CRITICAL-PATH PLANNING AND SCHEDULING: MATHEMATICAL BASIS*

James E. Kelley, Jr.

Mauchly Associates, Inc., Ambler, Pennsylvania

(Received June 1, 1960)

This paper is concerned with establishing the mathematical basis of the Critical-Path Method—a new tool for planning, scheduling, and coordinating complex engineering-type projects. The essential ingredient of the technique is a mathematical model that incorporates sequence information, durations, and costs for each component of the project. It is a special parametric linear program that, via the primal-dual algorithm, may be solved efficiently by network flow methods. Analysis of the solutions of the model enables operating personnel to answer questions concerning labor needs, budget requirements, procurement and design limitations, the effects of delays, and communication difficulties.

DURING the past few years there has developed a growing interest in the problems of managing large projects. The literature on the subject ranges from statements of problem areas through case histories to mathematical and statistical analyses of some sophistication. Most of these latter analyses are concerned primarily with discovering and exploring structural characteristics of projects with the objective of making plans, schedules, and forecasts of various types.

While it is generally realized that the fundamental characteristic of all projects is that all the activities involved must be performed in some well-defined order, it appears that little has been done to make *explicit* use of this fact. Recently, however, two parallel efforts, which take their origins in the series-parallel relations among project activities, have been under way. One of these efforts is reported in reference 12 and is called the PERT system. It is concerned primarily with monitoring progress on R and D projects. The other effort, called the Critical-Path Method, is reported in references 1, 3, 6, 7, 8, 10, 11, 13, and 14. It is concerned with the planning, scheduling, and cost-control aspects of project work.

In the present paper we will be concerned with only the mathematical basis of the Critical-Path Method. The historical development of the method and applications, as well as a comparison with PERT, is contained in reference 11.

* Part of this work was done while the author was with Remington Rand UNIVAC.

The mathematical model upon which the Critical-Path Method is based is a parametric linear program that has the objective of computing the utility of a project as a function of its duration. For each feasible project duration, a feasible project schedule is obtained that has maximum utility among all feasible schedules of the same project duration.

The parametric linear program involved has special structural properties that allow it to be solved efficiently by special techniques. Three methods have been developed. The first is based on the work of GASS AND SAATY.[5] Considerable simplification of their method is made possible by the fact that the constraint matrix involved here is similar to and has essentially the same properties as the Hitchcock-Koopmans transportation problem: all matrix entries are 0 or ± 1 and every basis matrix is triangular. This method, which has only historical interest at present, is reported in references 7 and 8, and will not be considered further here.

However, as a result of the discovery of the relation between parametric programming and the primal-dual algorithm,[9] an even more efficient algorithm has been developed. Fundamental to this algorithm is the solution of a maximum network flow problem with both positive upper and lower bound capacity restrictions. A slight variation of the Ford-Fulkerson flow algorithm[2] is all that is required to solve this flow problem.

The third method, developed by FULKERSON,[3] also approaches the problem via network flow theory. However, his approach to the problem, and the network flow algorithm he develops, differs enough from ours to warrant separate publications. The differences in approach to the problem should be of philosophical interest to those concerned with constructing special linear-programming algorithms. Further, which of our algorithms is better computationally is essentially unknown at present. Indeed, it is not unlikely that a hybrid version of the two methods might turn out to be more advantageous than either technique alone.

Significant results have been obtained by applying the Critical-Path Method to the design, procurement, and fabrication functions of a variety of project types. The following list of applications is merely suggestive:

1. All types of construction and maintenance.
2. Retooling programs for high-volume production.
3. Low-volume production scheduling.
4. Scientific missile countdown procedures.
5. Budget planning.
6. Mobilization, strategic and tactical planning.
7. New product launching.
8. Assembly and testing of electronic systems.
9. Installation, programming, and debugging of computer systems.

FORMULATION OF THE PROBLEM

LET E be a finite partially ordered set of $n+1$ elements called *events*. There are two distinguished events in E, *origin* and *terminus*, respectively, with the property that origin precedes and terminus follows every event in E.

Each event is denoted by a nonnegative integer, its label. Since E is partially ordered, we may assume that the events are labeled such that if event i precedes event j then $i<j$. In particular, origin is given the label 0 and terminus is given the label n.

Also associated with event i is a nonnegative number, t_i, which represents the time at which the event occurs. Thus, if event i precedes event j then $t_i \leqq t_j$. We will always let $t_0=0$.

An *activity* is an element, (i,j), of $E\times E$, such that $i<j$. Associated with each activity is a nonnegative number, y_{ij}, its *duration*. It is assumed that activity (i,j) must be performed sometime between the occurrences of event i and event j. Thus we must have

$$y_{ij}+t_i-t_j \leqq 0. \tag{1}$$

A *project*, P, is a set of events and activities with the property that if event k is in P then k is either origin or terminus, or else there exist events i and j in P such that activities (i,k) and (k,j) are both in P.

An assignment of durations, y_{ij}, to activities and occurrence times, t_i, to events in P is called a *schedule*. A schedule will be denoted by $\{\boldsymbol{y},\boldsymbol{t}\}$, where $\boldsymbol{y}$ and $\boldsymbol{t}$ are vectors whose coordinates are the y_{ij} and t_i, respectively, which define the schedule. If there are m activities in P, $\{\boldsymbol{y},\boldsymbol{t}\}$ may be interpreted as a vector in an $(m+n+1)$-dimensional Euclidean space.

Sometimes the duration of an activity is a matter of management decision subject to certain restrictions. The simplest restrictions, and the only ones with which we will deal, are that y_{ij} be bounded above and below for each activity in P. That is, there are numbers d_{ij} and D_{ij} such that

$$0 \leqq d_{ij} \leqq y_{ij} \leqq D_{ij} < \infty \tag{2}$$

for all (i,j) in P. We will call D_{ij} the *normal* duration of activity (i,j); d_{ij} will be called the expedited or *crash* duration.

REMARK 1: The value of d_{ij} is an approximation to the fastest time in which an activity can be performed and is determined by the nature of the activity and the environment in which it must be performed. On the other hand, D_{ij} must *usually* be established by fiat. It represents a 'reasonable' performance time under 'normal' circumstances.

A schedule satisfying (1) and (2) with $t_0=0$ is called a *feasible schedule*.

The duration actually selected for each activity when forming a feasible schedule is made to depend upon its *utility*. For the moment we will as-

sume that the utility of an activity is a linear function of its duration on the closed interval defined by (2) and has the form: $a_{ij}y_{ij}+b_{ij}$, where $0 \leqq a_{ij} < \infty$ and $-\infty < b_{ij} < \infty$.

The *utility* of a schedule is defined as the sum of the utilities of the individual activities in P, viz.:

$$\sum_{(i,j)\in P} (a_{ij}y_{ij}+b_{ij}). \tag{3}$$

The *duration of a schedule* is $\lambda = t_n$.

It is clear that among all feasible schedules having a given duration, λ, there is at least one which has maximum utility, i.e., maximizes (3). Such a feasible schedule will be called *optimal*. We denote this value of (3) for this schedule by $U(\lambda)$.*

Considered as a function of λ, $U(\lambda)$ will be called the *project utility function*.

Our main objective is to *find an algorithm for generating* $U(\lambda)$ *and optimum feasible schedules that define it.* Some uses for $U(\lambda)$ will be considered in a later section.

REMARK 2: There is some justification outside of the limitations of the mathematical techniques we employ for assuming that activity utility functions are nondecreasing and linear. Assume that the utility of activity (i,j) is greatest at d_{ij}. Then in any optimal feasible schedule we would always have $y_{ij}=d_{ij}$. Thus, the 'effective' utility of (i,j) on $[d_{ij}, D_{ij}]$ is a constant equal to the utility at d_{ij}. For general utility functions the same remark holds on any subinterval $[p,q]$ of $[d_{ij}, D_{ij}]$ where the utility at p is greater than the utility at any other point in $[p,q]$. Thus, the 'effective' utility of (i,j) on $[d_{ij}, D_{ij}]$ is always nondecreasing. Further, it is often very difficult in practice to obtain estimates of an activity's utility for more than just a few durations. In such circumstances, taking the trend is a reasonable thing to do. However, as we will see in a later section, the linearity assumption may be replaced by the assumption that an activity's utility function is piece-wise linear, nondecreasing, and concave between its crash and normal durations.

