in this manner. This table should be inspected for obvious errors—for example, an activity preceded or followed by itself, presence of an activity in the second or third column not listed in the first, etc. The next step is to generate the **F** column of Table 8 containing the precedence relations implicit but not explicitly given in Table 7. Thus, from Table 7: "C is preceded by A" so that the entry "A is followed by C" is made in Table 8; "E is preceded by B, C" gives rise to "B is followed by E" and "C is followed by E" in Table 8, etc. The **P** column of Table 8 is similarly formed from the antecedent relations (**F** column) of Table 7.

Before combining Tables 7 and 8 into a single table, we are in a position to check for certain errors and for redundancy. Error or redundancy is signaled if: for those activities labelled Case 1, each set of activities occurring in either the second or third column of Table 8 are not included in the set of activities appearing in the same position of Table 7; if for Case 2, sets in the second column of Table 8 are not included in the corresponding positions of Table 7; or if for Case 3, sets in the third column of Table 8 are not contained by the similar entries of Table 8. Inspection of present tables reveals, with respect to the entries in the second row and third column, that the set $\{E, I\}$ is not contained in the set $\{E, G\}$ as required, and further examination indicates that in

Table 7, the relation, "*I* is preceded by *B*," is redundant in view of the relations, "*B* is followed by *E*" and "*I* is preceded by *E*," also included in the table. After removing this redundant relation, Tables 7 and 8 are combined to obtain the desired result, Table 9. However, error and redundancy may still be present. In later sections, the location of cycles and the removal of any remaining redundant relations are considered.

ordering by levels

The process of recording the input data according to *levels* will now be explained. Consider the data contained in Table 10. The first step is to select those activities with no preceding activities, place them first and label them as *initial level* 0 (see Table 11, **IL** = 0). The two entries, *D* and *J*, are now deleted from the **P** column of Table 10 to obtain Table 12 which is to be examined for activities having no preceding activities to obtain the next level. Thus, in Table 12, *A*, *B*, *E*, *K* have no precedents and are labelled as initial level 1 and appended to Table 11 to form Table 13, and the residual Table 14 is constructed from Table 12 by deleting *A*, *B*, *E* and *K*. This process is continued until all activities are ordered by initial level resulting in Table 15.

A similar process is carried out to obtain the terminal levels—again starting with Table 10, but this time using the **F** entries—so that finally Table 16 may be constructed.

location of cycles

It is possible for the above process to fail, in that a point may be reached at which no further deletions can be made. In this case there are cycles in the input data, and the procedure can be used to locate all of them. A *cycle* is a series of precedence statements such as "*B* follows *A*," "*C* follows *B*," "*A* follows *C*."

The next example involves data which contains cycles, and illustrates how they are located. It should be noted that the ordering pass either accomplishes the ordering or locates and specifies *all* the cycles in the project description.

To illustrate the location of cycles, consider the data displayed in Table 17. We proceed exactly as above in attempting to order the data by level and obtain Tables 18 and 19 successively. Examination of Table 18 indicates the process cannot be continued because of the absence of activities without precedent

Table 15

IL	A	P	F
0	*D*	—	*A E K*
0	*J*	—	*B K*
1	*A*	*D*	*C*
1	*B*	*J*	*F H*
1	*E*	*D*	*F L*
1	*K*	*D J*	*I*
2	*F*	*B E*	*C H*
2	*L*	*E*	*C*
3	*C*	*A F L*	—
3	*H*	*B F*	*G I*
4	*G*	*H*	—
4	*I*	*H K*	—

Table 16

IL	TL	A	P	F
0	4	*D*	—	*A E K*
0	4	*J*	—	*B K*
1	1	*A*	*D*	*C*
1	3	*B*	*J*	*F H*
1	3	*E*	*D*	*F L*
1	1	*K*	*D J*	*I*
2	2	*F*	*B E*	*C H*
2	1	*L*	*E*	*C*
3	0	*C*	*A F L*	—
3	1	*H*	*B F*	*G I*
4	0	*G*	*H*	—
4	0	*I*	*H K*	—

