OCCASION

This publication has been made available to the public on the occasion of the 50th anniversary of the United Nations Industrial Development Organisation.

DISCLAIMER

FAIR USE POLICY

CONTACT

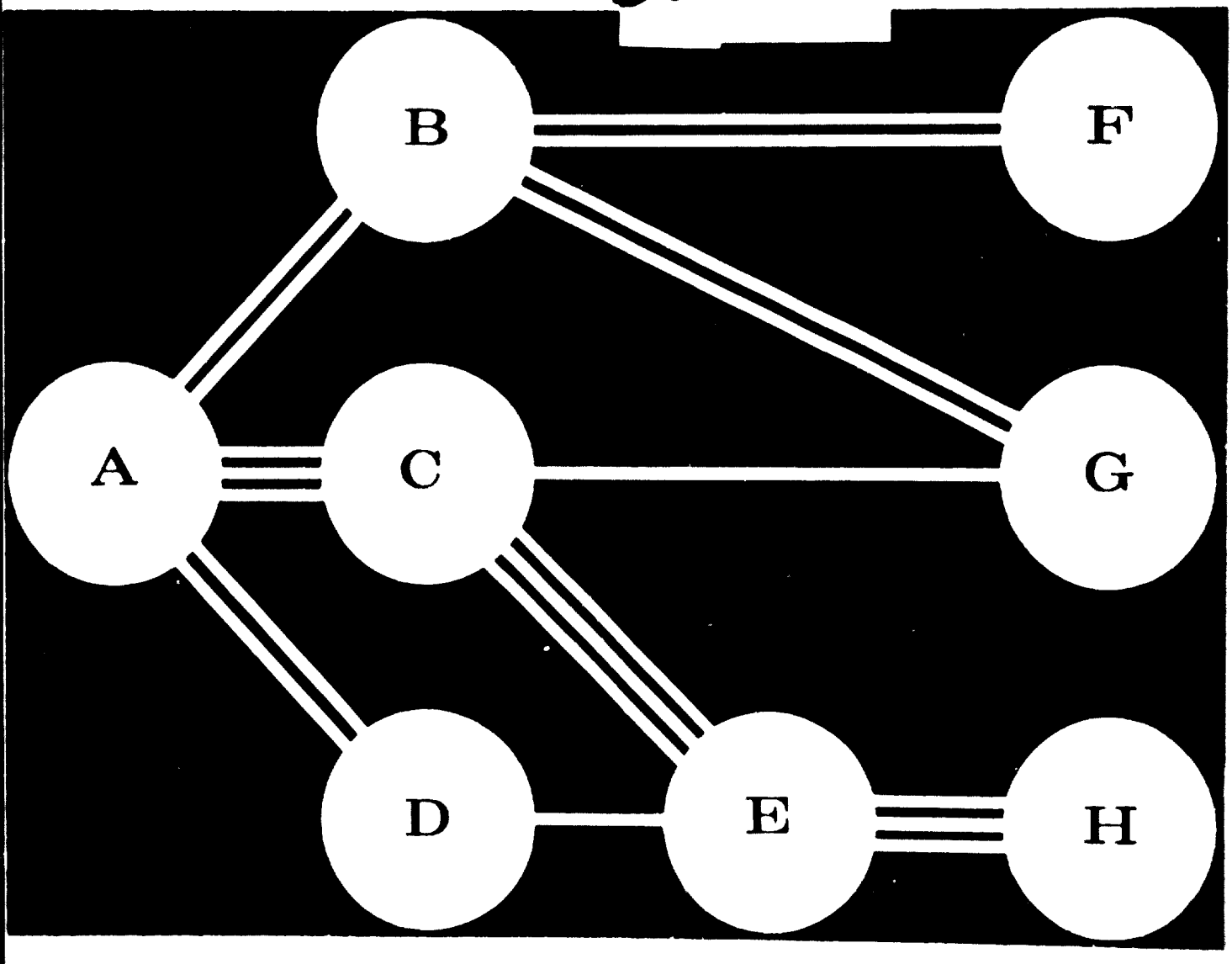
Please contact publications@unido.org for further information concerning UNIDO publications.

For more information about UNIDO, please visit us at www.unido.org

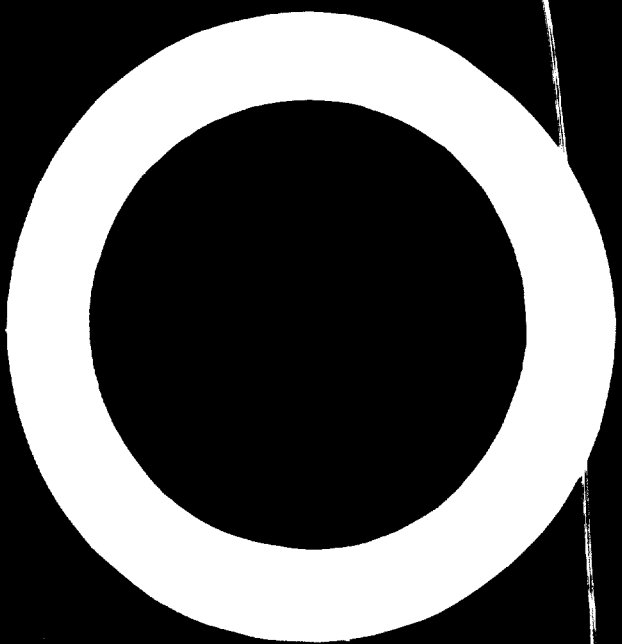
UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION
Vienna International Centre, P.O. Box 300, 1400 Vienna, Austria
Tel: (+43-1) 26026-0 · www.unido.org · unido@unido.org

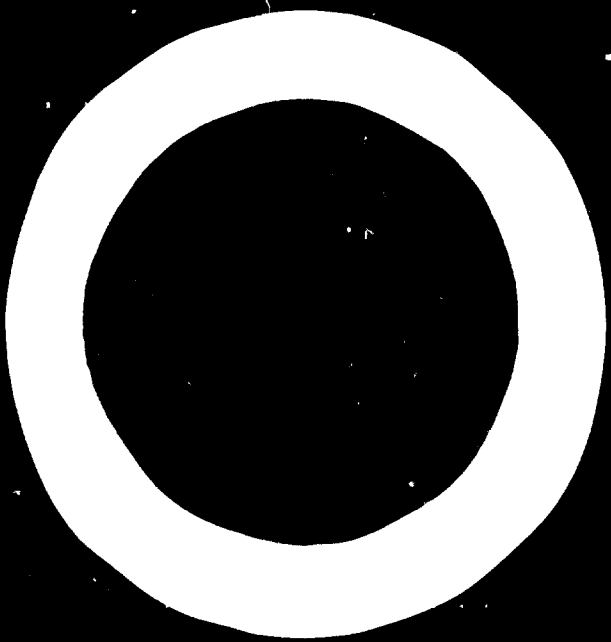
INDUSTRIAL IMPLEMENTATION SYSTEMS: No.1

PROGRAMMING AND CONTROL OF IMPLEMENTATION OF INDUSTRIAL PROJECTS IN DEVELOPING COUNTRIES

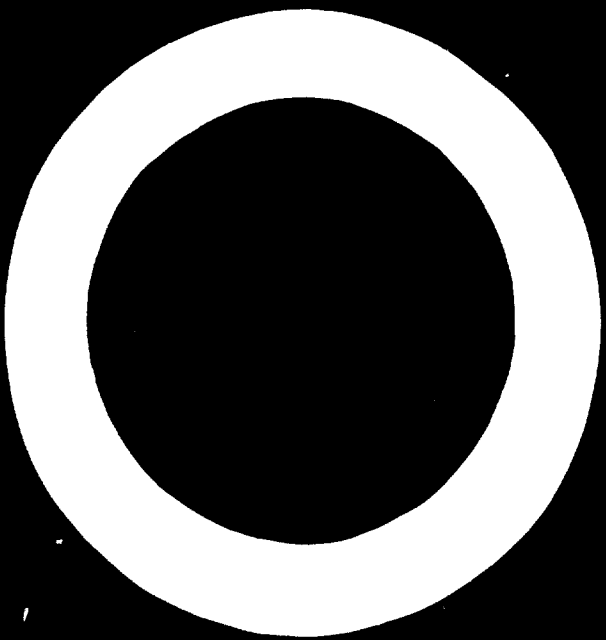


UNITED NATIONS





PROGRAMMING AND CONTROL OF IMPLEMENTATION OF INDUSTRIAL PROJECTS IN DEVELOPING COUNTRIES



UNITED NATIONS INDUSTRIAL DEVELOPMENT ORGANIZATION
VIENNA

INDUSTRIAL IMPLEMENTATION SYSTEMS: No. 1

PROGRAMMING AND CONTROL OF IMPLEMENTATION OF INDUSTRIAL PROJECTS IN DEVELOPING COUNTRIES

UNITED NATIONS
NEW YORK, 1970

ID/SER. L/1

UNITED NATIONS PUBLICATION
Sales No.: E.70.II.B.18
Price: $U.S. 2.00
(or equivalent in other currencies)

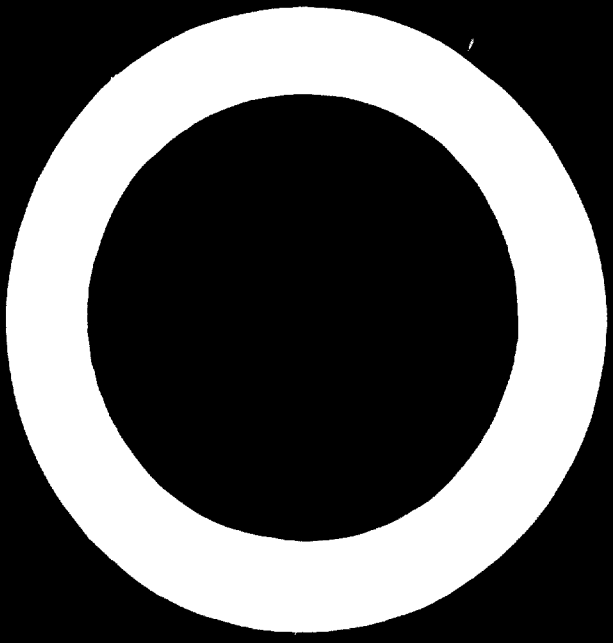
FOREWORD

Many developing countries have devoted considerable effort to formulating comprehensive and consistent industrial development programmes but have not devoted similar effort to carrying out these programmes, with the result that many countries have failed to attain their industrial development goals. Experience has shown that a developing country encounters various obstacles in its efforts to implement industrial projects, some of which are beyond its control. Bearing this in mind, the United Nations Industrial Development Organization (UNIDO) has initiated a series of publications entitled "Industrial Implementation Systems" dealing with problems encountered in the implementation and follow-up of industrial programmes and projects.

Experts of the United Nations Development Programme and other international advisers have reported that the lack of programming and control of implementation of industrial projects is one of the most important of the various factors contributing to implementation shortcomings. In most developing countries no formal techniques or procedures for such programming have been available. In the absence of these it has not been possible to draw up successful plans of operation or effective implementation schedules. Thus projects have been hampered by delays, costs have exceeded estimates, and project implementation has fallen short of expectations.

This publication, the first in a series, is concerned therefore with operational techniques for programming and control of the implementation of industrial projects in the hope that the ideas expressed may reach technical assistance experts and those working in planning, programming, implementation and follow-up in developing countries.

This study was prepared by the secretariat of UNIDO with the help of a consultant, John W. Fondahl, of Stanford University, Stanford, California, USA.



CONTENTS

EXPLANATORY NOTES

CPM — Critical Path Method
PERT — Project Evaluation and Review Technique
ES — Activity earliest start time
EF — Activity earliest finish time
LS — Activity latest start time
LF — Activity latest finish time
FF — Activity free float
TF — Activity total float
NIL — Network interaction limit

INTRODUCTION

For effective project implementation, management must work according to an implementation plan that indicates the chronological order of the various component activities or tasks that must be completed before a project can start operations. Management must ensure that all these component activities are accomplished according to schedule. But, since resources in most developing countries are frequently in short supply, project implementation is usually rather difficult. Funds to finance undertakings are frequently limited. Implementation delays waste scarce resources, increase maintenance and repair work and hence increase costs. Skilled and experienced manpower is frequently scarce, and thus dependable estimates of productivity and resulting time, resource, and cost requirements are not easy to make. For the same reasons actual execution frequently deviates from the forecast execution; this means that if an implementation plan is to continue to guide the execution of a project, it must be updated frequently. On the other hand, early completion with a minimum of delays allows the new facilities to start production and furnish output that is usually badly needed.