PRACTICAL RULES FOR DESCRIBING A PROJECT

THE PROCESS of describing the order relations among the activities of a project is facilitated by the use of a graphical technique. Each activity in the project is denoted by an arrow that depicts the activity's existence and the direction of time-flow (time flows from the tail to the head of an arrow). The arrows are then interconnected to show the sequence relations among

* Whenever the measure of utility is cost, loss, etc., which require minimization, we simply take the negative of the function and then maximize.

the activities. The result is a directed graph with no directed circuits. The nodes of the graph correspond to the events of the project. Figure 5 typifies a project graph.

Note that it is not usual to represent the elements (project activities here) of a partially ordered set by the edges of a directed graph. More commonly, one represents them by the nodes, as with a lattice diagram. However, it is clear that each node of a lattice diagram can be replaced by a directed edge or arrow in such a way that all edges directed to a node are directed to the tail of the arrow replacing it and all edges directed from a node are directed from the head of the arrow. In this way all elements of the partially ordered set may be represented by directed edges.

If this approach were to be adopted, the resulting project graph would be replete with fictitious or dummy activities—anywhere from n to $\frac{1}{2}n(n-1)$ dummies in a project of n activities. Of course, many of these dummies are not necessary and could be eliminated.*

A more practical approach, requiring the introduction of far fewer dummies, may be obtained by starting directly with the 'arrow' representation of an activity. As the arrows are connected to form the project graph, dummies are introduced, as needed, to preserve the order relations among the activities. The following rules cover these situations and others of practical interest:†

Rule 1: Composite Activities. Let activity A be a predecessor of activity B. In practice, A is often interpreted in such a way that B can be started as soon as A is partially completed. Or, more generally, many activities may be started as soon as A is some percentage toward completion. In this situation we consider A to be a *composite* of many activities. For example, if activity C can be started when A is half completed, activity D when A is three-quarters completed, and activity B when A is fully completed, we consider A to be a composite of three different activities: A_1, A_2, and A_3. The example is illustrated in Fig. 1.

By making use of this device of redefining activities in terms of their components when the occasion demands it, we may always assume that each job in a project is fully completed before any of its successors can

* An interesting combinatorial problem is suggested by the desire to eliminate unnecessary dummies, although it is not of much practical interest in the present context. Given a directed graph with no directed circuits in which certain edges are distinguished, the remaining edges being undistinguished. A distinguished edge may be contracted to a point only if the order relations among undistinguished edges are preserved thereby. Problem: What is the maximum number of distinguished edges that can be contracted to points?

† It has been observed that by using these rules on 'real' projects, the resulting number of activities (including dummies) averages 1.7 the number of events.

begin. Of course, a separate analysis of the utility function of each component must be made.

Rule 2: Concurrent Activities. It can happen that two or more activities must begin and end at the same events in a project. This situation is ambiguous, since all these activities would have the *same* designation numbers. In order to eliminate this ambiguity we consider all but one of these activities to be a composite of two activities: one is the activity itself while the other is a fictitious activity or *dummy*. This situation is illustrated in Fig. 2. Activities B and C must be completed before E can start. Dummy activity X is introduced in order to distinguish B and C from one another but still maintain the required sequencing of the activities. A utility of zero and normal and crash durations of zero are assigned to X.

Rule 3: Aggregated Activities. Sometimes a certain group of activities can be considered as one activity. This interpretation may be quite de-

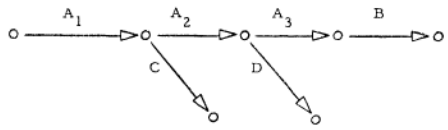

Figure 1

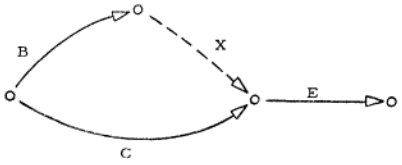

Figure 2

sirable, especially when all the activities in the group are technologically ordered and can be considered to form a small project in their own right. When this type of situation occurs a dummy may be substituted for the whole group of activities. Indeed, by this type of aggregation, major projects may be simplified for certain purposes. Figure 3 illustrates this case. The dummy X is substituted for activities B, C, D, E, and F. Of course, a special analysis of the utility function and durations limits for X must be made (*see* Remark 5 in the section "A Non-Linear Extension").

Rule 4: Dependent and Independent Activities. It can often occur that a certain activity, C, is a successor of two concurrent activities, A and B, but B may have a successor, D, that is *not* also a successor of A. This situation may be handled by introducing a dummy, X, as indicated in Fig. 4. The interpretation is clear. A utility of zero and normal and crash durations of zero are assigned to X.

Rule 5: Lead-Time Activity. Eventually a project must be put on the calendar. It is thus necessary to introduce at least one activity that starts at the zero date on the calendar and terminates at the start of the project.

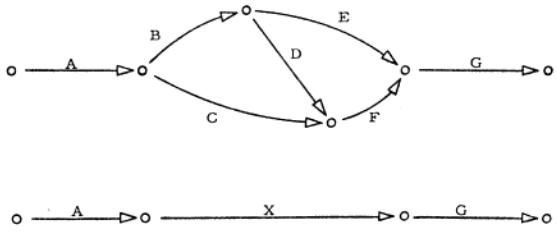

Figure 3

This activity is interpreted as the *lead time* for the entire project. It may be used to aggregate certain administrative matters that must be taken care of before the actual start of the project.

Usually the normal duration of lead time is the time from some fixed calendar date to the assumed project start date. The crash duration is zero. Other duration limits are possible, depending on the interpretation of lead time.

The utility function of lead time is usually given a zero slope. However, when it is desirable to delay the start of the project as much as possible, a large positive slope is sometimes assigned to the utility function.

Rule 6: Start Conditions. The initiation of some activities in a project may depend on the delivery of certain items or on proper conditions—materials, plans, authorization of funds, weather, etc. When all activities emanating from event i depend on the same start condition, a constraint of the form $t_i \geqq T$ must be included in the model of the project. The time, relative to the start of the project, at which these activities may start is denoted by T.

The addition of start conditions to the model does not alter its form essentially. We may reduce the modified problem to the standard form of (1) and (2) by introducing the dummy activity $(0,i)$ such that $D_{0i} = d_{0i} = T$ and $a_{0i} = b_{0i} = 0$. It is easy to see that with these conditions any feasible schedule must have $t_i \geqq T$.

When not all activities emanating from event i are dependent on the start condition, Rule 4 above is applied in the obvious way to ensure that they will be independent of the start condition.

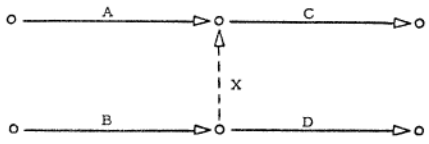

Figure 4

If several deliveries are to be made to event i from origin, Rule 2 must be applied to eliminate the ambiguity of them all having the same designation numbers.

AN EXAMPLE OF A PROJECT

THE FOLLOWING is a typical situation encountered in a chemical processing plant:

Imagine a reactor and storage tank interconnected by a 3-inch insulated process line. Because of process conditions, the line needs periodic replacement. Interspersed along the line and at the terminals are valves. All valves need replacing as well. No pipe or valves are in stock. Accurate 'as built' drawings exist and are readily available. The line is overhead so scaffolding is required for the replacement operation. Adequate craft labor is available. It is assumed that in order to minimize down time on the production unit, work on the project will proceed on an 'around-the-clock' basis.