Table 17

A	P
A	—
B	*F*
C	*E*
D	*B*
E	*D I J*
F	*G*
G	*A B H*
H	*B*
I	*C*
J	*B*

Table 18

A	P
B	*F*
C	*E*
D	*B*
E	*D I J*
F	*G*
G	*B H*
H	*B*
I	*C*
J	*B*

Table 19

A	P
X	*X H*
C	*E*
D	*X*
E	*D I J*
H	*X*
I	*C*
J	*X*

activities. Thus, Table 18 must contain cyclic data and further examination involving only the first entry of the **P** column reveals the cycle: B preceding F, F preceding G, and G preceding B. We replace each of activities B, F, and G in Table 18 by X, obtaining Table 19 where the **P** set for X contains the union of the **P** sets of B, F and G and naturally contains X. Again a cycle, X before H, and H before X, is present and by replacing H by X, Table 20 results. Since the **P** set of X includes only X, we may delete X from the latter table to obtain Table 21 from which Table 22 results by the deletion of D and J, and a second cycle— C before E, E before I, I before C— is evident and all of the cycles have been identified.

separation into subprojects

The next step is separation of the tables into the smallest subtables, such that each has no precedence relation with any other. Of course, it may be that no separation exists. Consider the data of Table 23 from which Table 24 has been derived by procedures already explained.

The process of completing Table 24 begins by writing an *f* (*forward scan*) in the first column of analysis opposite some activity of initial level 0, G here. The **F** set for G says that G is followed by A, B. These are marked with an *f*, and the *f* associated with G is starred, to indicate that it has been scanned. The next activity with an *f* in column 1 (A in this case) is now scanned, leading to an *f* for activity Q, and a star for the *f* associated with A. The process is iterated to termination. Certain activities of terminal level 0 are selected by this process, here C and X. These

Table 20

A	P
X	X
C	E
D	X
E	$D\ I\ J$
I	C
J	X

Table 21

A	P
C	E
D	—
E	$D\ I\ J$
I	C
J	—

Table 22

A	P
C	E
E	I
I	C

Table 23

A	P	N
A	$G\ P\ N$	Q
B	$F\ G$	$C\ K\ W$
C	$B\ Q$	—
D	J	—
E	Y	$L\ R$
F	P	$B\ Q$
G	—	$A\ B$
H	$L\ R$	—
I	U	$V\ M$
J	$T\ Y$	$D\ R$
K	B	X
L	E	H
M	$I\ S$	—
N	—	A
P	—	$A\ F$
Q	$A\ F$	$C\ W$
R	$E\ J$	H
S	U	M
T	—	J
U	—	$I\ S$
V	I	—
W	$B\ Q$	X
X	$K\ W$	—
Y	—	$E\ J$

Table 24

IL	TL	A	P	F	*Analysis* 1	2	3
0	4	G	—	$A\ B$	f^*	b^*	
0	4	N	—	A		b^*	f^*
0	4	P	—	$A\ F$		b^*	f^*
0	3	T	—	J			
0	2	U	—	$I\ S$			
0	3	Y	—	$E\ J$			
1	3	A	$G\ P\ N$	Q	f^*	b^*	
1	2	E	Y	$L\ R$			
1	3	F	P	$B\ Q$		b^*	f^*
1	1	I	U	$V\ M$			
1	2	J	$T\ Y$	$D\ R$			
1	1	S	U	M			
2	2	B	$F\ G$	$C\ K\ W$	f^*	b^*	
2	0	D	J	—			
2	1	L	E	H			
2	0	M	$I\ S$	—			
2	2	Q	$A\ F$	$C\ W$	f^*	b^*	
2	1	R	$E\ J$	H			
2	0	V	I	—			
3	0	C	$B\ Q$	—	f^*	b^*	
3	0	H	$L\ R$	—			
3	1	K	B	X	f^*	b^*	
3	1	W	$B\ Q$	X	f^*	b^*	
4	0	X	$K\ W$	—	f^*	b^*	