Programming for implementation of industrial programmes and projects in the environment of a developing country will be more effective if it does not depend on highly sophisticated data-processing equipment that is not readily available. Manual methods applicable locally by project personnel can produce better results than procedures that require electronic computer processing at remote facilities, often performed by outside experts lacking close daily contact with the project.

The implementation of any important and complex project should be carefully programmed. Objectives should be clearly defined, and those persons having the deepest knowledge of the work should participate in this programming. The term "implementation programming" is used here to indicate not only the process of subdividing the project into its component activities and developing their sequential relationships but also to include for each of these activities the selection of methods, the assignment of resources, the estimating of time requirements, and the establishment of scheduling data. The most effective techniques available for implementation programming should be applied. Advanced methods such as those for time-cost trade-offs and resource allocation can generally be used to improve implementation plans initially developed by the more basic and conventional procedures.

Once an implementation plan has been formulated it must be communicated to those responsible for its execution; then it must be carried out. An implementation plan is of little value unless it is actually executed.

To be executed successfully, it is not enough that the accomplishment of each component activity be technically feasible the plan as a whole must be practicable. A consideration of resource requirements will make this evident. The performance of each project activity demands the utilization of various resources; these include certain labour skills and different types of equipment. The total requirement for any resource at any moment must not exceed the level of its availability. Unreasonable demands result in an unrealistic implementation plan, which is impossible to carry out.

Even an implementation plan that is initially sound and well conceived may cease to be so during its execution; unforeseen conditions may be encountered that present problems. Actual duration times required for individual tasks may vary from those forecast by the most competent estimators. Suppliers of services and materials may fail to perform at the time or in the manner promised. External factors over which there is little or no control, such as unusual weather, labour strikes, changing regulatory requirements, etc., may prevent performance according to the original plan. Therefore, techniques for programming and control of implementation should be dynamic ones that permit modifications when necessary or advantageous. If an implementation plan is not altered to reflect changes, it ceases to be valid. This has been the fate of many implementation plans that were competently conceived, often at great expense, and that originally offered an excellent solution to the problem involved. Soon after the undertaking had commenced, changes occurred, but the plan was not updated. Subsequent work was performed in the same manner as if it had not been programmed, or, what can be still more hazardous, in accordance with a plan that had ceased to be valid.

It is essential, as indicated above, that an implementation plan be realistic and that it be kept updated. Also, there should be continued programming in greater detail as the job progresses, and continual replanning of the existing strategy. No matter how excellent the original programming is, only a certain amount of detail should be developed at an early stage. In view of the inevitable changes that occur as a project proceeds, extremely detailed over-all programming of implementation is not justified. Detailed implementation programming for limited periods, however, should be carried on as work progresses and should supplement the original or updated master implementation programming. There should be a constant effort to improve job performance through replanning. Although unforeseen conditions may often be encountered that present problems, sometimes these unforeseen conditions may offer opportunities for improvements. Also, a good knowledge of job conditions and actual productivity levels may make advantageous changes possible. While it is commendable to follow faithfully a well-conceived plan of implementation in order to benefit from the skill and thought that have gone into it, it is even more commendable to continue to seek better solutions and, when they are found, to change the implementation plan accordingly. Of course, it is important to analyse care-

fully the changes to ensure that they are in fact improvements. It is also important that the existing plan be updated, not abandoned.

For an implementation plan to be executed successfully, those who manage the work must know the relative importance of the elements of the plan so that they can concentrate their efforts where they are needed most. For example, the effects of deviations in the timing or the sequencing of project activities can range from negligible to extremely serious. The information developed by the implementation programming procedure should indicate to management the nature of these effects.

To determine appropriate corrective action after a change has occurred, it is necessary to establish that a problem exists. Schedule updating can indicate the effects of a change on other activities, on the project completion date, and on the timing of important intermediate events. Resource updating can indicate whether these requirements have been thrown out of balance or have become excessive. Once the extent of the problem has been discovered a realistic implementation plan can be re-established by using reprogramming techniques. Such techniques should allow the costs of these corrective actions to be determined.

In summary, implementation of an industrial programme or project must be intelligently programmed, and the implementation plan must be effectively executed. These two principal requirements involve:

(a) Programming of project implementation:

Definition of objectives;
Breakdown of work into component activities;
Statement of sequential relationships;
Determination of methods, resource requirements and costs;
Time estimates;
Calculation of resulting time schedule;
Calculation of resulting resource schedules;
Improvements by consideration of alternative strategies;
Improvements by time-cost trade-offs;
Improvements by resource-allocation methods.

(b) Control of project implementation:

Communication of a realistic implementation plan;
Updating of plan as changes occur;
Expansion of basic implementation plan in greater detail;
Continual attempts to improve through reprogramming.

Chapter 1

SCOPE AND APPROACH OF THIS PUBLICATION

The purpose of this publication it to present operational techniques for identifying component activities of projects, determining their sequential relationships and representing them in a network diagram, making time-cost trade-off decisions and allocating resources. The time-cost trade-off problem arises because most of the activities into which the over-all project is subdivided can be performed by alternative approaches requiring different amounts of time, resources and, hence, expense. Generally, methods of performance that decrease the time requirements tend to increase direct, or variable, costs. These direct costs rise more rapidly in some cases than in others as work is expedited. If a project completion time is arbitrarily specified or is set by external controls, the time-cost trade-off procedure attempts to develop the combination of activity scheduling that meets the completion deadline with the lowest total direct cost. A more general problem arises when the procedure is applied also to determine the most favourable completion date. In this case, since reductions in project duration result in lower indirect, or fixed, costs, the time-cost trade-offs are made with the objective of finding the schedule that gives the lowest combination of direct and indirect costs, i.e., the lowest total costs.

The resource-allocation problem is to determine the schedule that satisfies resource restraints in as favourable a manner as possible. Most activities in a project require the use of one or more resources. If these requirements are stated and an initial schedule is developed, the number of units of each separate resource needed during each time period can be determined. If the demands at any time exceed the availability of any resource, some activities must be rescheduled. When rescheduling requires that the duration of a project be extended, a principal objective should be to minimize the extension. Frequently a time-cost trade-off approach can be used to do this. Excessive resource requirements can usually be satisfied by means other than mere rescheduling, but these may involve higher costs. Although time-cost trade-offs and resource allocation have generally been considered separately, greater attention should be devoted to developing procedures that make use of their interrelationships. A secondary resource-allocation problem is to keep resource requirements as constant as possible. Peaks and valleys in resource schedules invariably indicate uneconomic performance. Improvements can be achieved to some degree by rescheduling. The utilization of idle resources can also offer important opportunities for

favourable time-cost trade-offs. This again illustrates an interrelationship between time-cost trade-offs and resource allocation.

A more complex resource-allocation problem occurs when it is necessary to schedule several projects concurrently that draw resources from the same resource pools. This multi-project problem presents added difficulty because it involves a simultaneous consideration of a larger amount of data than demanded by individual analysis of each project. It also requires proper consideration of the priorities of the various projects and of the mobility of resources.

Problems relating to time-cost trade-offs and resource allocation involve a considerable amount of data, and the procedures developed to solve them have generally been mathematically complex. It is not surprising that the techniques used have usually been computer oriented. In developing countries computers may not exist or may not be readily available at the required level. This publication, therefore, presents procedures that can be applied without computer processing. While this restricts the number of methods that can be used, it does not mean that the results obtained will necessarily be inferior to those obtained by more sophisticated approaches. Actually, the use of computers for solving time-cost trade-off and resource-allocation problems, even where an ample supply of processing equipment is available, has met with rather relatively limited success. Computer techniques cannot recognize the interaction between activity costs when changes take place. Satisfactory solutions require considerable discernment. Good judgement is needed not only in the stages of preparing data and in analysing results but also during the intermediate steps of the calculation phase. However, the exercise of such judgement during the course of calculations is very difficult to programme in mathematical form.

Project implementation has a definite beginning and a definite end, as contrasted with the cyclic type of operations characteristic of manufacturing. By its very nature the work involved in project implementation is non-repetitive. Several effective methods using network techniques have been developed during the past decade for the planning, scheduling, and control of projects. The best known of these network techniques are the Critical Path Method (CPM) and the Programme Evaluation and Review Technique (PERT). The procedures proposed in this publication are based on the principles of network methods.

As a prerequisite for the advanced network procedures for time-cost trade-offs and resource allocation, a thorough understanding of the basic network procedures is imperative. Chapter 2 briefly considers related methods for the development of a project implementation plan and its reduction to a model on paper. This involves drawing a network diagram, which is in fact a graphic portrayal of the precedence relationships between the various activities of a project from the beginning to the end of the plan. Furthermore, the methods included in Chapter 2 involve the estimation of the duration of activities and provide computational procedures for establishing basic scheduling data to determine the relative importance of each activity in the over-all project network. This is essential, since it draws the attention of management to those activities that limit or control the duration of a project. Methods for communicating the implementation plan and schedule to those who will carry them out and

methods for the application of data to project control are also included in Chapter 2.

In Chapter 3 the mechanics of the network diagram are analysed and methods for updating project data based on an understanding of the means by which changes are transmitted through the network are presented. These methods are useful for the normal updating required as unforeseen changes occur, but they are also a key factor in the development of time-cost trade-off procedures, where it is essential to update project data as a result of intentional changes in activity duration. In Chapter 4 the time-cost trade-off problem is presented in detail and procedures for solving it are suggested. Chapter 5 considers the resource-allocation problem and offers an approach to its solution. Chapter 6 discusses multi-project resource allocation and the necessary modifications of the single-project method of solution. Finally, Chapter 7 considers the interrelationships between time-cost trade-offs and resource allocation and methods for taking into account the effects of these interrelationships.

It is not the intention of this publication to stress clever mathematical manipulations or to attempt to achieve mathematically optimum solutions. The stress is rather on simplicity and practicality of application at the project level by manual methods. Solutions that are short of theoretical perfection are acceptable as long as they improve project performance.

Chapter 2

BASIC NETWORK ANALYSIS TECHNIQUES

Developing a model

After the objectives of a project have been defined it is necessary to programme the manner in which they can be achieved. For this purpose it is advantageous to consider the over-all job as consisting of a number of related but separate activities. It is not practical to attempt to work with the entire project as a single entity; subdivision is not only necessary for implementation programming but also for time estimating, cost accounting, and project control. Component activities should consist of logical subdivisions of work. Factors governing subdivision of a project into component activities are discussed later in this chapter.

Component activities normally have very definite sequential relationships with one another that must be properly considered in programming. If the project it at all complex, the programmer should not attempt to carry these relationships solely in his head. Moreover, at some point his implementation plan must be communicated to others. A strictly verbal description will result in many different interpretations and the loss of much of the detail that has been developed through careful analysis. A programmer should reduce his ideas to a model on paper in order to keep track of what he is doing and the restraints involved, and he needs the model to transmit the results of his efforts to others in a form that can be visually comprehended and referred to whenever necessary.

The type of model most commonly used to convey project implementation plans has been the bar, or Gantt, chart. This chart shows the programmer's breakdown of the project into its component activities and the scheduling developed by him for each of these activities. While the bar chart conveys the scheduling data quite effectively, its usefulness as a programming tool is limited. It does not show clearly the sequential relationships that the programmer must constantly keep in mind. It does not force the programmer to consider all the restraints that may be involved in scheduling, since it does not require him to show all the activities that must be completed before another can begin. It does not indicate whether he has considered the various prerequisite activities. Although a bar chart may technically satisfy a specified requirement to furnish a documented implementation plan, careful examination frequently shows that the plan is not sufficiently subdivided, that it totally omits many restraining activities, that it does not indicate whether careful analysis has been made, and that it

will need further interpretation to be fully understood. An implementation plan presented on a bar chart may be based on careful and masterly programming or on sloppy and incompetent programming. Those who review the paper model find it difficult to judge which is the case because insufficient detail is shown.

Figure 1 shows a very simple bar chart having only four activities. In this example it might be conjectured that Activity *B* is dependent on Activity *A* and that Activity *D* is dependent on Activity *B*, since in both cases one starts at the time when the other has been completed. It might also be suspected that Activity *C*'s performance depends on the partial completion of Activity *B*. Activity descriptions may also provide hints in actual cases, but in charts involving many bars such deductions would be much more difficult to make and much more likely to prove incorrect. There is no evidence that other relationships exist between the activities shown, nor is there any reason to believe that every activity offering a potential restraint to those shown has also been included on the chart. For example, if the reviewer thinks of another activity that might affect the starting time of Activity *C*, he has no way to know whether the programmer also considered this possibility or, if so, what was the basis for the conclusion that he reached. It is obvious that the bar chart has shortcomings as a programming aid both to programmers and to others who must review and understand the results.