The Works Engineer has requested the Maintenance and Construction Superintendent, who is responsible for renewing the pipeline, to prepare a plan and schedule for review with the Operating Departments. The plant Methods and Standards Section has furnished the data of Table I for the various activities involved in the project.

TABLE I

Activity	Description	Normal		Crash	
		Duration (hours)	Cost ($)	Duration (hours)	Cost ($)
A	Develop material list	8	100	8	100
B	Deactivate old line	8	150	8	150
C	Erect scaffold	12	300	8	450
D	Remove scaffold	4	100	2	170
E	Procure pipe	200	850	130	1100
F	Prefab pipe sections	40	1200	25	2000
G	Place new pipe	32	800	12	1900
H	Weld pipe	8	100	4	300
I	Fit-up pipe and valves	8	100	4	250
J	Procure valves	225	300	140	600
K	Place valves	8	100	4	250
L	Remove old pipe and valves	35	400	18	1000
M	Insulate pipe	24	300	12	700
N	Pressure test	6	50	3	100
P	Clean-up and start-up	4	100	2	200

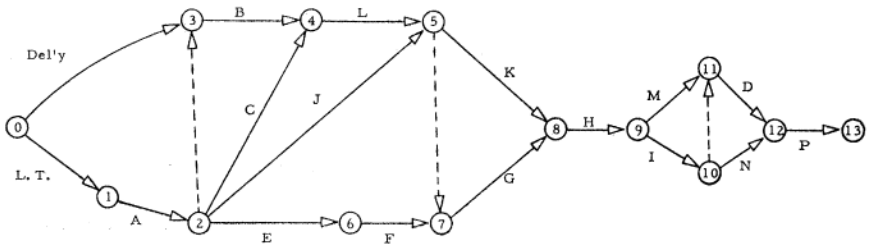

Figure 5

The sequencing relations among the activities of the pipeline renewal project are shown in the graph of Fig. 5. For example, the arrow labeled B is activity (3,4), 'deactivate old line.'

Activity (0,1) labeled L.T., is lead time for the project. Activity (0,3) represents the delivery of the line from the Operating Department to the Maintenance Department.

Note that in Fig. 5 there are certain activities represented by broken-line arrows. These are 'dummy' activities that only signify the sequencing of certain other activities. Thus dummy activity (10, 11) implies that removing the scaffold not only depends on the completion of the insulating of pipe but also on the fitting-up of pipes and valves.

It is important to note that the project might have been set up differently, depending on the desires of management. For example, instead of prefabricating *all* the pipe sections before placing them, both activities might be done together. Similarly, the welding of the pipe might also be done together with the placing of new pipe. Indeed, this latter situation may be required by the nature of the pipe fitting.

There may be other possibilities for combining activities or rearranging their order of execution, each of which gives the project a different aspect. An important feature of a project graph is that it allows one to see at a glance all the interrelations among the activities. In practice, a project graph often suggests how activities may be arranged to better advantage than originally conceived.

THE PRIMAL-DUAL ALGORITHM

We may view the problem of maximizing (3) subject to (1) and (2) with $t_0=0$ and $t_n=\lambda$ as a parametric linear program with parameter λ. We propose to use the primal-dual algorithm[9] to solve it and proceed as follows:

Let $\{\boldsymbol{y},\boldsymbol{t}\}$ be an optimal feasible schedule of duration λ and define the following sets of activities:

$$Q_1=\{(i,j)|y_{ij}+t_i-t_j=0,\ (i,j)\in P\},$$
$$Q_2=\{(i,j)|y_{ij}=D_{ij}>d_{ij},\ (i,j)\in P\},$$
$$Q_3=\{(i,j)|y_{ij}=D_{ij}=d_{ij},\ (i,j)\in P\},$$
$$Q_4=\{(i,j)|y_{ij}=d_{ij}<D_{ij},\ (i,j)\in P\}.$$

The salient features of the primal-dual algorithm, when specialized to the present case, may be summarized in the following

THEOREM 1: *Let* $\{\boldsymbol{y},\boldsymbol{t}\}$ *be an optimal feasible schedule of duration* λ. *If* σ_{ij}, $(i,j)\in P$, *and* δ_i, $0\leqq i\leqq n$, *minimize the linear form*

$$\sum_{(i,j)\in P} a_{ij}\,\sigma_{ij} \tag{4}$$

subject to

$$p_{ij}=\sigma_{ij}+\delta_i-\delta_j\geqq 0,\quad (i,j)\in Q_1$$

$$\sigma_{ij}\begin{cases}\geqq 0, & (i,j)\in Q_1\cap Q_2\\ =0, & (i,j)\in P-(Q_1-Q_3)\\ \leqq 0, & (i,j)\in Q_1\cap Q_4\end{cases} \tag{5}$$

$$\delta_0=0\quad \textit{and}\quad \delta_n=1,$$

then the schedule $\{\boldsymbol{y}',\boldsymbol{t}'\}$ *defined by*

$$y'_{ij}=y_{ij}-\theta\sigma_{ij},\quad (i,j)\in P$$
$$t'_i=t_i-\theta\delta_i,\quad 0\leqq i\leqq n$$

is an optimal feasible schedule of duration $\lambda'=\lambda-\theta$, *where* $0\leqq\theta\leqq\theta_0>0$ *and*

$$\theta_0=\min[\alpha,\beta,\gamma]$$

$$\alpha=\begin{cases}\min_{p_{ij}<0}[(y_{ij}+t_i-t_j)/p_{ij}]\\ +\infty, \quad \textit{if } p_{ij}\geqq 0 \textit{ for all } (i,j)\in P\end{cases}$$

$$\beta=\begin{cases}\min_{\sigma_{ij}<0}[(y_{ij}-D_{ij})/\sigma_{ij}]\\ +\infty, \quad \textit{if } \sigma_{ij}\geqq 0 \textit{ for all } (i,j)\in P\end{cases}$$

$$\gamma=\begin{cases}\min_{\sigma_{ij}>0}[(y_{ij}-d_{ij})/\sigma_{ij}]\\ +\infty, \quad \textit{if } \sigma_{ij}\leqq 0 \textit{ for all } (i,j)\in P.\end{cases}$$

However, if (5) *is inconsistent then there are no feasible schedules of duration less than* λ.

This theorem follows, *mutatis mutandis*, from the corollary and Theorem II of reference 9.

The primal-dual algorithm now consists in finding an optimal feasible schedule $\{\boldsymbol{y},\boldsymbol{t}\}$ of duration λ and then solving (4) and (5) to determine $\{\boldsymbol{y}',\boldsymbol{t}'\}$ of duration $\lambda-\theta_0$. $\{\boldsymbol{y}',\boldsymbol{t}'\}$ *is called a characteristic schedule.* (The problem of minimizing (4) subject to (5) is called the *restricted dual*

problem.) Using this new optimal feasible schedule the process is repeated until no feasible schedules of shorter duration can be found. At this point the algorithm terminates.

Clearly, the above process generates a continuum of optimal feasible schedules and $U(\lambda)$ between the starting and terminating values of λ. However, the questions of how to find an initial optimal feasible schedule and how to generate optimal feasible schedules of greater duration than that of the initial one as well as how to solve (4) and (5) are as yet unanswered. This we will do in the next three sections.

PROPERTIES OF THE PROJECT UTILITY FUNCTION

THEOREM 2: *The feasible schedule,* $\{\mathbf{y},\mathbf{t}\}$, *defined by*

$$y_{ij}=D_{ij}, \quad (i,j)\in P$$
$$t_0=0,$$
$$t_j=\max_{(i,j)\in P}(y_{ij}+t_i), \quad 1\leqq j<n,$$

is an optimal feasible schedule of duration $\lambda\ (=t_n)$ *whenever*

$$\lambda\geqq\max_{(i,n)\in P}(D_{in}+t_i)=M.$$

Proof. That the schedule is feasible is obvious. That it is optimal follows from the fact that $a_{ij}\geqq 0$ for all $(i,j)\in P$.