Table 25

IL	TL	A	P	F
0	4	*G*	—	*A B*
0	4	*N*	—	*A*
0	4	*P*	—	*A F*
1	3	*A*	*G N P*	*Q*
1	3	*F*	*P*	*B Q*
2	2	*B*	*F G*	*C K W*
2	2	*Q*	*A F*	*C W*
3	0	*C*	*B Q*	—
3	1	*K*	*B*	*X*
3	1	*W*	*B Q*	*X*
4	0	*X*	*K W*	—

Table 26

IL	A	F	*Analysis* 1	2	3	4
0	*A*	*C D G*				
0	*B*	*D E G*				1
1	*C*	*F G*				
1	*D*	*D*			1	1
1	*E*	*G H I*			1	
2	*F*	*I*		1		2
2	*G*	*I*	1	1	1, 2	1, 2
2	*H*	*I*	1			
3	*I*	—	1, 2			

are marked with *b* (*backward scan*) and the above process repeated "in reverse," leading to the second column of analysis in Table 24. Any elements of initial level 0 not already marked with f^*, are now marked for forward scan (here N, P), and the forward scan begun. Note that the activities which have already had a forward scan need not be scanned again, and likewise on subsequent backward scans. Here the only new activity subjected to a forward scan by N, P is F. But F is followed by B and Q which have already been scanned. Hence the process is complete, and the scanned set has no connection with the unscanned. Thus, we have $\{G, N, P, A, F, B, Q, C, K, W, X\}$ as a separate project (Table 25). Similarly it can be shown that $\{T, Y, E, J, D, L, R, H\}$ and $\{U, I, S, M\}$ are also separate projects.

elimination of redundancy

The process of eliminating redundancy will be illustrated by its application to the data contained in the first three columns of Table 26 giving rise successively to the last four columns of the same table. The process is started with the last activity having more than one element in its **F** set (E), by placing for each element in its **F** set, $\{G, H, I\}$, a *one* in the first column of the analysis. Then for each activity in the **F** set of an activity which now has a *one* in the first column, a *two* is placed in the same column (in this case, I is the only such activity). This column is now examined for the simultaneous occurrence of *ones* and *twos* and each such activity (only I in this case) is deleted from the **F** set of the originating activity (E). As the next step, moving upward in the activity column from E and passing D, since clearly an activity with only one member in its **F** set need not be considered, we repeat the procedure on C to obtain the entries in the second column. Similar steps applied to activities B and A give rise to the third and fourth columns, respectively. From these columns we see, respectively, that G may be deleted from the **F** sets of B and A. Finally, for each **F** set deletion a corresponding **P** set deletion is required, thus deletion of E from the **P** set of I, B from the **P** set of G, and A from the **P** set of G is necessary.

event network construction

We may now assume initial and terminal level tables free of inconsistencies and redundancy and proceed with the construction

Table 27

IL	A	P
0	*A*	—
0	*B*	—
0	*C*	—
1	*E*	*A*
1	*F*	*A*
1	*G*	*A B*
1	*R*	*B C*
1	*K*	*A B C*
1	*S*	*B C*
2	*L*	*G R S*
2	*M*	*F*
3	*N*	*F K L*

Table 28

TL	A	F
0	*E*	—
0	*M*	—
0	*N*	—
1	*F*	*M N*
1	*K*	*N*
1	*L*	*N*
2	*G*	*L*
2	*R*	*L*
2	*S*	*L*
3	*A*	*E F G K*
3	*B*	*G K R S*
3	*C*	*K R S*

Table 29

A	P
A B C	—
E F	*A*
R S	*B C*
G	*A B*
K	*A B C*
L	*G R S*
M	*F*
N	*F K L*

Table 30

A	F
E M N	—
F	*M N*
K L	*N*
G R S	*L*
A	*E F G K*
B	*G K R S*
C	*K R S*

of the event network. Consider the data contained in Tables 27 and 28. The first step is to group the activities in the latter tables, respectively, according to identical **P** and **F** sets, thus obtaining Tables 29 and 30.