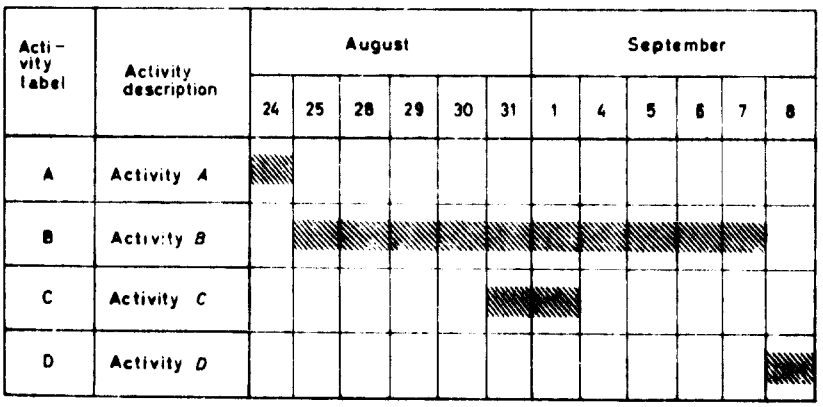


Figure 1. Simple bar chart

The network diagram has been introduced to overcome the defects of the bar chart. It is basically a programming and controlling tool and is developed prior to the determination of scheduling data. It is subsequently used to provide the relationships necessary for calculating the schedule and is sometimes plotted to a time scale to show the schedule.

The network diagram requires project activities to be well identified and their sequential relationships indicated. A network is represented mainly by arrow diagramming and precedence diagramming, although several variations of these two methods exist.

Diagramming methods

Arrow diagramming

This method is the most widely used one. It was employed in both the original Critical Path Method (CPM) and Project Evaluation and Review Technique (PERT). In this type of diagramming, a project activity, or task, is represented by an arrow. An activity requires resources such as manpower, equipment and time for its performance. It has definite starting and ending points. The terminal point of an arrow representing an activity is a node in the resulting network and represents an event, e.g., the start or finish of the activity. If one activity follows another, they share a common node where the head of the arrow representing the preceding activity is connected to the tail of the arrow representing the following activity. For example, after the land required to build factory buildings has been acquired, it may need certain preparation before construction work can start. In this case, the two activities shown in figure 2 may take place. If more than one activity precedes a following activity, the heads of the arrows representing the preceding activities merge at a node that is also the tail of the following activity; this is illustrated in figure 3. In a

Figure 2. Two activities – arrows, nodes

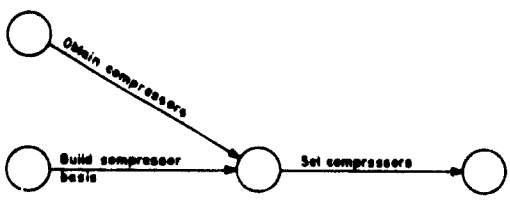


Figure 3. Merging at a node

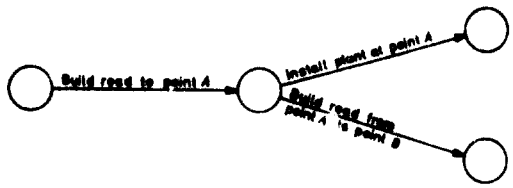


Figure 4. Arrow tails at a common node

similar way, if there is more than one following activity for a given activity, the arrows representing each of the following activities have their tails at a common node that is also the head of the arrow representing the preceding activity as shown in figure 4. In more complex cases where several activities each have

the same requirements for the completion of several activities, a number of arrows representing preceding activities may merge into a common node and a number of arrows representing following activities may burst or depart from the same node, as illustrated in figure 5. In this last case the node represents an

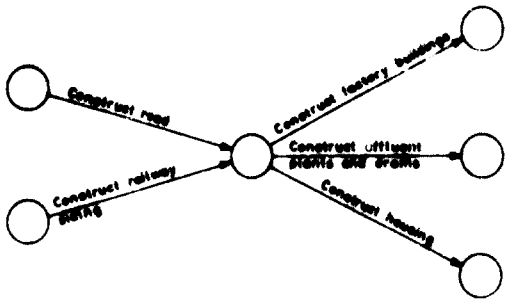


Figure 5. Several activities--common node

event that may be described as the instant at which all the preceding activities have been completed or it may be described as the instant at which all the following activities may be commenced.

Originally the Critical Path Method (CPM) used activity-labelled arrow diagramming, with description labels attached to activities or arrows, while the Project Evaluation and Review Technique (PERT) used event-labelled arrow diagramming, with description labels attached to events or nodes. This is shown in figure 6.

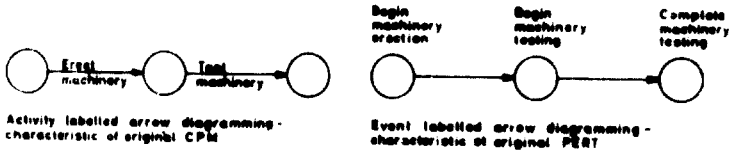


Figure 6. Original CPM and PERT

Arrow diagramming is complicated by the necessity for introducing dummy activities—activities having no physical significance and zero time durations and requiring no resources. The most common reason for using dummy activities is to show the correct sequential relationships when this cannot be done by bringing the terminals of the arrows involved together at a common node—in other words, when a number of preceding activities have one common following activity but other following activities are not common to all the preceding activities. Suppose, for example, as figure 7 indicates, Activities *A* and *B* must

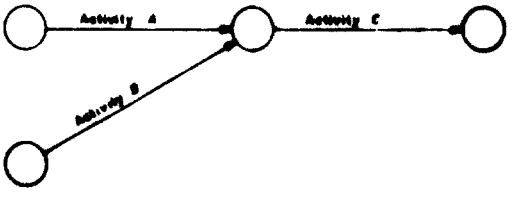


Figure 7. Example: Activities A, B and C

be completed before Activity C can be commenced. Now suppose it is necessary to add an Activity *D* that cannot be commenced until Activity *B* has been completed but which is not dependent on Activity *A* in any way. The arrow representing this activity cannot commence at the common node, since to do so would indicate a dependence on Activity *A*. The solution to this problem is to introduce an artificial, or dummy, arrow representing a zero time duration as illustrated in figure 8.

Figure 8. Use of dummy arrow

Dummy activities define precedence relationships only if such a situation occurs. A dummy activity is sometimes called a dependency arrow and is represented by a broken-line arrow. Knowing when and how to use dummy activities properly requires considerable skill in diagram construction.

Arrow diagramming uses a dual numbering system in which each activity arrow is given two numbers, one for its tail and one for its head. The tail of the activity arrow must have a number smaller than that of its head. Any activity is thus identified by two numbered events. All activities that commence from or finish in the same event have a common number and thus, arrow diagramming readily indicates the sequence of activities. In case of diagramming to time scale, activity time flows from left to right along the arrow representing it. The length of its projection on a horizontal time scale indicates its duration.

Precedence diagramming

Another approach to network diagramming uses the nodes of the network to represent activities rather than events. The lines, then, indicate the sequential relationships between activities. This type of diagramming has been referred to as "circle" or "circle-and-connecting-line" diagramming (Fondahl, 1962) and as "activity-on-node" diagramming (Moder and Phillips, 1966). More recently it has become known as "precedence diagramming" and has received increasingly favourable attention.

In precedence diagramming the relationships between Activities *A*, *B* and *C* of the previous example would be shown as in figure 9. The arrow heads on the

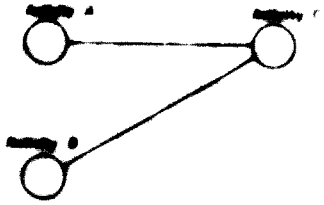


Figure 9. Work flow, left to right

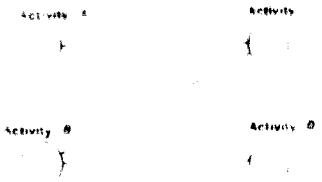


Figure 10 Appropriate sequence

sequence lines are unnecessary if the diagram is drawn with the flow of work always from left to right. When Activity *D* is added the corresponding node symbol is inserted and the appropriate sequence line is drawn as in figure 10. This is a simple and direct approach that requires no special diagramming skill. Activities are represented by a symbol, e.g. a labelled circle located at any convenient position in the diagram, with lines drawn to the preceding and following activities. There is no reason to be concerned with dummy activities. Sequence lines are actually dummy activities, but the diagrammer needs to give no particular attention to this fact.

Because of the nature of project implementation, it may be necessary to represent one or more events on the network diagram as significant milestones. These key events may indicate the commencement or completion of important activities or stages of a project (see figure 11). A key event is considered to be

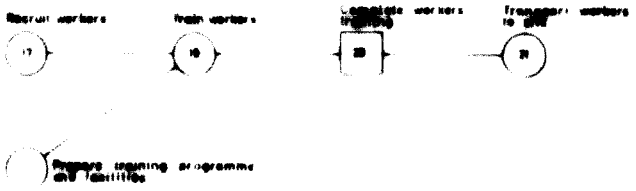


Figure 11 Precedence diagram with a key event

an activity of zero time duration. A special symbol may be used for key events, for example, a square, as shown in figure 11. Only those events having special significance need be shown.

Nevertheless, the inclusion of some events in the network diagram even if their importance does not warrant representation may facilitate and simplify the construction of the precedence diagram. This is particularly true when a number of activities cannot be commenced unless other activities have been completed. For example, in figure 12 (a) no activity in Activity Group II (Activities *D*, *E*, *F* and *G*) can be commenced until all the activities in Activity Group I (Activities *A*, *B* and *C*) have been completed, consequently there are twelve sequence lines. However, in order to simplify and improve the network diagram, the event signalling the completion of Activity Group I and the commencement of Activity Group II is represented, as shown in figure 12 (b), even if it has no significance as a milestone. The representation of events provides interface connecting points in subnetworks and is considered in Chapter 3.

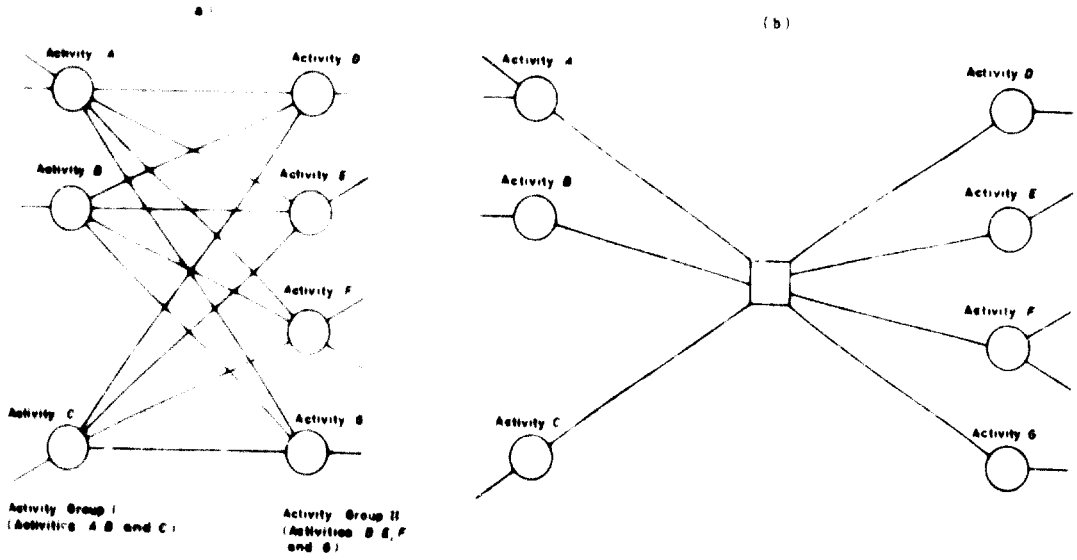


Figure 12. Precedence diagram with an event represented for simplification purposes

Precedence diagramming is much simpler to apply and to teach than arrow diagramming, and primarily for this reason it has been chosen for use in this report. If the skill required for the mechanics of diagram construction is kept to a minimum, those who have a thorough knowledge of the work to be accomplished will be able to develop the diagram themselves.

Arrow diagramming requires care in locating activity arrows at the event nodes where they merge and depart and is more difficult to revise than precedence diagramming. Adding an arrow to an arrow diagram requires changes in the positions of some arrows already drawn on the diagram. With precedence diagramming, however, adding an activity means merely placing a node in a suitable position on the diagram and connecting it with the preceding and following activity nodes by sequence lines as shown in figures 9 and 10. This facilitates updating.