As a corollary to Theorem 2 we obtain the obvious

THEOREM 3: *For all* $\lambda\geqq M$

$$U(\lambda)=\sum_{(i,j)\in P}(a_{ij}D_{ij}+b_{ij}).$$

In consequence of Theorem 2 and Theorem 3 we may take the optimal feasible schedule of Theorem 2 with $\lambda=M$ as the initial schedule required for the primal-dual algorithm.

THEOREM 4: *Consider the feasible schedule defined by*

$$y_{ij}=d_{ij}, \quad (i,j)\in P,$$
$$t_0=0,$$
$$t_j=\max_{(i,j)\in P}(y_{ij}+t_i), \quad 1\leqq j\leqq n,$$

and let $m=t_n$. *Then* $U(\lambda)$ *is not defined for any* $\lambda<m$.

Proof. Obvious.

For the remaining properties of interest it is expedient to use a theorem slightly more general than required.

THEOREM 5: *Consider the following function*:

$$\varphi(\boldsymbol{x}) = \max\{\boldsymbol{c}'\boldsymbol{y} | \boldsymbol{A}\boldsymbol{y} \leqq \boldsymbol{F}(\boldsymbol{x})\}$$

where $\boldsymbol{A}$ *is a matrix,* $\boldsymbol{y}$ *and* $\boldsymbol{c}$ *are column vectors (prime denotes transposition) and* $\boldsymbol{F}(\boldsymbol{x})$ *is a vector valued function of the vector* $\boldsymbol{x}$. *All dimensions are such that the matrix operations involved are defined. If* $\boldsymbol{F}(\boldsymbol{x})$ *is concave in each component for* $\boldsymbol{\alpha} \leqq \boldsymbol{x} \leqq \boldsymbol{\beta}$, *where* $\boldsymbol{\alpha}$ *and* $\boldsymbol{\beta}$ *are vector constants, and* $\varphi(\boldsymbol{x})$ *is defined for* $\boldsymbol{\alpha} \leqq \boldsymbol{x} \leqq \boldsymbol{\beta}$, *then* $\varphi(\boldsymbol{x})$ *is concave for* $\boldsymbol{\alpha} \leqq \boldsymbol{x} \leqq \boldsymbol{\beta}$.

Proof. Let $\boldsymbol{y}(\boldsymbol{x}) \in Y(\boldsymbol{x}) = \{\boldsymbol{y} | \varphi(\boldsymbol{x}) = \boldsymbol{c}'\boldsymbol{y}, \boldsymbol{A}\boldsymbol{y} \leqq \boldsymbol{F}(\boldsymbol{x})\}$.

If $\boldsymbol{\alpha} \leqq \boldsymbol{x}_1 \leqq \boldsymbol{x}_2 \leqq \boldsymbol{\beta}$ then $\boldsymbol{A}\boldsymbol{y}(\boldsymbol{x}_1) \leqq \boldsymbol{F}(\boldsymbol{x}_1)$, $\boldsymbol{A}\boldsymbol{y}(\boldsymbol{x}_2) \leqq \boldsymbol{F}(\boldsymbol{x}_2)$ and

$$\begin{aligned}\boldsymbol{A}[\mu\boldsymbol{y}(\boldsymbol{x}_1) + (1-\mu)\boldsymbol{y}(\boldsymbol{x}_2)] &\leqq \mu\boldsymbol{F}(\boldsymbol{x}_1) + (1-\mu)\boldsymbol{F}(\boldsymbol{x}_2) \\ &\leqq \boldsymbol{F}[\mu\boldsymbol{x}_1 + (1-\mu)\boldsymbol{x}_2],\end{aligned}$$

where $0 \leqq \mu \leqq 1$. Therefore it follows that

$$\boldsymbol{c}'[\mu\boldsymbol{y}(\boldsymbol{x}_1) + (1-\mu)\boldsymbol{y}(\boldsymbol{x}_2)] \leqq \boldsymbol{c}'\boldsymbol{y}[\mu\boldsymbol{x}_1 + (1-\mu)\boldsymbol{x}_2],$$

or

$$\mu\varphi(\boldsymbol{x}_1) + (1-\mu)\varphi(\boldsymbol{x}_2) \leqq \varphi[\mu\boldsymbol{x}_1 + (1-\mu)\boldsymbol{x}_2].$$

This completes the proof.

With the preceding theorems we may prove

THEOREM 6: $U(\lambda)$ *is bounded, continuous, piecewise linear, nondecreasing and concave for* $m \leqq \lambda < \infty$.

Proof. $U(\lambda)$ is bounded because a_{ij} and y_{ij} are bounded for all $(i,j) \in P$. That it is concave follows from Theorem 5 because by substituting $t_0 = 0$, $t_n = \lambda$ in (1) we see that $\boldsymbol{F}(\boldsymbol{x})$ becomes linear function of λ in each component. That $U(\lambda)$ is continuous is clear. That it is piecewise linear follows from Theorem 1. Since $\{\boldsymbol{y}', \boldsymbol{t}'\}$ is an optimal feasible schedule for $0 \leqq \theta \leqq \theta_0$, we have that (3) is linear in θ for $0 \leqq \theta \leqq \theta_0$. Finally, that $U(\lambda)$ is nondecreasing follows from the fact that it is concave for $m \leqq \lambda < \infty$ and nondecreasing for $\lambda \geqq M$.

A NETWORK FLOW ALGORITHM

IT REMAINS to develop a method for solving (4) and (5). To do this consider the dual of (4) and (5), called the *restricted primal* problem:

Find u_{ij}, $(i,j) \in P$, *that maximize the linear form*

$$\sum_{(i,n) \in Q_1} u_{in} \tag{6}$$

subject to

$$\sum_{(i,j) \in P} u_{ij} - \sum_{(j,k) \in P} u_{jk} = 0, \quad 1 \leqq j \leqq n-1 \tag{7}$$

and

$$0 \leqq u_{ij} \begin{cases} \leqq a_{ij}, & (i,j) \in Q_1 \cap Q_2 \\ = a_{ij}, & (i,j) \in Q_1 - (Q_2 \cup Q_3 \cup Q_4) \\ \geqq a_{ij}, & (i,j) \in Q_1 \cap Q_4 \\ = 0, & (i,j) \in P - Q_1. \end{cases} \tag{8}$$

We may interpret u_{ij} to be the amount of a homogeneous commodity being transported through a network whose nodes correspond to the events of P and whose branches correspond to the activities of P. Equations (7) are flow conservation equations. Capacity restrictions on the allowable flow in a branch are stated in constraints (8). The problem is to maximize the flow into node n subject to the capacity restrictions. The following variant of the Ford-Fulkerson flow algorithm [2] is used to solve it.

Assume that (7) and (8) are consistent and a feasible set of u_{ij} is available. [If $y_{ij}=D_{ij}$ for all $(i,j)\in P$, the initial schedule of Theorem 2, then $u_{ij}=0$ for all $(i,j)\in P$ is a feasible solution. The more general case is treated in Theorem 12.] The process for solving the restricted primal problem consists of two parts:

Part I. In this part labels of the form $(\pm i,h)$ are attached to nodes in accordance with the following rules:

1. Label origin with the label $(-, \infty)$.

2. Consider any labeled node, i, not yet scanned. Suppose node i is labeled $(\pm k,h)$. (*a*) If $(i,j)\in Q_1\cap Q_2$ for some unlabeled node j and $u_{ij}<a_{ij}$, attach the label $(+i, \min[h, a_{ij}-u_{ij}])$ to node j. (*b*) If $(i,j)\in Q_1\cap(Q_3\cup Q_4)$ for some unlabeled node j, attach the label $(+i,h)$ to node j. (*c*) If $(i,j)\in Q_1-(Q_2\cup Q_3\cup Q_4)$ for some unlabeled node j, leave node j unlabeled.