It is particularly important to note that only elements belonging to the same level may have identical **P** sets or **F** sets, as the case may be, which reduces the search problem very substantially. Next it is necessary to locate sets of activities having the same **P** and **F** sets. Thus, A, B, C have the same **P** set, but the **F**'s are all different; E, F have the same **P** but different **F**'s; R, S, have the same **P** set and the same **F** set. For the time being the set $\{R, S\}$ will be replaced by a single activity H, having **P** set $\{B, C\}$ and **F** set $\{L\}$. No other sets of activities here have the same **P** set and the same **F** set. The above replacement generates the **P** and **F** columns in Tables 31 and 32, respectively.

We digress to note an interesting implication of a more general nature in the kind of property that has just been discussed. Suppose there is some subset of activities such that: at most one of the **P** sets consists only of activities not belonging to the subset, at most one of the **F** sets consists only of activities not belonging to the subset, and all other **P** and **F** sets for activities in the subset consist *only* of activities belonging to the subset. Such subsets can be determined automatically without much difficulty, and rather naturally define subprojects of the original project. They may, in fact, be treated as single activities, thus simplifying the task of event assignment. Of much more interest, however, is the consideration that there will, in general, be hierarchies of subprojects. Thus, the prospect exists of describing projects by nested subprojects, in a series of networks proceeding from least detail to fine detail.

Resuming the computation, for each non-null **P** set an associated set, denoted as a **U** set, is formed from the intersection of all **F** sets belonging to an activity in the **P** set. For example, G has the **P** set $\{A, B\}$. The **F** sets are: $\{E, F, G, K\}$ for A, $\{G, K, H\}$ for B. The intersection of $\{E, F, G, K\}$ and $\{G, K, H\}$ is $\{G, K\}$, which is thus the **U** set for G. If the **P** set is empty then the **U** set is the set of all activities. This yields the **U** column of Table 31.

The next step is to determine which **U**'s are identical with **F** sets. Using the row numbers designated in the **R** column as subscripts, $\mathbf{U}_2 = \mathbf{F}_5$, $\mathbf{U}_3 = \mathbf{F}_7$, $\mathbf{U}_6 = \mathbf{F}_4$, $\mathbf{U}_7 = \mathbf{F}_2$, $\mathbf{U}_8 = \mathbf{F}_3$. The magnitude of the search is also reduced by the general principle that a **U** set is identical with at most one **F** set of an activity belonging to the **P** set with which **U** is associated. Thus $\mathbf{U}_2$ can at most be identical with the **F** set for A, that is, $\mathbf{F}_5$. $\mathbf{U}_3$ can at most be identical with one of the **F** sets for B or C, etc.

The assignment of events can now take place in Tables 31 and 32. Since $\mathbf{U}_2 = \mathbf{F}_5$, the event $c_{2,5}$ is assigned adjacent to $\mathbf{U}_2$ and to $\mathbf{F}_5$ in the initial and terminal event (**IE**, **TE**) columns of Tables 31 and 32.

Table 31

R	A	P	U	IE
1	$A\ B\ C$	—	*Set*	a_1
2	$E\ F$	A	$E\ F\ G\ K$	$c_{2,5}$
3	H	$B\ C$	$K\ H$	$c_{3,7}$
4	G	$A\ B$	$G\ K$	a_4
5	K	$A\ B\ C$	K	a_5
6	L	$G\ H$	L	$c_{6,4}$
7	M	F	$M\ N$	$c_{7,2}$
8	N	$F\ K\ L$	N	$c_{8,3}$

Table 32

R	A	F	TE
1	$E\ M\ N$	—	b_1
2	F	$M\ N$	$c_{7,2}$
3	$K\ L$	N	$c_{8,3}$
4	$G\ H$	L	$c_{6,4}$
5	A	$E\ F\ G\ K$	$c_{2,5}$
6	B	$G\ H\ K$	b_6
7	C	$K\ H$	$c_{3,7}$