The diagram is the basis for the application of all network techniques and must realistically represent the work to be performed. Otherwise even the most sophisticated procedures and processing equipment are useless. It is thus essential that only those with the best knowledge of the work develop the diagram and revise it when necessary, this means that diagramming mechanics must be kept simple.

A point of interest is that computers use the dual numbering system of arrow diagramming. Although precedence diagramming uses the single numbering system the dual numbering of sequence lines can also be used with computers.

To illustrate the precedence method of network diagramming, the project previously shown in the bar chart of figure 1 may now be represented by precedence diagramming instead. Activity *B* must be further subdivided, since overlapping is not permitted in network diagramming. (Some computer programmes appear to permit overlapping through the use of "lag factors", but actually the subdivision is still performed internally by the computer.) Activity *B1*

is a portion of Activity *B* that can be described separately and that must be completed before Activity *C* may commence. Activity *B2* is the remaining portion of Activity *B* that may be performed concurrently with Activity *C*. A new activity, Activity *X*, is also added. This represents, perhaps, an external activity, such as the furnishing by others of an item of equipment to be installed, or the checking and approval of plant drawing. Since this is work that is not directly performed by the programmer's own organization, he will frequently omit showing it on the bar chart even though te may have considered the resulting restraint. However, such an activity must be shown on the network diagram, since it requires time to accomplish and must be completed before Activity *C* can commence. Its inclusions in the diagram results in a better model on paper because it indicates to others who use the implementation plan that this activity has been considered. Although at the time the project implementation is programmed this activity may not be a controlling restraint, it may become so later because of delays and may affect the scheduling of other project activities or even of project completion.

The subdivision of the project into activities and their sequential relationships are shown below:

Activity	*Activity duration (days)*	*Must precede activity*
A	1	*B1 X*
B1	4	*B2 C*
B2	6	*D*
X	2	*C*
C	2	*D*
D	1	—

Figure 13 shows one possible precedence diagram. This diagram would serve as the model of the project and would provide the basis for the application of other network techniques. It conveys the project implementation plan more effectively than the bar chart of figure 1, but it does not convey the project schedule. For the sake of comparison, figure 14 shows three methods of presenting the same simple project shown in figures 1 and 13. Figure 14 (a) reproduces

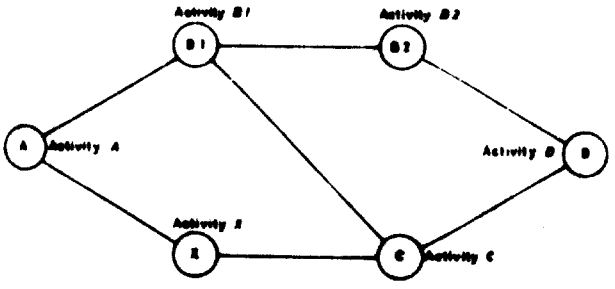


Figure 13. Precedence diagram

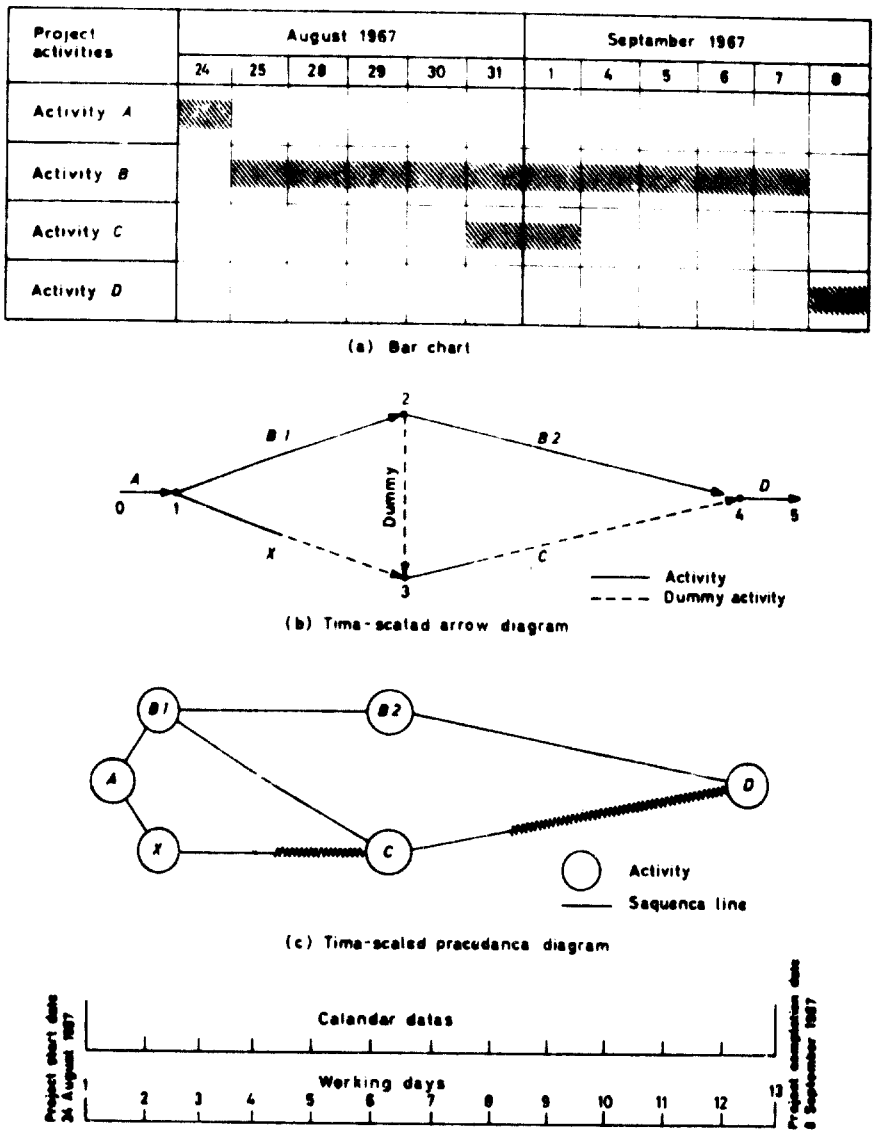


Figure 14. Comparison of presentation methods for a project implementation plan: (a) Bar chart, (b) Time-scaled arrow diagram, (c) Time-scaled precedence diagram

the bar chart of figure 1 and does not need further explanation. Figures 14 (b) and 14 (c) present the network of the project implementation plan as an arrow diagram and a precedence diagram respectively. To make the comparison more effective, both diagrams are drawn to a time scale, shown at the bottom of figure 14. The time scale, however, shows both calendar dates and working days. In figure 14 (b), Activity *A* is represented by the arrow 0—1, Activity *B1* by 1—2, etc. To show the precedence relationships between Activities *B1* and *C* illustrated in figure 13, a dummy activity is needed. This is represented by the broken-line arrow 2—3, which indicates that Activity *C* (represented by the arrow 3—4) cannot be commenced until Activity *B1* (1—2) is completed. Also, in figure 14 (b) the last portions of Activities *X* (1—3) and *C* (3—4) are a broken line. This means that these two activities are expected to be completed prior to the occurrence of the two succeeding events, events 3 and 4 respectively and,

therefore, the broken-line portion denotes the slack, or float, which each of these activities has. This is because Activity *B1* (1—2) has a duration time of 4 days, as shown in the above table, but Activity *X* (1—3) has a duration time of only 2 days; since event 3 cannot occur before the end of the fifth working day, i.e., the beginning of the sixth working day as shown in figure 14 (b), Activity *X* (1—3) has a slack, or float, of 2 days. This means that within this period the finish time of Activity *X* (1—3) can be delayed without affecting the occurrence time of event 3. In the same fashion, Activity *C* (3—4) has a float of 4 days, and within this period its finish time can be delayed without affecting the occurrence time of event 4. One more point is that these activities lie on the path 1—3—4 and the time for accomplishing them is 2 + 2 = 4 days, but from the time scale it can be seen that the period between the occurrence of event 1 and event 4 (the starting and terminating events of the path 1—3—4 respectively) is 10 days; this indicates that path 1—3—4 has 6 days of float. This path may therefore be called "float or non-critical path". On the other hand, Activities *B1* (1—2) and *B2* (2—4) have no float. They are represented by solid arrows between events 1 and 2 and between events 2 and 4, where the time for accomplishing them is 4 + 6 = 10 days, the same as the period between the occurrence of events 1 and 4. Therefore, the path 1—2—4 has no float and may be called "critical path".

The precedence diagram in figure 14 (c) illustrates the same relationships. Instead of arrows, nodes are used to represent project activities, and sequence lines are used to show precedence relationships. Event *E* denotes the completion of Activity *D* and the completion of the project as well. The zigzag portion of the sequence lines between Activities *X* and *C* and Activities *C* and *D* denotes lag times of 2 and 4 days for these two sequence lines respectively. In this particular case, these lag times equal the float of the preceding Activities *X* and *C* respectively; and the path *X*—*C*—*D* may be called "float or non-critical path", since it has again 6 days of float. The other sequence lines *B1*—*B2* and *B2*—*D* are solid, denoting that the path *B1*—*B2*—*D* has no float and hence may be referred to as "critical path". All these terms are discussed in detail in Chapter 3.

Development of the network diagram for a project implementation plan

A basic step for programming the implementation of a project is to construct an original network diagram for project activities including their interrelationships. As programming of implementation progresses, the original network diagram can be developed. This will assure that all project activities and their sequential relationships have been considered and are represented.

To start this phase of implementation programming the following steps should be considered:

Preparation of a list of project activities

All project activities should be listed as they come to mind. This should be a help to beginners, but may not be necessary after experience in diagramming

has been acquired. However, for complex projects with a large number of activities, an activity list may be of importance. When such a list is being prepared, a decision must always be taken as to the level of breakdown of project activities. This depends on the following factors:

Nature of the work involved
Project activities or activity groups need different types of resources (labour, equipment, etc.) for their accomplishment. For example, aggregate activities such as "Finalization of project plans", "Construction of buildings" and "Installation of machines and equipment" are carried out by different types of labour and machinery, and hence they can be considered separately.

Place and time of work
Work undertaken at different locations or at different times may be considered as separate activities.

Supervision and responsibility for work
If supervisors or key personnel who can assume responsibility for some parts of a project are scarce, activities may be aggregated so that each person is assigned one or more activity groups. If activities are to be aggregated, such factors as the nature of the work, location, interdependency, and duration should be considered. Work undertaken by different departments, contractors or subcontractors may be considered separately.

Method of financing the project
The breakdown of project activities is sometimes determined by the way in which a project is financed; it may be essential to facilitate financial control. This is true when a project is financed by several agencies, such as a development bank, a technical assistance institution of a foreign government, or an international organization, each of which finances one or more stages or activity groups.

Construction of an original project network diagram

Using a sheet of paper, a diagrammer can now develop the original project network diagram. Initial project activities are determined, and each is represented by a node at the left hand side of the paper; these nodes are then labelled. As each activity is represented on the diagram, the questions arise: "What activities must be completed before this activity can begin?" and "What activities may begin when this activity is completed?" As these questions are answered, the corresponding activities are added to the diagram, labelled and connected with other activity nodes with sequence lines so that the appropriate relationships are shown. This procedure is repeated until the last project activity has been entered on the network diagram. Since network diagramming requires each activity shown to be completed before the following activity can commence, it requires a further breakdown of overlapping work. For instance, when an activity can commence before a preceding activity has been completed, the earlier work is to be subdivided into an activity that represents the portion of the total

that must be completed before the later work can start and the portion that can be performed concurrently. Network diagramming is thus more difficult to construct than the bar chart. But the detailed subdivision of the network diagram means that the breakdown will not be too gross to be helpful, and it prevents careless overlapping of activities. The extra work involved in subdividing is justified, since it both allows and forces the programmers to do a better job, and the results convey the implementation plan much more clearly and effectively than would otherwise be the case.

A second procedure for developing the network diagram may be followed. Diagrammers start by entering the last project activity on the right side of the paper, label it and work backwards, adding the activities that must be completed before the commencement of the activity just entered. This procedure continues until initial project activities have been entered. Here again appropriate sequence lines between nodes are drawn as the network diagram is developed. During the construction of the diagram arrow heads are put on sequence lines to show precedence relationships.

Although the first procedure is more common, a diagrammer can choose the method he prefers. Diagrammers should present a project implementation plan network in such a way that others engaged in the project understand it and see clearly the sequential relationships of the activities.