3. Consider any unlabeled node i not yet scanned and suppose there is a node j with the label $(\pm k,h)$ such that $(i,j)\in P$. (*a*) If $(i,j)\in Q_1\cap(Q_2\cup Q_3)$ and $u_{ij}>0$, attach the label $(-j, \min[h,u_{ij}])$ to node i. (*b*) If $(i,j)\in Q_1\cap Q_4$ and $u_{ij}>a_{ij}$, attach the label $(-j, \min[h,u_{ij}-a_{ij}])$ to node i. (*c*) If $(i,j)\in Q_1-(Q_2\cup Q_3\cup Q_4)$, leave node i unlabeled.

Use labeling rules **2** and **3** alternately where applicable until it is no longer possible to label an unlabeled node. When applying these rules, if a node is a candidate for a label in several ways, use any applicable label.

When the labeling process terminates, if terminus is labeled, proceed to Part II of the algorithm. If terminus is not labeled, the algorithm terminates, the maximum flow having been obtained.

Part II. In this part we modify the present solution to (7) and (8). Thus, if terminus is labeled $(+k,h)$, replace u_{kn} by $u_{kn}+h$. Terminus cannot be labeled $(-k,h)$ because, by labeling rule **3**, (n,k) would have to be in P, an impossibility. Note that $h>0$, otherwise terminus would not have been labeled.

Now consider event k. In general, if node k is labeled $(+j,m)$, replace u_{jk} by $u_{jk}+h$; if it is labeled $(-j,m)$, replace u_{kj} by $u_{kj}-h$. Now proceed in the same manner to consider node j. Eventually origin will be reached. At that time Part II terminates. Using the new values of u_{ij} and erasing all labels, Part I is repeated.

This completes the rules for the network flow algorithm.

PROPERTIES OF THE FLOW ALGORITHM

WE NOW determine some properties of the flow algorithm.

THEOREM 7: *The flow algorithm with* $u_{ij}=0$, $(i,j)\in P-Q_1$ *constructs a feasible solution to* (7) *and* (8).

Proof. First note that Part II of the algorithm selects a sequence of nodes, S, with the following obvious properties: (a) terminus is the first term of S, (b) origin is the last term of S, (c) if i and j are two successive terms in S, then either $(i,j)\in P$ or $(j,i)\in P$, (d) no term in S appears more than once. Now assume that a feasible set of u_{ij} are given and the nodes have been labeled by the rules of *Part I*.

To show that equation (7) is satisfied after the u_{ij} are modified by Part II, let k, j, i be three successive terms in S. Since the modification of the u's with subscripts i and j, and j and k, respectively, depend on the labels attached to k and j we must consider four cases:

	1	*2*	*3*	*4*
k's label	$(+j,m)$	$(-j,m)$	$(+j,m)$	$(-j,m)$
j's label	$(+i,r)$	$(+i,r)$	$(-i,r)$	$(-i,r)$

In case *1*, u_{jk} becomes $u_{jk}+h$ and u_{ij} becomes $u_{ij}+h$. Thus,

$$\sum_{\substack{(p,j)\in P\\ p\neq i}} u_{pj}+u_{ij}+h-\sum_{\substack{(j,q)\in P\\ q\neq k}} u_{jq}-u_{jk}-h=0.$$

In the same way it can be shown that equation (7) is also satisfied for the remaining cases.

That the modified u's satisfy constraints (8) is clear from the way labels were assigned. This completes the proof.

For the next few theorems we need the following sets: Let I be the set of labeled nodes and J the set of unlabeled nodes obtained at the termination of the flow algorithm. Further, let

$$Q_5=\{(i,j)\,|\,i\in I, j\in J \text{ or } j\in I, i\in J \text{ and } (i,j)\in Q_1\}.$$

Then we obtain

THEOREM 8: *At the termination of the flow algorithm*

$$(a)\quad \text{if } i\in I, j\in J, \text{ then } u_{ij}=a_{ij}, \text{ if } (i,j)\in Q_1-(Q_3\cup Q_4),$$

$$(b)\quad \text{if } i\in J, j\in I, \text{ then } u_{ij}=\begin{cases} a_{ij} \text{ if } (i,j)\in Q_1-(Q_2\cup Q_3) \\ 0 \text{ if } (i,j)\in Q_1\cap(Q_2\cup Q_3), \end{cases}$$

$$(c)\quad \text{if } i\in I \text{ and } (i,j)\in Q_1\cap(Q_3\cup Q_4), \text{ then } j\in I.$$

Proof. (a) follows from labeling rules (2a) and (2c). The first part of (b) follows from labeling rules (3b) and (3c); the second part follows from rule (3a). (c) follows from labeling rule (2b).

THEOREM 9: *σ_{ij} and δ_i defined by*

$$\sigma_{ij}=\begin{cases} 1, \text{ if } (i,j)\in Q_1-(Q_3\cup Q_4) \text{ and } i\in I, j\in J \\ -1, \text{ if } (i,j)\in Q_1-(Q_2\cup Q_3) \text{ and } i\in J, j\in I \\ 0, \text{ otherwise}, \end{cases}$$

and

$$\delta_i=\begin{cases} 0, & i\in I \\ 1, & i\in J \end{cases}$$

constitute a feasible solution to (5).

Proof. It is only required to show that $p_{ij}\geqq 0$, $(i,j)\in Q_1$, the remaining relations of (5) being satisfied trivially. We may enumerate all possibilities as follows:

σ_{ij}	δ_i	δ_j	p_{ij}	
1	0	1	0	$(i,j)\in Q_1-(Q_3\cup Q_4),\ i\in I,\ j\in J$
-1	1	0	0	$(i,j)\in Q_1-(Q_2\cup Q_3),\ i\in J,\ j\in I$
0	1	0	1	$(i,j)\in Q_1\cap(Q_2\cup Q_3),\ i\in J,\ j\in I$
cannot occur				$(i,j)\in Q_1\cap(Q_3\cup Q_4),\ i\in I,\ j\in J$
0	0	0	0	$(i,j)\in Q_1,\ i\in I, j\in I$
0	1	1	0	$(i,j)\in Q_1,\ i\in J, j\in J$

Since $p_{ij}\leqq 0$ in all cases, the σ_{ij} and δ_i constitute a feasible solution to (5).

THEOREM 10: *The flow algorithm constructs an optimal feasible solution to the restricted primal problem, and σ_{ij} and δ_i, defined by Theorem 9, constitute an optimal feasible solution to the restricted dual problem.*

Proof. From the duality theorem of linear programming, if the value of (4) for some feasible solution to (5) equals the value of (6) for some feasible solution to (7) and (8), then the respective feasible solutions are also optimal. Thus, in view of Theorem 7 and Theorem 9, it is only re-

quired to show that the value of (4) equals the value of (6) for the feasible solutions in question.

Let $H=\{i|i\in I,\ (i,j)\in Q_5 \text{ or } (j,i)\in Q_5\}$.

Define a new origin, O', and consider the project, P', consisting of the following events and activities:

$$\text{events of } P' = \{O'\}\cup H\cup J, \qquad \text{activities of } P' = R\cup Q_5\cup T,$$

where $R=\{(O',i)|i\in H\}$, $T=\{(i,j)|i\in J, j\in J\}$.