Similarly $\mathbf{U}_3 = \mathbf{F}_7$ leads to event $c_{3,7}$ in both tables, etc. When this process is completed, unfilled event locations are filled with distinct notations: subscripted a's and b's in Tables 31 and 32, respectively, with value of the subscript indicating the row. The result is that activities A, B and C have initial event a_1; E and F have $c_{2,5}$, etc.; E, M and N have terminal event b_1, F has terminal event $c_{7,2}$, etc. All activities have now been assigned initial and terminal events and it is now time to assign dummy activities to maintain precedence relations. Now A, B, and C have initial event a_1, and no preceding activities, hence no dummies are required. E, F have initial event $c_{2,5}$, and are preceded by A, which has terminal event $c_{2,5}$. Again no dummy is required. H has initial event $c_{3,7}$, C has terminal event $c_{3,7}$, but B has terminal event b_6. Thus a dummy activity with b_6 and $c_{3,7}$ as initial and terminal events is required. $(b_6, c_{3,7})$ will be used as notation for this dummy activity. Continuing in this manner the list of dummy activities is: $(b_6, c_{3,7})$, (b_6, a_4), $(c_{2,5}, a_4)$, $(c_{2,5}, a_5)$, (b_6, a_5), $(c_{3,7}, a_5)$, $(c_{7,2}, c_{8,3})$. All necessary dummies are included in this list, but there may be unnecessary ones. To check this point, the events involved in the notation for these dummies are treated as activities, the pairings indicated by the notation are treated as precedence relations, that is b_6 precedes $c_{3,7}$, etc., and Table 33 results.

Table 33

A	P	F
b_6	—	$c_{3,7}\ a_4\ a_5$
$c_{3,7}$	b_6	a_5
a_4	$b_6\ c_{2,5}$	—
$c_{2,5}$	—	$a_4\ a_5$
a_5	$c_{2,5}\ b_6\ c_{3,7}$	—
$c_{7,2}$	—	$c_{8,3}$
$c_{8,3}$	$c_{7,2}$	—

When this table is reordered (there will be no cycles), separated, and redundancies located as before, it will be found that b_6 preceding a_5 is the only redundant precedence, hence the dummy activity (b_6, a_5) may be deleted. It should be noted that once the ordering, separation, and redundancy routines have been written, this is only a question of using them once more. The event network, then, is specified by Table 34.

Table 34

A	IE	TE
A	a_1	$c_{2,5}$
B	a_1	b_6
C	a_1	$c_{3,7}$
E	$c_{2,5}$	b_1
F	$c_{2,5}$	$c_{7,2}$
G	a_4	$c_{6,4}$
H	$c_{3,7}$	$c_{6,4}$
K	a_5	$c_{8,3}$
L	$c_{6,4}$	$c_{8,3}$
M	$c_{7,2}$	b_1
N	$c_{8,3}$	b_1
D_1	b_6	$c_{3,7}$
D_2	b_6	a_4
D_3	$c_{2,5}$	a_4
D_4	$c_{2,5}$	a_5
D_5	$c_{3,7}$	a_5
D_6	$c_{7,2}$	$c_{8,3}$

This event network can be shown to have the minimum number of vertices and dummies, and to have unique structure. Two points remain to be discussed. First, H represents a pair of activities, R and S. Depending on the subsequent use of the network, it may suffice to leave the pair as the single activity H, or it may not. If not, one further dummy must be introduced to maintain the difference. Thus, we may, for example, replace $H \mid (c_{3,7}, c_{6,4})$ by

$R \mid (c_{3,7}, c_{6,4})$

$S \mid (e, c_{6,4})$ and add a dummy $(c_{3,7}, e)$.

In general, this treatment introduces $n - 1$ new dummies and events whenever there are n parallel activities.