Figure 15 gives an example of an original network diagram of a project implementation plan that is condensed. The work breakdown structure illustrated by this figure shows "work packages" as the nodes of the diagram. A work package is a group of activities for which an individual or an organizational unit is responsible. For reliable day-to-day decisions and for effective programming and control of project implementation, each work package should be broken down into a subnetwork of detailed activities. As can be seen, the original network diagram does not present the project implementation plan clearly, since it has for the most part been drawn free-hand and the positions of the nodes do not follow any logical grouping. Some nodes have been located in such a way that a few of the sequence lines connecting them with other nodes run from right to left instead of from left to right, the logical direction of the flow of work. For this reason arrow heads are put on sequence lines during the construction of the original project network diagram to show precedence relationships. Also, some sequence lines may be cancelled and thus appear crossed out on the diagram as shown in figure 15. Nevertheless, the original network diagram does provide a useful work sheet for programmers and for those interested in the project. It provides the basis for the original scheduling computations.

Redrawing of the network diagram of a project implementation plan

A clearer and better presentation of a project implementation plan than the original network diagram is required in order to: *(a)* evaluate the plan more quickly and make the necessary adjustments; *(b)* obtain more easily data and information from the network diagram for further computations or periodic

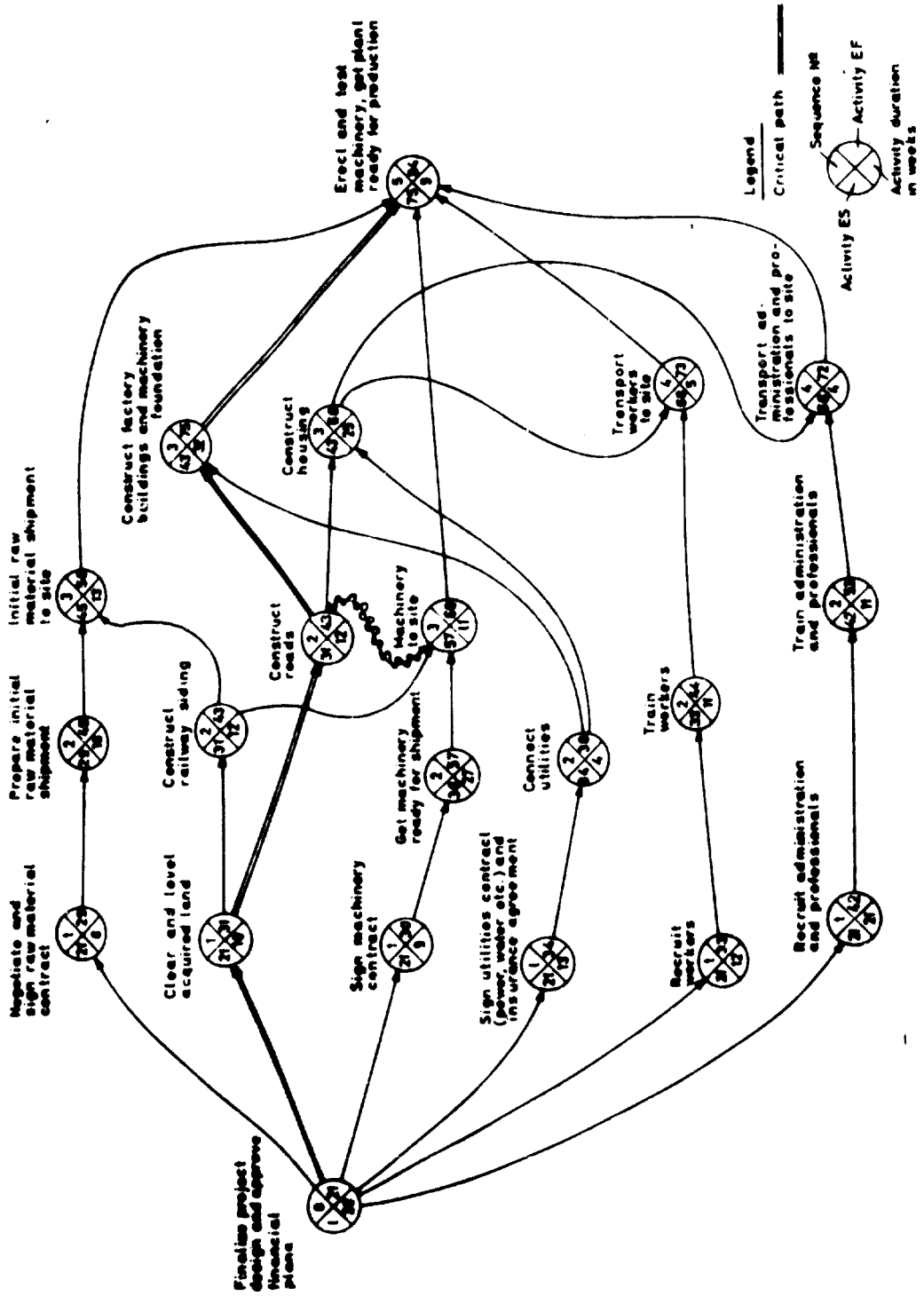


Figure 15. Original network diagram of a project implementation plan

revision; *(c)* facilitate updating; *(d)* make necessary computations on the network diagram; *(e)* provide a common language for those engaged in project implementation.

It is advantageous to postpone the redrawing of the original network diagram until it has been proved satisfactory. To find out whether the implementation plan meets requirements, the initial time computations based on activity duration estimates may be entered on the original diagram. Here programmers need do no more than make the "forward pass" of basic time computations starting from the beginning and working to the end of the project where the earliest start and earliest finish times of project activities are calculated and hence the earliest project completion date. The earliest start time of an activity is the latest or greatest of the earliest finish times of all activities preceding it, and the earliest finish time of an activity is its earliest start time plus the activity duration time. Then, to determine the critical activities and hence critical path(s) without computing activity latest finish and latest start times and total floats, one can start from the last project activity and go backward to the initial activities along the sequence lines connecting those activity nodes of equal earliest start and earliest finish times. For example, in figure 15, starting from the last project activity and going backward to the initial activity(ies) as mentioned above, it can be seen that the activities "Erect and test machinery", "Get plant ready for production", "Construct factory buildings and machinery foundation", "Construct roads", "Clear and level acquired land" and "Finalize project designs and approve financial plans" are critical and hence the path along which they lie.[1] If the project duration is now found to meet requirements, and if the activity sequential relationships included in the implementation plan (mainly those concerning the critical activities) are found acceptable, then the original project network diagram may be redrawn. On the other hand, if the project duration does not meet requirements the project implementation plan should be reconsidered. Reprogramming of project implementation may be indispensable, and hence certain changes in the network diagram may be incorporated. Consideration of the critical activities may suggest changes in activity breakdown, i.e., level of detail of project activities presented on the diagram, so that some activities, if possible, may be overlapped. After these changes have been made and the project duration and the implementation plan have been reviewed and accepted, the network diagram can be redrawn using sequence-step or time-scale diagramming methods.

Sequence-step procedure

A suitable scale of sequence steps can be chosen and vertical lines drawn at each sequence step on this scale. The size or length of a sequence step is arbitrary. It should be selected to provide a graphic portrayal of the project network diagram. After the sequence steps and hence these vertical lines have been numbered, nodes representing activities (and perhaps key events) can be located on the vertical lines according to their sequence-step numbers. Each node is placed horizontally to the right of activities preceding it. Next to each node is a brief description of the activity it represents. As shown in figure 16, the sequence

[1] Basic scheduling computations included in figures are discussed in detail below.

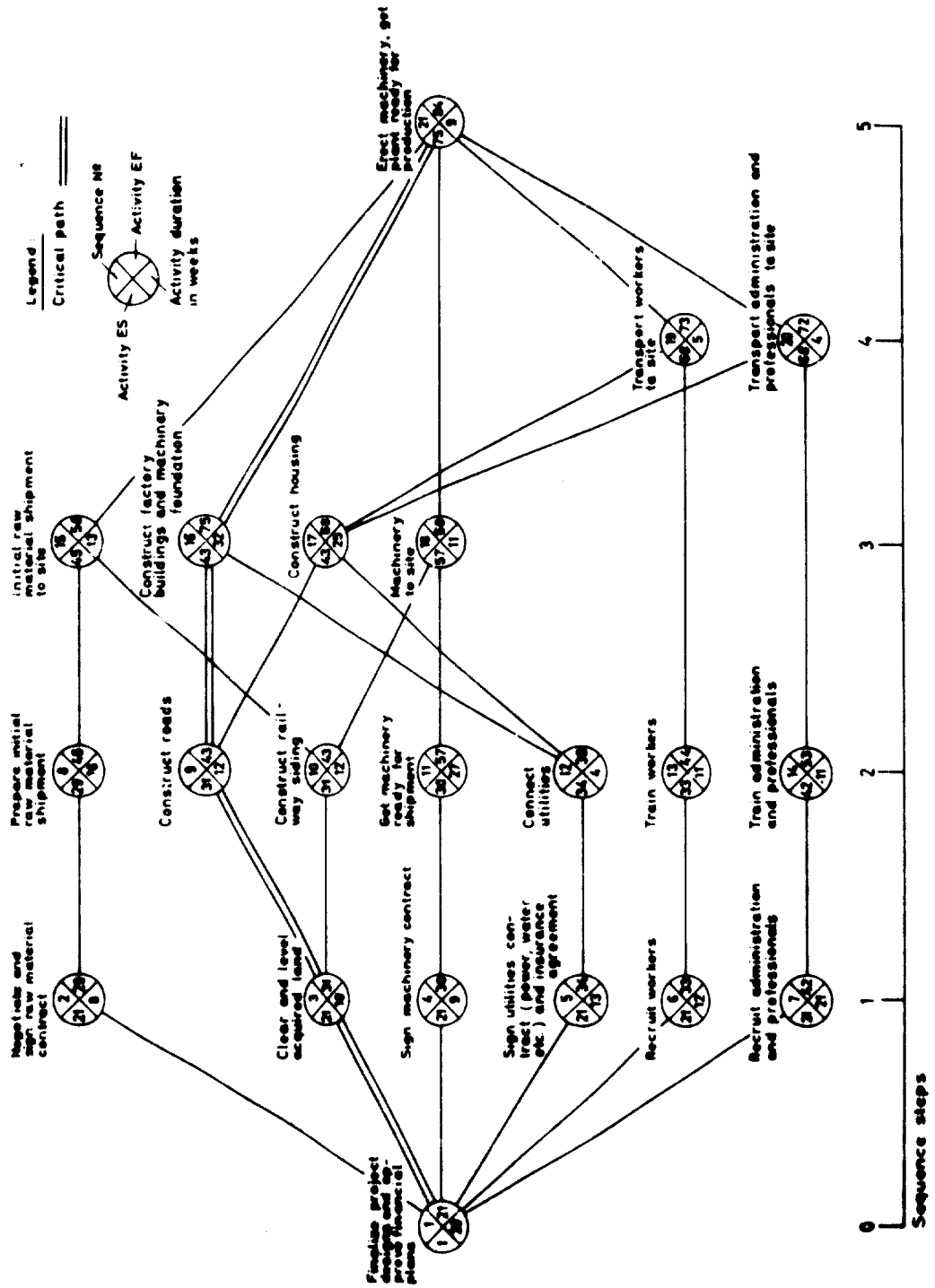


Figure 16. Sequence-step precedence diagram

step of each activity is determined by giving initial activities a sequence step number of zero, plotted on the corresponding vertical line. Then, each other activity is given a sequence-step number that is one greater than the highest sequence-step number directly preceding it, and so on. The vertical position of the nodes on the network diagram should be chosen so as to achieve logical

groupings and to enable sequence lines to be clearly drawn. The nodes are then numbered. Numbering starts from the top of the first sequence vertical line by giving the node in the upper left hand corner of the diagram a number of one, then proceeding downward along that vertical line giving the second node a number of two, and so on. Numbering then proceeds to the top of the second line of activities and goes down that line continuing this way until the farthest node to the right of the network diagram is reached. Thus, no activity will precede another activity of a lower number. After all necessary computations and revisions have been made, this type of diagram provides an effective step towards a final diagram or time-scale diagram. Sometimes the sequence-step diagram may be considered as the final diagram, and activity duration as well as other basic scheduling data may be included next to or inscribed into corresponding nodes, as illustrated in figure 16. This depends on the need of the diagrammer and those who will use the diagram and on the degree of accuracy with which activity relationships must be represented.