Let $$\bar{u}_{ij}=\begin{cases} u_{ij} & \text{if } (i,j)\in Q_5\cup T \\ \sum_{(j,k)\in Q_5} u_{jk}-\sum_{(h,j)\in Q_5} u_{hj} & \text{if } (i,j)\in R. \end{cases}$$

Now, by Theorem 8 and Theorem 9,

$$u_{ij}=\begin{cases} a_{ij}\,\sigma_{ij}, & i\in I,\ j\in J, \\ -a_{ij}\,\sigma_{ij}, & i\in J,\ j\in I. \end{cases}$$

Thus

$$\sum_{(O',j)\in R} \bar{u}_{O'j}=\sum_{j\in H}\left[\sum_{(j,k)\in Q_5} u_{jk}-\sum_{(h,j)\in Q_5} u_{hj}\right]=\sum_{(i,j)\in Q_5} a_{ij}\,\sigma_{ij}.$$

Further, it is clear that

$$\sum_{(i,n)\in Q_1} u_{in}=\sum_{(i,n)\in P'} \bar{u}_{in},\ i\in P'.$$

Finally, consider the following series:

$$-\sum_{(O,j)\in P'} \bar{u}_{Oj}+\sum_{j\in P'}\left[\sum_{(i,j)\in P'} \bar{u}_{ij}-\sum_{(j,k)\in P'} \bar{u}_{jk}\right]+\sum_{(i,n)\in P'} \bar{u}_{in}.$$

Note that $\bar{u}_{ij}$, $(i,j)\in P'$, appears twice in this series, once with a plus sign and once with a minus sign. Thus, the series vanishes. Similarly, the double sums in the series vanish. Therefore, since $\sigma_{ij}=0$ if $(i,j)\in P-Q_5$,

$$\begin{aligned}\sum_{(i,j)\in P} a_{ij}\sigma_{ij}=\sum_{(i,j)\in Q_5} a_{ij}\sigma_{ij}&=\sum_{(O,j)\in P'} \bar{u}_{Oj} \\ &=\sum_{(i,n)\in P'} \bar{u}_{in}=\sum_{(i,n)\in Q_1} u_{in}.\end{aligned} \tag{9}$$

Thus the value of (4) equals the value of (6). This completes the proof.

REMARK 3: It is to be noted that Theorem 10 provides a proof of an analogue of the min-cut max-flow theorem. The set Q_5 forms a cut set for Q_1. From (9) we see that a minimum weighted cut sum equals the maximum flow; i.e.,

$$\sum_{(i,j)\in Q_5} a_{ij}\,\sigma_{ij}=\sum_{(i,n)\in Q_1} u_{in}.$$

THEOREM 11: *If in the labeling process of the flow algorithm terminus is given a label of the form* $(+k, \infty)$, *then no feasible schedules of shorter duration exist and the primal-dual algorithm terminates.*

Proof. Assigning a label of the form $(+k, \infty)$ means that the restricted primal problem has an unbounded maximum. It follows from the duality theorem that the restricted dual problem is inconsistent. The conclusion follows from Theorem 1.

Now assume that it is desired to compute the complete project utility function. In virtue of Theorem 3, the computation may begin with the optimal feasible schedule defined by Theorem 2.* A feasible flow for the corresponding restricted primal problem is $u_{ij}=0$, $(i, j)\in P$. As the computation proceeds, the restricted primal problem changes and a feasible flow is required in order to continue the process. The following theorem ensures that such a flow is always available.

THEOREM 12: *If u_{ij}, $(i,j)\in P$, is an optimum flow for the restricted primal problem associated with the optimum feasible schedule $\{\mathbf{y}, \mathbf{t}\}$ of duration λ, then it is also a feasible flow for the restricted primal problem associated with the optimum feasible schedule $\{\mathbf{y}', \mathbf{t}'\}$ of duration $\lambda-\theta_0$ defined by Theorem 1.*

Proof. The flow obviously satisfies (7) for the new schedule. It must also satisfy (8) because, by Theorem 8 and Theorem 9, $u_{ij}=0$ or a_{ij} in the proper place for every case where an equality or inequality has changed, thus satisfying the modified conditions.

REMARK 4: It should be noted that if d_{ij} and D_{ij} are integers for all $(i,j)\in P$ then y_{ij} is an integer, all $(i,j)\in P$, for every characteristic schedule computed by the primal-dual algorithm.

A NONLINEAR EXTENSION

As WAS remarked earlier, the algorithm of the preceding sections may be extended to handle activity utility functions that are piecewise linear, nondecreasing and concave. Such a function may be represented in the following form:

$$F_{ij}(y_{ij})=\min_{1\leqq k\leqq r}[a_{ij}^{(k)}\, y_{ij}+b_{ij}^{(k)}], \quad d_{ij}\leqq y_{ij}\leqq D_{ij},$$

where $\infty>a_{ij}^{(1)}>\cdots>a_{ij}^{(r)}\geqq 0$ and each value of k is necessary. We now show how such functions may be incorporated within the formulation of (1), (2), and (3). In order to simplify notation we drop the subscripts i and j from the applicable variables and restrict, without loss of generality, our discussion to one project activity and its terminal events.

We replace activity (i, j) with the nonlinear utility function $F(y)$ by a series of r activities with linear utility functions in the following way: Consider the sequence $\{i_k\}$ of events where

$$i=i_0<i_1<\cdots<i_r=j.$$

* FULKERSON has suggested that when it is desired to start the computation at an arbitrary point a variant of his 'Out-of-Kilter' method[4] might be used to advantage.

Then (i,j) is replaced by $(i_0,i_1), \cdots, (i_{r-1},i_r)$. The duration of (i_{k-1},i_k) is $y^{(k)}$ and the time at which event i_k occurs is $t^{(k)}$. Of course,

$$y^{(k)}+t^{(k-1)}-t^{(k)}\leqq 0.*$$

Each of these new activities is made to correspond to a linear section of $F(y)$. Except for the first activity, the duration of each activity is bounded by zero and the length of the projection on the y-axis of the linear section to which it corresponds. In summary, the duration limits are

$$D^{(1)}=(b^{(2)}-b^{(1)})/(a^{(1)}-a^{(2)})\geqq y^{(1)})\geqq d,$$

$$D^{(k)}=\frac{b^{(k+1)}-b^{(k)}}{a^{(k)}-a^{(k+1)}}-\frac{b^{(k)}-b^{(k-1)}}{a^{(k-1)}-a^{(k)}}\geqq y^{(k)}\geqq 0. \qquad (2\leqq k\leqq r)$$

The utility function of activity (i_{k-1},i_k) is then

$$F^{(k)}[y^{(k)}]=\begin{cases} a^{(1)}y^{(1)}+b^{(1)}, & (k=1) \\ a^{(k)}y^{(k)}. & (2\leqq k\leqq r) \end{cases}$$

Clearly, the nonlinear problem has been reduced to a linear problem of the form of (1), (2), and (3).

To show that an optimal solution to the linear problem solves the nonlinear problem, let the duration of (i,j) in an optimal solution to the nonlinear problem be y. Then, as is easily seen,

$$y^{(k)}=\min[D^{(k)},t^{(k)}-t^{(k-1)}], \qquad t^{(0)}=t_i,$$

$$t^{(k)}=\min[D^{(k)}+t^{(k-1)},y] \quad \text{for} \quad 1\leqq k\leqq r,$$

constitutes a feasible solution to the linear problem with the same project utility value.

On the other hand, if $y^{(k)}$, $t^{(k)}$, $1\leqq k\leqq r$, with $t^{(0)}=t_i$ and $t^{(r)}=t_j$ is an optimal feasible solution to the linear problem, then, since $a^{(k)}\geqq 0$ for all k, we may take

$$y^{(k)}=\min[D^{(k)},\ t^{(k)}-t^{(k-1)}].$$

This equality must occur if $a^{(k)}>0$. When $a^{(k)}=0$ we may use the equality to define $y^{(k)}$ without losing anything essential to our argument.

Therefore, $y=\sum_{k=1}^{k=r} y^{(k)}$ is a feasible solution to the nonlinear problem. However, since $a^{(k)}>a^{(k+1)}$, if $y^{(k)}<D^{(k)}$ then $y^{(k+1)}=0$ and $F(y)=\sum_{k=1}^{k=r} F^{(k)}[y^{(k)}]$.