The second point has to do with the identification which has been made of the initial events of the starting activities A, B, C, and the terminal events of the terminal activities E, M, N. This identification is made for convenience of processing, and can now be separated in any fashion desired. The diagram for the event network is given in Figure 3 (with H not replaced).

programming considerations

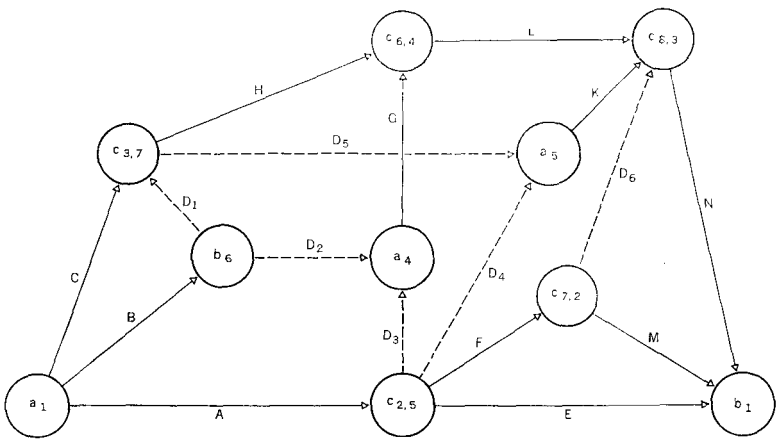
It is assumed in the following that the project activity network can be contained entirely in core storage, which will certainly be true for a 32K IBM® 7090 and for 4000 activities (based on an average of three following activities—which is probably higher than normal), provided efficient names are used. This in general calls for the replacement of input activity names with names for internal use, and thus for establishment of a dictionary for translation between the two names. It is also assumed that the entire dictionary can be contained in core, which again is true for 4000 activities, if not more than six characters are required for its input name. The discussion below is concerned with the major aspects of the programming system contemplated.

sort and dictionary formation

Suppose that the original input is on magnetic tape, and that each record on tape contains three things: activity name, name of activities in the preceding set, and names of activities in the following set. There will also need to be some sort of marker to distinguish the set of preceding activities from the set of following activities.

The first operation to be performed is a sort of these records according to activity name. The second operation is the establishment of the dictionary which relates input activity name to a convenient code name. The code name is established as follows. Let some core memory cell be assigned as initial storage cell for the data presently on tape. Read in the first record, assign the

Figure 3 The event network



initial storage cell address as the code name for its activity. The code name for the activity name of the second record will be the same address plus a number sufficiently large so that enough storage is left for the associated activities of the first record. That is to say, the coded activity name will be the address of the first cell in memory associated with its record; the record itself will consist of coded names of preceding and following activities. For example, suppose the first two records are as shown in Table 35. Suppose that cell 1000 is assigned as starting location. Then the dictionary has the form given in Table 36, assuming say, that a half word is assigned for activity name. Core storage ultimately will have the following appearance indicated in Table 37 where $C\ (A_3)$ represents the coded name of A_3, etc. The fact that $C\ (A_6)$ represents the start of a new record could be indicated by using a minus sign on that word. M of course is the marker to separate sets. A half word might be left free at the beginning of each record, for storage of temporary data in processing these records.

Table 35

A	P	F
A_1	$A_3\ A_4\ A_6\ A_8$	$A_{10}\ A_{12}$
A_2	A_6	$A_5\ A_8\ A_{14}$

Table 36

A	*Cell*
A_1	1000
A_2	1004
A_3	1007

Table 37

Cell	*Content*
1000	$C\ (A_3)\ C\ (A_4)$
1001	$C\ (A_6)\ C\ (A_8)$
1002	$M\ C\ (A_{10})$
1003	$C\ (A_{12})$ —
1004	$C\ (A_6)\ M$

coding of project tables

Once the dictionary is established, the input tape can be passed, a record at a time, through core and coded records produced and stored on an auxiliary tape. Any activity name which appears can be located quite rapidly in the dictionary by successive halving, inasmuch as the dictionary has been sorted by input activity name. Of course, if some activity name is not to be found in the dictionary then there is an error, and the fact should be reported by the printer. It is also easy, at this time, to determine whether an activity precedes or follows itself.