Plotting the network diagram on a time scale

The network diagram can be plotted on a time scale from the original or from the sequence-step network diagram. The need for a time-scale diagram arises because neither the original rough diagram nor the sequence-step diagram represents project activities in their appropriate time relationships. The sequence-step network may show two or more activities with the same sequence number (located on the same vertical sequence line) that in reality are performed at different times. If a network diagram is plotted to a time scale it shows the real activity interdependencies, at least at the beginning of project implementation. The sequence-step and the time-scale diagrams can be used together, in this case, a sequence-step diagram is constructed for the entire implementation plan. At each time interval (one or two months) a detailed time-scale diagram can be prepared for the portion of the project that is to be implemented during the next time interval. The selection of the time interval depends on the nature of the project, the degree of detail required, and the complexity of the work. When conditions change, the updating of network diagrams constructed to time scale requires a great deal of work. Therefore, in the case of large, complex projects, especially when non-computer methods are used, updating the entire network diagram is not to be recommended. The whole diagram may be revised once, and at each time interval during project implementation a time-scale network diagram of the portion of the project network that will be undertaken in the following time interval can be revised and updated. Revision of diagrams plotted to time scales may be undertaken only when sequential relationships of activities change or some breakdown of activities is modified.

When the project network diagram as shown in figure 17 is plotted, a time scale is drawn. At each activity's earliest (or scheduled) start time, a vertical line is drawn on the time scale on which the node representing this activity is located. If a sequence line has a lag time (see Chapter 3) it is shown on the sequence line by a corresponding zigzag portion.

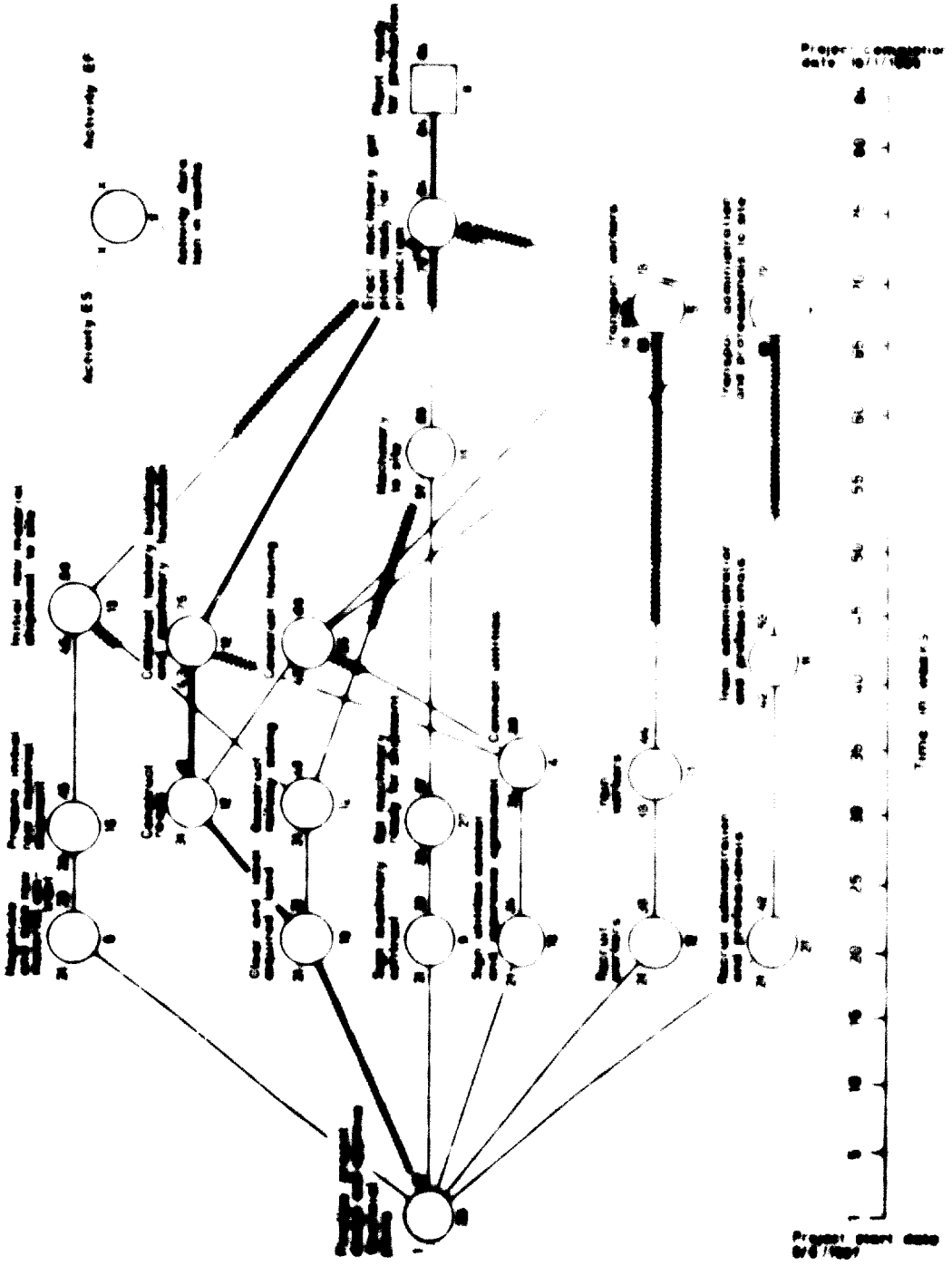


Figure 1.7 Time-scaled precedence diagram

One of the advantages of constructing a network diagram to a time scale is that if a number of vertical lines are drawn on the diagram at various time intervals, it is possible to determine at any point in time what activities should have been performed, what activities are being performed, and what activities are to be performed next. Since the vertical lines on the diagram show what activities are concurrently being undertaken, time periods with excessive demand

in the use of more key resources can be readily seen. This may suggest some quick improvements in resource scheduling even before more effective resource-allocation techniques are applied, such as shifting certain activities along the diagram so that more levelling of resource utilization can be achieved.

It should be emphasized that the construction of the network diagram is of prime importance for the programming and control of project implementation and is the basis for applying the advanced network procedures for time–cost trade-offs and resource allocation.

Developing Project Scheduling Data

Once a realistic network diagram has been constructed for the project, the next step is to introduce time data by making estimates of the duration of each activity. These estimates may be made in any appropriate units, such as months, weeks, days or hours. Any unit can be used so long as it is used for all activities. When days are used there must be a decision whether these are calendar or working days. Use of the latter is more common. It requires the conversion of durations that are generally quoted in calendar days, such as delivery times or concrete-curing periods, to a corresponding number of working days. In general, owing to the non-working days of weekends and holidays, and days on which work could not be undertaken because of storms, rains etc., time could be lost, and hence the elapsed calendar days would be more than the number of actual working days. In construction work, for example, rainy seasons and storms can delay or interrupt work. If a construction activity is interrupted by a rainy season for 4 weeks and its original duration was estimated at 12 weeks without weather consideration, the duration should be adjusted to 16 weeks (12 + 4). Other activities, such as delivery of machines and equipment, are usually not affected by climatic factors.

All scheduling calculations depend on activity durations. Estimates of activity durations should be based on a thorough understanding and knowledge of the work to be performed and the techniques that may be used. Good results in network diagramming are obtained only through the exercise of competent judgement.

During the course of project implementation the need may arise to change activity duration. Changing the technique or the amount of resources used to carry out an activity always changes its duration, as for instance, when more resources are allocated to an activity or when the estimated duration is found to be inaccurate because of an inadequate understanding of the work involved and/or some influencing factors at the time the estimation was made.

If the type of work is one in which a reasonable amount of experience has been gained, single time estimates for each activity are the most practical and simple. The approach followed by the Project Evaluation and Review Technique permitted three time estimates for each activity: a most likely duration, an optimistic forecast, and a pessimistic forecast. Much of the missile programme work that PERT was originally designed to control involved research and de-

velopment and the manufacture of components never built before. The personnel involved were understandably reluctant to provide a single time estimate for the performance of their activities. Therefore, a weighing formula was developed to convert the three obtainable time estimates to a single, statistically equivalent time. The activity duration, then, was applied in the same manner as if a single time estimate had been made. This publication will assume single time estimates, a method followed by the Critical Path Method. There may be cases involving engineering design work, for example, where similar reluctance to provide such estimates is encountered. If so, a standard text on PERT will provide precise definitions for each of the multiple time estimates and will give the weighing formula.

After activity durations have been furnished, whether they result from single time estimates or have been calculated from weighted multiple estimates, certain routine scheduling computations can be made. These generally furnish six items of data for each activity: earliest or early start time, earliest or early finish time, latest or late (allowable) start time, latest or late (allowable) finish time, total float, and free float.

Rules governing the computation of the above-mentioned scheduling data are based on network logic and may be summarized as follows:

(a) *Activity earliest start time* (ES) is equal to the latest or largest of earliest finish times of the activities preceding it.

(b) *Activity earliest finish time* (EF) is equal to the earliest start time of an activity plus its duration time.

(c) *Activity latest finish time* (LF) is equal to the earliest or smallest of the latest start times of the activities following it.

(d) *Activity latest start time* (LS) is equal to the latest finish time of an activity less its duration time.

(e) *Activity total float* (TF) is the difference between the earliest finish time and the latest finish time of an activity (or the difference between the earliest start time and the latest start time of an activity). Activity total float time gives the amount of time (number of days, for example) by which the finish time of an activity can exceed its earliest finish time without affecting the over-all project duration. In other words, it is a measure of the extra time or leeway available for the performance of an activity without causing the project duration to be extended.

(f) *Activity free float* (FF) is the difference between the earliest finish time of an activity and the earliest or the smallest of the earliest start times of the activities following it. Activity free float gives the amount of time (number of days, for example) by which the finish time of an activity can exceed its earliest finish time without affecting the earliest start time of any other activity. In other words, it is a measure of the extra time or leeway available for the performance of an activity without causing the delay of any other activity.

Based on the network diagram shown in figure 13, and by referring to the activity durations already estimated and included again below, the scheduling computations can be made.

Activity	*Duration (days)*
A	1
B1	4
B2	6
X	2
C	2
D	1

The first calculations determine the earliest start and finish dates for each activity and are often referred to as the "forward pass". These calculations can be performed on a separate tabulation with reference to the network diagram for the sequential relationships, or they can be performed directly on the diagram as shown in figure 18.

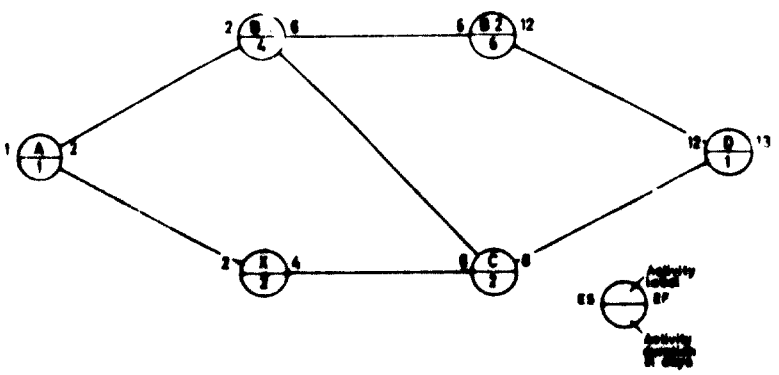


Figure 18. Forward-pass calculations

Starting at the beginning of the project, the earliest start time for Activity *A* is 1, the beginning of the first day. Adding its duration of 1 day, its earliest finish time is 1 + 1 = 2, the beginning of the second day (or the end of the first day). Since the diagram indicates that Activities *B1* and *X* can commence after Activity *A* is completed, their earliest start dates are 2, the beginning of the second day. This process is continued through the network until the final activity is completed. Where an activity follows more than one preceding activity, its earliest start time is determined by the earliest finish date of the preceding activity that is completed the latest. This is the case with Activities *C* and *D*. For example, Activity *C* follows both Activities *B1* and *X*. The earliest finish times or dates of these two activities are 6 (the beginning of the sixth day) and 4 (the beginning of the fourth day) respectively. Consequently, the earliest start date of Activity *C* is 6. For Activity *D*, the preceding activities are Activities *B2* and *C*, with earliest finish dates of 12 and 8 respectively. Thus, the earliest start date of Activity *D* is 12, the beginning of the twelfth day. Besides providing the earliest dates at which each activity can be started and completed, the forward-pass calculations also provide the project duration. This is set to be the earliest completion time of the final activity and in this case is 12 days, since the com-

pletion date of Activity *D* is the beginning of the thirteenth day (or the end of the twelfth day).