It follows, therefore, that y is an optimal solution to the nonlinear problem. Thus, the solution to the linear problem solves the nonlinear problem.

REMARK 5: The ability to handle the nonlinear problem within the frame-

* Here $t^{(0)}$ and $t^{(k)}$ are not to be confused with earliest and latest event times.

work of the linear problem is not only useful of itself, but also provides an important method for reducing the effort required to solve extremely large project scheduling problems. Quite often large portions of a project diagram form individual projects in their own right. They connect into the whole project at only two points, their origin and terminus, respectively. Each of these subprojects can be solved separately, a spectrum of solutions and a utility function being obtained in each case. Each subproject may now be replaced in the project by a single activity with the same utility function as the subproject it replaces (see diagramming *Rule 3* on aggregated activities). The original project is now smaller by many activities and events and can be solved more easily than before.

REMARK 6: Of course explicit replacement of an activity with a nonlinear utility function by a series of r activities is not required in practice. Thus, the project diagram for the nonlinear problem is not complicated when reduced to the linear case.

APPLICATIONS OF THE PROJECT UTILITY FUNCTION

THERE ARE many possibilities available for measuring utility. For example, it may be desirable to minimize cost, effort, loss, manpower, etc., or maximize profit, sales, return on investment, efficiency, quality, etc. Which of these criteria is selected depends largely on the nature of the project and the desires of management. The two most commonly used in industrial project work are cost and return on investment. We will limit our discussion to these two only.

Let us assume that the utility of an activity is measured in terms of its cost. Maximizing utility then means minimizing cost. The result of the project utility function computation is a project cost curve that is piecewise linear, nonincreasing, and convex where it is defined. However, this cost curve generally only reflects the direct costs involved in performing project activities. These costs include such things as labor, equipment, and materials—the 'direct' costs of the project. The computed cost curve will thus be called the *direct cost curve* of the project.

Clearly there are other costs that contribute to the total project cost such as overhead and distributives, and perhaps penalties for not completing the project or a portion of it by a certain time. These external costs must also be taken into account when management plans how the project should be implemented relative to over-all objectives. The major portion of the external costs usually vary only with the duration of the project. Thus, they form a cost curve that will be called the *indirect cost curve* of the project.

A typical question that management might ask is "How should the project be implemented so that the total investment cost is a minimum?"

The answer to this question can be approximated by adding the *direct* and *indirect* cost curves together to form a *total investment cost curve* for the project and then selecting the schedule corresponding to the minimum total investment.

Another situation of interest occurs when it is desired to determine how to implement a project so that return on investment is maximized. For example, the project might be to launch a new product in time to meet a rising market. The later the project is completed the smaller will be the manufacturer's share in the market. On the other hand, in the proper circumstances, the more the project completion date is moved up the more the project will cost. Both of these factors must be weighed in order to determine when production should begin.

In this case we compute the estimated loss to the manufacturer for each day the start of production is delayed and obtain a *loss function*. This loss function is then added to the total investment cost curve. The project duration where the minimum of this composite function occurs is the point where return on investment is maximized.

Regardless of the over-all objective criterion, once management has the project utility function, it is in a position to select *one* schedule from *many* alternatives.

Although the preceding remarks are a little oversimplified they illustrate what can happen in practice. An actual situation where it is required to maximize return on investment occurs in overhaul and maintenance projects during shutdowns of chemical plants and oil refineries. Every hour of downtime incurs an hour of lost production. When reserves are small it is important that the shutdown be as short as possible.

We may illustrate this situation by means of the 'renew pipeline project' of Fig. 5. However, we first note that there are several pieces of information about the project that have not been supplied in Table I. We will assume that dummy activities (2, 3), (5, 7) and (10, 11), which serve only to preserve sequence relations, cost nothing and take no time to perform. That is, if $(i, j) = (2, 3), (5, 7)$, or $(10, 11)$ let $a_{ij}=b_{ij}=D_{ij}=0$.

Further, it is assumed that the Operating Department will not release the line to the Maintenance Department for 15 days (or 360 hours) in order to complete a production run. This delivery costs nothing, so $a_{03}=b_{03}=0$. However, $d_{03}=D_{03}=360$ hours.

Finally, we assume that due to prior commitments, the Maintenance Department cannot assign a man to activity (1,2), developing a materials list, for six days (or 144 hours). Thus for lead time, activity (0,1), we have $d_{01}=D_{01}=144$ hours. It is assumed that lead time costs nothing, so $a_{01}=b_{01}=0$.

On the basis of this information, that of Table I, and the project diagram

of Fig. 5, we may apply the algorithm of the previous sections to obtain the cost curve approximated in Fig. 6.

The minimum time in which this project may be completed is 418 hours at an approximate minimum cost of $6,807. If each activity were performed at minimum time, the project duration would still be 418 hours but would cost $9,270. The longest the project need take to attain the minimum cost of $4,950 is 475 hours.

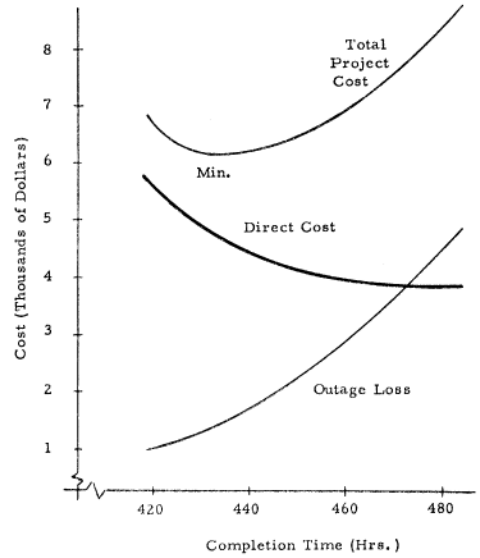

Figure 6

If the storage tank fed by the pipeline contains enough product to supply production units downstream in the process for some feasible duration of the shutdown, then a 'best' schedule to operate with is easily obtained provided the stock is not too much in abundance. In the latter case it would be well to re-analyze the project on the basis of a shorter work day and thus save on overtime. However, if the stock is in short supply a re-evaluation of the project is also in order to see if it is possible by changing the method of doing the project, whether a shorter project duration is possible. It may also be that the project is really not necessary at the time. There are several other possibilities.

CALENDAR LIMITS

HAVING selected a schedule on an objective basis, it becomes necessary to furnish the entire project organization with a detailed working schedule that conforms to the management decision. Supervision, engineers, expediters, technicians, and craftsmen must know, for each part of the project,

what is required *when*, in order to implement the decision. It is also necessary to index limiting deliveries and activities in order to prepare for unexpected reverses in progress. In this section we consider how to compute information of this type.

Given an assignment of durations to activities in the project P. Generally there are many ways to assign event times that result in a feasible schedule of duration $\leqq\lambda$. Certainly, relative to a project start time of zero there is a smallest possible value any event time can be. We denote the earliest event time for event j by $t_j^{(0)}$. If we define the length of a chain or path of activities from origin to event j to be the sum of their assigned durations, then it is easy to see that the value of $t_j^{(0)}$ is the length of the longest path. The length of the longest path from j origin to event j, or the earliest time for event j, may be determined inductively in the obvious way as follows:*

$$t_0^{(0)}=0, \qquad t_j^{(0)}=\max_{(i,j)\in P}\,(y_{ij}+t_i^{(0)}) \quad \text{for} \quad 1\leqq j\leqq n.$$

Similarly, relative to a fixed completion time, λ, for the project, each event time has a largest value denoted by $t_i^{(1)}$. It is equal to the length of the longest path from event i to terminus and may be computed analogously as follows:

$$t_n^{(1)}=\lambda, \qquad t_i^{(1)}=\min_{(i,j)\ P}\,(t_j^{(1)}-y_{ij}) \quad \text{for} \quad 0\leqq i<n.$$

It is now clear that $t_i^{(0)}$ and $t_j^{(1)}$ are the limits between which activity (i,j) must occur. Indeed, we are immediately provided as a result with the following important information about activity (i,j):

$$\begin{aligned}
\text{Earliest start time} &= t_i^{(0)},\\
\text{earliest completion time} &= t_i^{(0)}+y_{ij},\\
\text{latest start time} &= t_j^{(1)}-y_{ij},\\
\text{latest completion time} &= t_j^{(1)},\\
\text{maximum time available} &= t_j^{(1)}-t_i^{(0)}.
\end{aligned}$$

If there is a path from origin to terminus whose length equals the duration of the schedule, it is called a *critical-path*. All the activities in a critical-path are limiting in the sense that a delay in any one of them will cause a comparable delay in the completion of the project. Therefore, they are called *critical activities*. Of course, a schedule, $\{\boldsymbol{y},\boldsymbol{t}\}$, of duration λ possesses a critical-path and critical activities if, and only if, $\lambda=t_n^{(0)}$.