The next operation is a sort of the dictionary, the key being the coded name. The set of coded records can then be brought into core, ready for further processing.[7]

reordering

Since now the name of an activity is also the location of its record in core, the reordering processing previously defined requires no search whatever. That is to say, if an activity A is preceded by no activity, it is to be eliminated from all other preceding sets. But the following set for A gives the location of all records containing A in the preceding set. Moreover, if a count has been established of the number of preceding activities and of the number of following activities in each record, then this number can be updated for every deletion, thus locating as part of the process null sets which have been produced by deletion.

For separation into subtables and elimination of redundancy, it is necessary only to redefine the dictionary in terms of the new record locations, which will be those determined by initial level order. Also, computing will be much expedited if the **P** and **F** sets are also ordered by initial level.

computation of event network

Once this is done, separation into subtables and elimination of redundancy proceed very efficiently, and thus the data is ready for event network construction. The first step requires, on the one hand, a grouping of activities that have common **P** sets, and on the other hand, a grouping of activities that have common **F** sets. It has been observed already that groups of the first kind can only

Table 38

A	P	F	A	P	F
A	p_1	f_5	G	p_4	f_4
B	p_1	f_6	K	p_5	f_3
C	p_1	f_7	L	p_6	f_3
E	p_2	f_1	M	p_7	f_1
F	p_2	f_2	N	p_8	f_1
H	p_3	f_4			

Table 39

Core	*Rec*	*Core*	*Rec*
p_1	—	f_1	—
p_2	A	f_2	$M\ N$
p_3	$B\ C$	f_3	N
p_4	$A\ B$	f_4	L
p_5	$A\ B\ C$	f_5	$E\ F\ G\ K$
p_6	$G\ H$	f_6	$G\ K\ H$
p_7	F	f_7	$K\ H$
p_8	$F\ K\ L$	—	—

contain certain activities having the same initial level, groups of the second kind can only contain activities having the same terminal level. The first set of groups can be obtained from the project table already at hand. The location of groups of the second kind requires only that part of the reordering process which separates out activities of common terminal level, together with their **F** sets. An expedient way to store all this information for the example previously given in the section on event network construction (see Tables 29, 30) is illustrated in Tables 38 and 39, where p_i, f_i represent core location of the pertinent records. It is quite clear how the remainder of this processing is to be performed.

No discussion of the case of event network reduction, or the treatment of mixed input is undertaken here. These would appear to be readily managed by variations of methods discussed above. In passing, it should be observed that the ordering process, if applied to the nodes of the event network, will achieve the result otherwise known as "node numbering."

CITED REFERENCES AND FOOTNOTES

1. *PERT Summary Report Phase I*, Special Projects Office, Bureau of Naval Weapons, Washington, D.C. (July 1958).
2. *Project PERT, Phase II*, Special Projects Office, Bureau of Naval Weapons, Washington, D. C. (Nov. 1958).
3. Fey, C. F., *Application of Least Cost Estimating and Scheduling*, Management Science Report MS-1, IBM, Bethesda, Maryland (1962).
4. Fulkerson, D. R., *A Network Flow Computation for Project Cost Curves*, Journal of the Institute for Management Science, 7 (1961), pp. 167–178.
5. Kelley, J. E., Jr., *Critical Path Planning and Scheduling: Mathematical Basis*, Journal of Ops. Res. Soc. Am. 9 (1961) pp. 296–320.
6. Dimsdale, B., *On Project Networks*, Western Data Processing Center, UCLA (1962).
7. If the original data is in some other form, it is clear that appropriate modifications of these procedures can be made.

BIBLIOGRAPHY

- Berge, *Theorie des Graphes et ses Applications*, Paris, (1958).
- Harary, F., *On the Consistency of Precedence Matrices*, Journal Assn. for Comp. Mach., 7 (1960), pp. 255–259.
- Jornagin, M. P., *Automatic Machine Methods of Testing Pert Networks for Consistency*, U. S. Naval Weapons Lab T. M. No. K-24-60.
- Ore, O., *Theory of Graphs*, Am. Math. Soc. Coll. Pub. 38 (1962).
- Prostick, Joel M., *Loop Tracing in PEP-PERT Networks*. Paper presented at sixteenth National Conference of the Assn. Comp. Mach., Los Angeles, Calif., (1961).