With the project duration held fixed, the latest finish time for the final activity is set equal to its earliest finish time. Then the "backward pass" is performed to obtain the latest start and finish dates of the remaining activities. The latest finish time of Activity *D* is therefore 13, the beginning of the thirteenth day. The latest start time of Activity *D* is 13 − 1 = 12, i.e., the latest finish time of the activity less its duration. Since the diagram indicates that Activities *B2* and *C* must be completed before Activity *D* can commence, their latest finish dates are each 12. This process is continued back to the beginning of the project. If an activity precedes more than one other activity, its latest finish date depends on the following activity whose latest start date occurs earliest. This is the case with Activities *B1* and *A*. Activity *B1* precedes Activities *B2* and *C*, with latest start dates of 6 and 10 respectively. Therefore, the latest finish date of Activity *B1* is 6. In the same way, the latest finish date of Activity *A* is 2. The backward pass calculations are shown in figure 19.

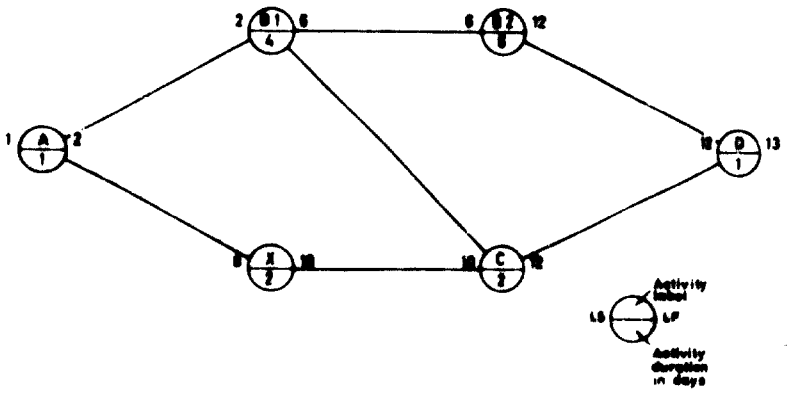


Figure 19. Backward-pass calculations

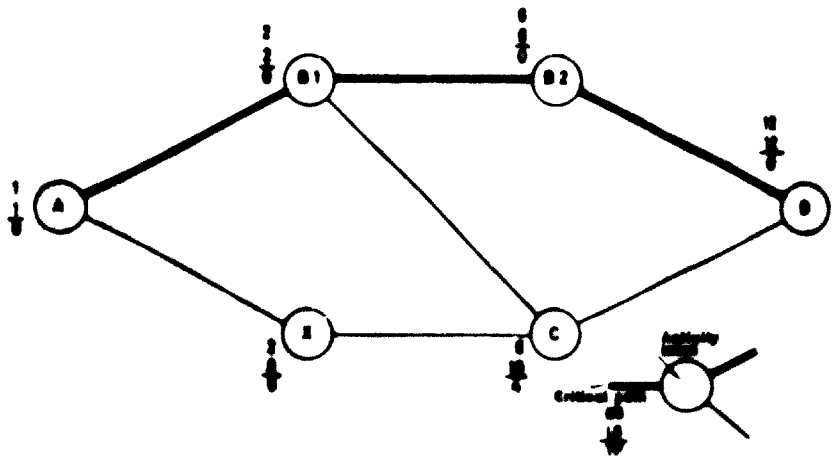


Figure 20. Activity total float calculations

Figure 20 shows activity total float calculations. For example, Activity C has a total float of 4 days, which is equal to 10 minus 6 days (activity latest start

time minus activity earliest start time). This is obvious, since the earliest that Activity C can be commenced is the beginning of the sixth day, and the latest that it can be commenced without affecting the project completion time is the beginning of the tenth day; therefore, it has a total float of 4 days. Activities having zero total float have no scheduling leeway; they must be performed at the earliest possible time in order not to delay project completion. Any delay in performing a critical activity will result in a corresponding delay in project completion time. In every network there are one or more chains of critical activities extending from the beginning to the end of the network that determine project duration. Such a chain is referred to as a "critical path". In this example it consists of Activities *A*, *B1*, *B2* and *D*. Although in this network two thirds of the total activities are critical, a more common situation in large networks is that only a small proportion, perhaps 10 to 20 per cent, of the activities are critical.

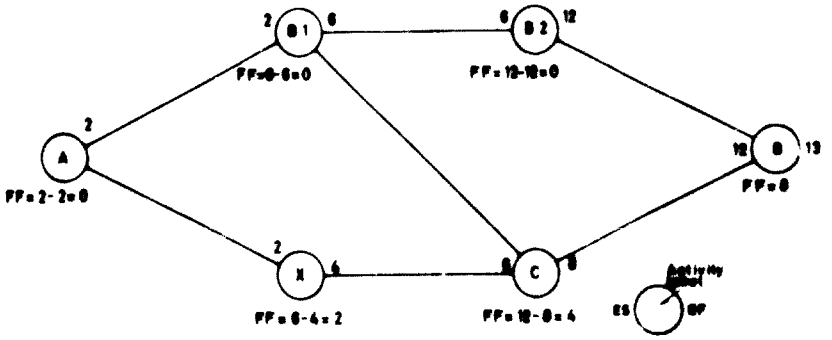


Figure 21. Activity free float calculations

Free float calculations are shown in figure 21. Activity *X* has a free float of 2 days. This indicates that its completion could be delayed as much as 2 days without affecting any other activity. Its total float of 6 days indicates that it can be delayed another 4 days without affecting project duration. Such a delay would require that Activity *C* be postponed, which will affect its total float. This four-day period, the difference between total and free float of an activity, will be referred to as "interfering float", since it involves interference with the scheduling of other activities.

Since the scheduling computations just described are purely mechanical, and there may be many of them to perform in a large network, an electronic computer could be useful. On the other hand, these calculations are extremely simple, requiring nothing more than adding or subtracting two numbers at a time, and can easily be performed manually if computer equipment is not available.

COMMUNICATING THE IMPLEMENTATION PLAN AND SCHEDULE

After a satisfactory implementation plan and schedule have been developed, it is generally necessary to communicate this information to those who review the project implementation programming or who participate in executing and

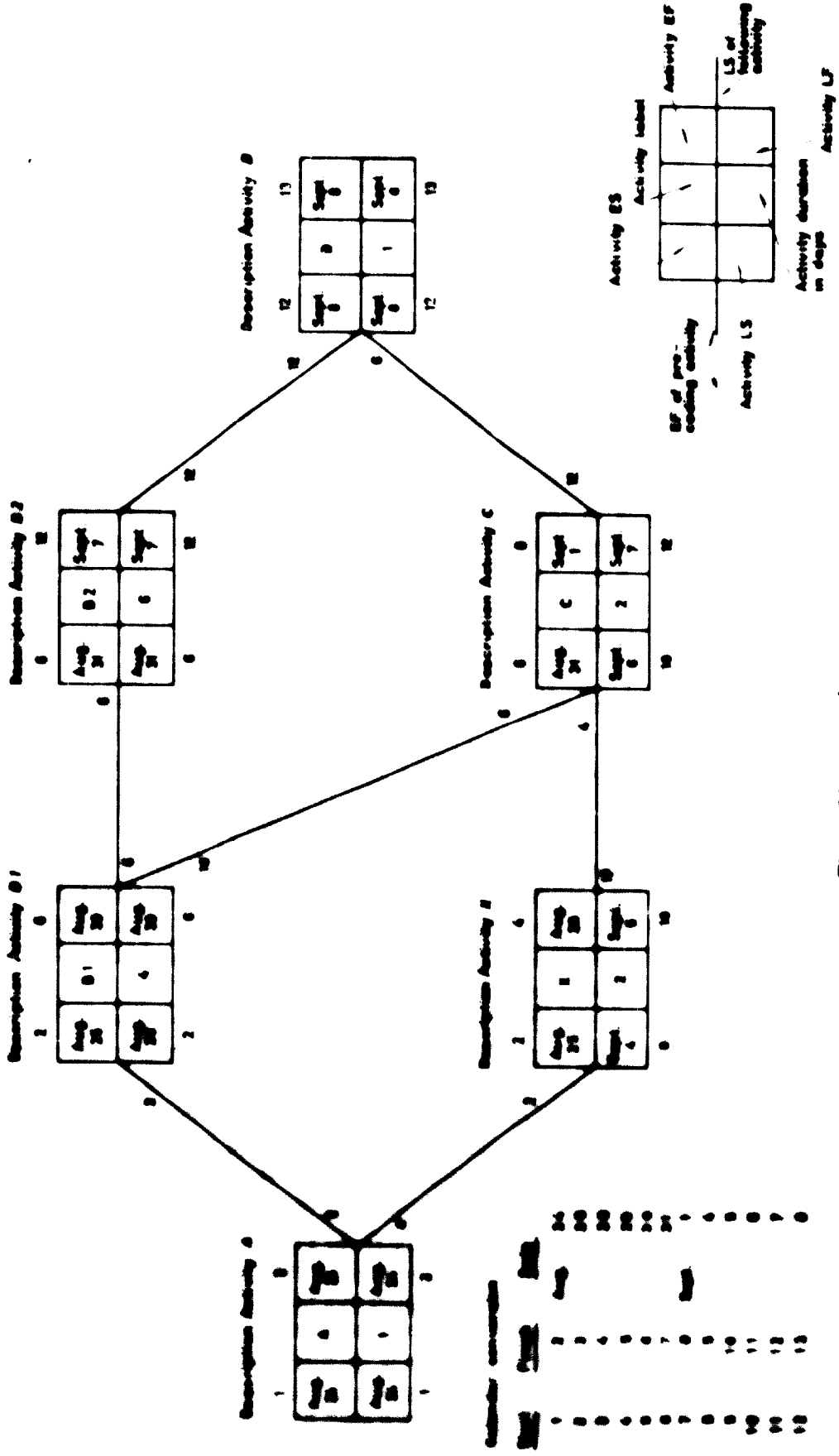


Figure 22. Network presentation

controlling the work. There are a number of ways in which this can be done; three methods are briefly described below.

One method is shown in figure 22; here all information is developed and shown directly on the diagram. Each node symbol is divided into compartments for showing the activity label, duration, earliest start time, earliest finish time, latest start time, and latest finish time. Time data are initially calculated in undated form. The earliest finish time of each activity is transferred to the far end of its following sequence lines and is put above the lines. The earliest start time of an activity is the largest of the numbers on the incoming terminals of its preceding sequence lines. In figure 22, the numbers 6 and 4 are written above the adjacent ends of the sequence lines connecting Activity *C* to Activities *B1* and *X* respectively. Six is the earliest finish time of Activity *B1* whereas 4 is the earliest finish time of Activity *X*. Hence, the earliest start time of Activity *C* is 6. Earliest start and finish times of an activity are temporarily entered above their respective compartments. On the backward pass, the latest start times are transferred to the far end of the preceding sequence lines and put below the lines. The latest finish time of an activity is equal to the smallest of the numbers on the adjacent terminals of the following sequence lines. Again in figure 22, the numbers 6 and 10 are put below the following sequence lines connecting Activity *B1* to Activities *B2* and *C*. Six and 10 are the latest start times of Activities *B2* and *C* respectively; therefore, the latest finish time of Activity *B1* is 6. Latest start and finish times are temporarily entered below their respective compartments. Finally, using a conversion table shown on the diagram, undated start and finish times are converted to calendar dates and entered into the appropriate node compartments.[2] This single diagram, then, shows both the project implementation plan and the scheduling data. The total float or free float of any activity can be readily obtained by subtracting the appropriate undated entries from one another. An additional possibility is to plot the diagram to a time scale so that the time relationships as well as the logical relationships of the activities are shown. This can be accomplished effectively by plotting each activity node at a position corresponding to its earliest start time with respect to a horizontal time scale as previously mentioned.

A second method of presentation is that of the bar chart (figure 23). The programmer has used a network diagram, and therefore the breakdown of the bar chart is the same as that required by the network. Sequencing relationships are given by listing the labels of preceding activities at the beginning of each bar and the labels of following activities at the end of each bar. Total, free, and interfering float periods are shown as extensions of the basic bar that shows the duration of an activity. As in the first system, all programming and scheduling information as well as float data are available from a single drawing. This system has the not unimportant advantage that it presents the information in a form familiar to most people and therefore requires the least adjustment to new concepts.

[2] Calendar dates allow for weekends, in this example five working days per week have been assumed.