An easy way to determine if an activity is critical or not is to observe that activity (i,j) is critical if, and only if, it utilizes the maximum time available to it, i.e., $y_{ij}+t_i^{(0)}-t_j^{(1)}=0$. It follows that if (i,j) is critical then $t_i^{(0)}=t_i^{(1)}$ and $t_j^{(0)}=t_j^{(1)}$.

* It is easy to show that if the algorithm is started with the schedule of Theorem 2, then the t_i of Theorem 1 are always the earliest event times ($t_i^{(0)}$).

On the other hand, if the maximum time available for an activity exceeds its duration, it is called a *floater*. Some floaters can be delayed temporarily or displaced in time without interfering with the completion of any following activities. However, others, if displaced, although not affecting the completion of the project may delay a whole chain of following activities.

TABLE II

Sequence		Activity code	Description	Cost	Duration	Earliest		Latest		Float		
i	*j*					Start	Finish	Start	Finish	Total	Free	Ind.
0	1	L.T.	Lead time	0	144	0	144	11	155	11	0	0
0	3	—	**Deliver line**	0	360	0	—	—	360	0	0	0
1	2	*A*	Develop mat'l list	100	8	144	152	155	163	11	0	0
2	3	—	Dummy	0	0	152	152	360	360	218	218	207
2	4	*C*	Erect scaffold	300	12	152	164	356	368	204	204	193
2	5	*J*	Procure valves	300	225	152	377	178	403	26	26	15
2	6	*E*	Procure pipe	850	200	152	352	163	363	11	0	0
3	4	***B***	**Deactivate line**	150	8	360	—	—	368	0	0	0
4	5	***L***	**Remove pipe & valves**	400	35	368	—	—	403	0	0	0
5	7	—	**Dummy**	0	0	403	—	—	403	0	0	0
5	8	*K*	Place valves	100	8	403	411	417	435	14	14	14
6	7	*F*	Prefab sections	1200	40	352	392	363	403	11	11	0
7	8	***G***	**Place new pipe**	800	32	403	—	—	435	0	0	0
8	9	***H***	**Weld pipe**	100	8	435	—	—	443	0	0	0
9	10	*I*	Fit-up	100	8	443	451	457	465	14	0	0
9	11	***M***	**Insulate**	300	24	443	—	—	467	0	0	0
10	11	—	Dummy	0	0	451	451	467	467	16	16	2
10	12	*N*	Pressure test	50	6	451	457	465	471	14	14	0
11	12	***D***	**Remove scaffold**	100	4	467	—	—	471	0	0	0
12	13	***P***	**Clean up**	100	4	471	—	—	475	0	0	0

In order to determine, in advance, the character of any floater, several measures of float have been tested. The following measures, though not exhausting the possibilities, have been found useful. Their interpretations are clear.

$$\text{total float} = t_j^{(1)} - t_i^{(0)} - y_{ij},$$

$$\text{free float} = t_j^{(0)} - t_i^{(0)} - y_{ij},$$

$$\text{independent float} = \max(0, t_j^{(0)} - t_i^{(1)} - y_{ij}).$$

From our previous remarks it is clear that an activity is critical if, and only if, its total float is zero.

Free float measures the amount an activity may be displaced when all other activities are started as early as possible (a common practice in industrial projects). Note that Q_1 is the set of all activities whose free float is zero.

Independent float measures the amount an activity may be displaced no matter what the state of the other activities in the project, provided they remain within their calendar limits.

Table II is a summary of the information supplied to supervision for the pipeline renewal problem of Fig. 5. The durations used are the *normal* durations of Table I. Of course, the times shown are all relative to origin and must be transformed into calendar times. The information shown can be easily put into the form of bar charts for ease of reading.

The critical activities for this schedule are shown in boldface in Table II. The remainder of the table is self-explanatory. The total cost adds up to \$4950.

REMARK 7: It is of interest to note that in all 'real' projects studied to date, less than 10 per cent of the activities have been critical—even for the shortest duration schedules. This fact points out the fallacy, prevalent in project work, of embarking on an 'across-the-board' crash program when expediting the project end date is required. This is probably an illustration of Pareto's principle that "In any series of elements to be controlled, a selected small fraction, in terms of numbers of elements, always accounts for a large fraction, in terms of effect." There are exceptions, of course, and the pipeline renewal problem here illustrates this. In the minimum duration schedule, 15 out of the 20 activities involved are critical.

REFERENCES

1. A. ASTRACHAN, "Better Plans Come From Study of Anatomy of an Engineering Job," *Business Week*, 60–66 (March 21, 1959).
2. L. R. FORD AND D. R. FULKERSON, "A Simple Algorithm for Finding Maximal Network Flows and an Application to the Hitchcock Problem," *Canadian J. Math.* **9**, 210–218 (1957).
3. D. R. FULKERSON, "A Network Flow Computation for Project Cost Curves," Rand Paper P-1947, March 18, 1960, *submitted for publication.*
4. ———, "An Out-of-Kilter Method for Minimal Cost Flow Problems," Rand Paper P-1825, Jan. 18, 1960, *submitted for publication.*
5. S. GASS AND T. SAATY, "The Computational Algorithm for the Parametric Objective Function," *Naval Res. Log. Quart.* **2**, 39–46 (1955).
6. J. E. KELLEY, JR., "Computers and Operations Research in Road Building," *Operations Research, Computers and Management Decisions*, Symposium Proceedings, Case Institute of Technology, Jan. 31, Feb. 1, 2, 1957.
7. ———, "The Construction Scheduling Problem (A Progress Report)," UNI-

VAC Applications Research Center, Remington Rand UNIVAC, Philadelphia, April 25, 1957.
8. ———, "Extension of the Construction Scheduling Problem: A Computational Algorithm," UNIVAC Applications Research Center, Remington Rand UNIVAC, Philadelphia, Nov. 18, 1958.
9. ———, "Parametric Programming and the Primal-Dual Algorithm," *Opns. Res.* **7,** 327–334 (1959).
10. ——— AND MORGAN R. WALKER, "Critical-Path Planning and Scheduling: An Introduction," Mauchly Associates, Inc., Ambler, Pa., 1959.
11. ——— AND ———, "Critical-Path Planning and Scheduling," *Proc. Eastern Joint Computer Conference,* 160–173, Boston, Dec. 1–3, 1959.
12. D. G. MALCOLM, J. H. ROSEBOOM, C. E. CLARK, AND W. FAZAR, "Application of a Technique for Research and Development Program Evaluation," *Opns. Res.* **7,** 646–669 (1959).
13. R. L. MARTINO, "New Way to Analyze and Plan Operations and Projects Will Save You Time and Cash," *Oil/Gas World,* 38–46 (Sept. 1959).
14. ———, "How 'Critical-Path' Scheduling Works," *Canadian Chemical Processing* (Feb., 1960).