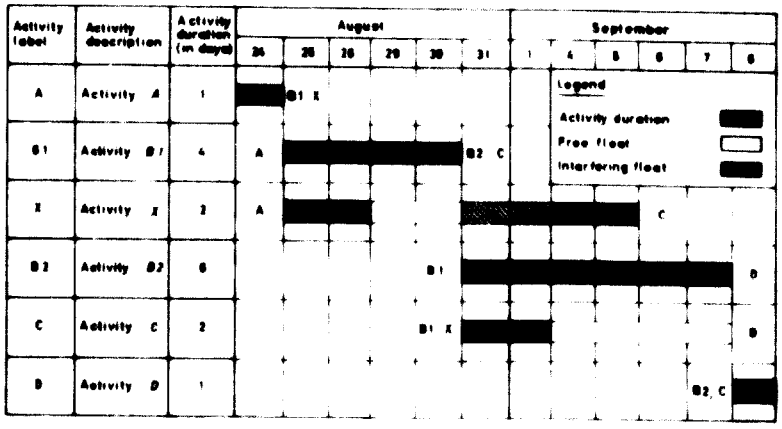


Figure 23. Bar-chart presentation

The third method is to provide the programming information and the scheduling information separately. A network diagram in its simplest form (figure 13) shows the project implementation plan. Scheduling data can be presented in the form of a printed tabulation. This is the common approach when computer processing is used, but it is equally applicable with non-computer processing and listing. Figure 24 provides a tabulation that could be used in

Activity label	Activity description	Activity duration (in days)	Critical activities	ES	EF	LS	LF	TF	FF
A	Activity A	1	*	Aug 24	Aug 24	Aug 24	Aug 24	0	0
B1	Activity B1	4	0	Aug 25	Aug 30	Aug 25	Aug 30	0	0
X	Activity X	2		Aug 28	Aug 28	Sept 4	Sept 5	6	2
B2	Activity B2	6	*	Aug 31	Sept 7	Aug 31	Sept 7	0	0
C	Activity C	2		Aug 31	Sept 1	Sept 6	Sept 7	4	4
D	Activity D	1	*	Sept 8	Sept 8	Sept 8	Sept 8	0	0

Figure 24. Schedule tabulation

conjunction with figure 13. It would be updated following variations in time data, and the diagram would be updated following the less frequent variations in sequencing data. The scheduling information might also be conveyed in the form of a bar chart, omitting the sequencing information described in the second method. In either case, both implementation programming and scheduling information might be required by some people, while only the scheduling information would be required by others. It is often useful to carry this a step further and transmit certain scheduling information only to those who can use it. For example, a subcontractor or supplier unacquainted with the master programming of a project is not in a position to judge whether he can use interfering float time or not. While it may be helpful to provide him with free float data so that he will have as much scheduling freedom as practical, it may not

be wise to reveal total float data or latest start and finish dates. This third method of communicating programming and scheduling data provides the maximum degree of flexibility.

CONTROLLING THE PROJECT

It is not enough to develop a good implementation plan and schedule for the execution of a project, there must be a strong effort to make them work. When they fail to work, there must be effective corrective action to minimize possible adverse effects. Network techniques provide more useful information than any other method for the control of the project type work.

Good project control requires knowing where to concentrate management effort to make the plan succeed, the degree of seriousness of changes that do occur or of those that are proposed, and what corrective action will be most effective after a change occurs.

Network techniques indicate where to concentrate management effort by providing float data. Activities with zero total float are critical and must be held on schedule if the project completion data is to be realized. Activities that are "near-critical", having very little total float time, need also to be watched carefully, since they will affect project duration after a slight delay in schedule. Activities having a comfortable margin of free float are at the other extreme and require the least amount of attention. If resource restrictions have made it advisable to schedule activities at certain dates, these activities must be watched regardless of the amount of float indicated. The resource schedules should provide management with an awareness of the importance of maintaining such scheduling.

Network techniques give a clear indication of the seriousness of changes that do occur or that may be proposed. If completion of a critical activity is delayed or if completion of a non-critical activity is extended beyond its latest finish date, project completion will be similarly delayed unless corrective action is taken. If completion of a non-critical activity is delayed within its free float range, neither project completion nor the scheduling of other activities is adversely affected (unless resource problems are created). If completion of a non-critical activity is delayed to a point within its interfering float range, a problem may or may not exist, further investigation is required. Project completion is not directly changed, but other activities must be postponed. Since these activities are delayed within their float periods, the effect is often unimportant. In certain cases, however, such postponements may be just as serious as delays of critical activities. For example, if an activity to be performed by a subcontractor is not permitted to begin on schedule, additional delays may result owing to previous commitments of the subcontractor at the time that the activity can commence. The total effect may be a considerably longer delay extending well beyond float ranges and causing an increase in project duration. Since network techniques provide the means for determining which other activities are affected when one is delayed within its interfering float range, such effects as that just described can be anticipated by management. Also, the effects on resource schedules may be forecast if such schedules have been developed.

Finally, network techniques help to determine the appropriate corrective action. They allow the scheduling of all activities to be properly updated. If project duration has been extended and it is desired to re-establish the previous completion date, the current critical activities have been determined. Project management knows which activities will shorten project duration if they are expedited. If cost data are available, the time-cost trade-off technique will indicate the most economical means to accomplish this expediting. If project management wishes to consider new implementation programming approaches, the network can be revised and the corresponding schedule developed. If resource data are available, resource allocation and levelling procedures may be used to achieve a new schedule that again satisfies resource restraints.

Chapter 3

NETWORK MECHANICS WITH APPLICATIONS TO UPDATING AND DEVELOPMENT OF SUBNETWORKS

Introduction

The representation of a project implementation plan by a network, indicating the component activities and their sequential relationships, is a fundamental concept on which all other network techniques depend. Some of these techniques, such as schedule updating and time-cost trade-offs, are dynamic ones. They involve making changes in the data and determining the effects of these changes on the remainder of the network. An understanding of basic network mechanics is a prerequisite for developing procedures for these dynamic applications. To explain basic network mechanics is the principal objective of this chapter.

One of the limitations of manual application is network size. A possible approach to working successfully with larger networks is to subdivide the overall, or master, network into smaller ones. This must be done with care in order that the important relationships and functioning of the master network are properly represented by the subnetworks. Such representation is facilitated by an understanding of network mechanics.

Lag values

In the precedence diagram the sequence lines between nodes show sequential relationships between the corresponding activities. A useful concept that has been proposed is the association of a time quantity called a "lag" value with each sequence line (Fondahl, 1962). In many cases the following activity commences as soon as the preceding activity has been completed. In such cases the lag value is zero. In the remaining cases the commencement of the following activity is delayed. The usual reason for this delay is that the commencement of the following activity is determined by some other preceding activity that has a later completion date. Sometimes the delay may be one that has been intentionally scheduled to satisfy some restraint, such as a wait for resources; in these cases there is a positive-lag value. Its magnitude is determined by subtracting the earliest finish time or scheduled finish time, whichever is later, of the preceding activity from the earliest start time or scheduled start time, whichever is later, of the following activity. A lag value cannot be negative, since according to the

rules of network diagramming a following activity cannot commence until all of its preceding activities have been completed.

Transmission of changes

The lag value indicates an important characteristic of the sequence line. Sequence lines with zero-lag values serve to transmit the effects of changes to the remainder of the network. Sequence lines with positive-lag values are inoperative in communicating changes. Figure 25 shows a sample network. Earliest

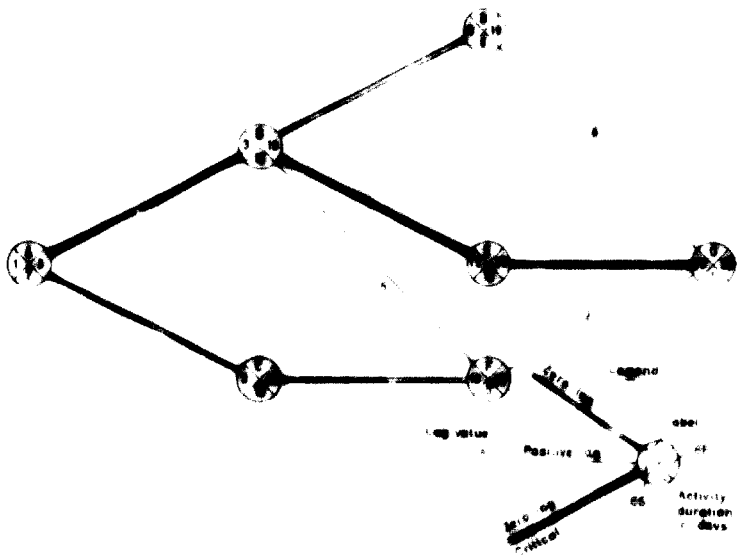


Figure 25 Sample network

start and finish times have been calculated for each activity and used to determine the lag values for each sequence line, as shown. If Activity *B* takes a day longer to perform than its estimated duration, its earliest finish time will increase by one day, as shown in figure 26. This change is conveyed to all following activities

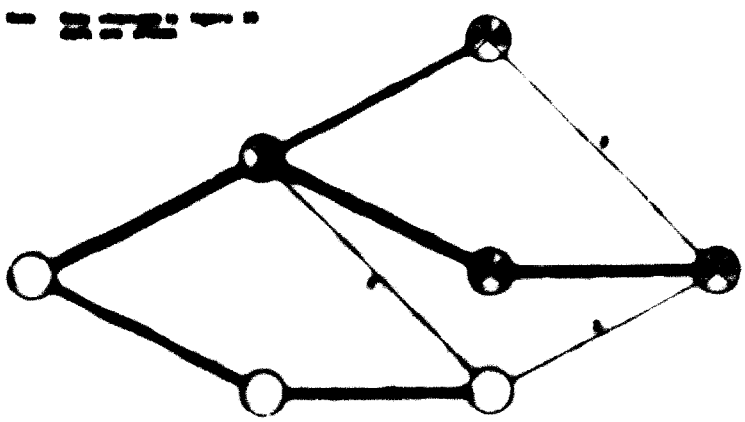


Figure 26 Updating for duration increase

connected to Activity *B* by zero-lag sequence lines. The earliest start and finish times of Activities *D*, *E*, and *G*, are all shifted to one day later. The scheduling of Activity *F* is unaffected, since the sequence line connecting it to Activity *B* has a positive-lag value. If the duration of Activity *B* were increased by one more day, the change would be transmitted in the same manner. If the duration of Activity *B* had been decreased, rather than increased, the same activities would have been affected, and each would have been shifted to an earlier date, as shown in figure 27.

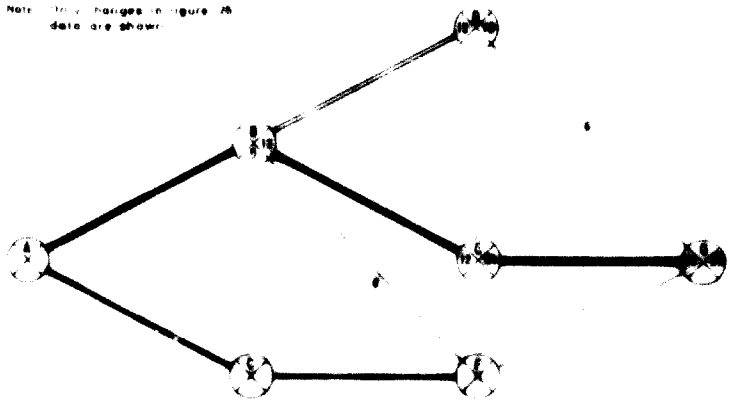


Figure 27. Updating for duration decrease

Changes in lag values

As scheduling changes take place in a network lag values may increase, decrease, or remain the same. If a following activity shifts to a later date while the scheduling of a preceding activity remains unchanged, the lag value of the sequence line that connects them will increase in value. The same result will occur when the preceding activity shifts to an earlier date while the following activity remains fixed. Conversely, lag values will decrease when a following activity shifts to an earlier date while the preceding one is fixed or the preceding one shifts to a later date while the following one is unaffected. Lag values will remain the same when both preceding and following activities are unaffected by the changes in scheduling or when both are affected and shift together by an equal amount. These points are illustrated by comparing figures 25 and 26. Because of the increased duration of Activity *B*, scheduling changes have been produced that cause the lag value of Sequence Line *F*–*G* to increase by one day and the lag value of Sequence Line *B*–*F* to decrease by one day. The lag values of Sequence *B*–*D*, *B*–*E*, *D*–*G* and *E*–*G* are unchanged, since both the finish time of the preceding activities and the start time of the following activities have shifted together to a later date. The lag values of Sequence Lines *A*–*B*, *A*–*C*, and *C*–*F* are unchanged, since neither the finish time of the preceding nor the start time of the following activities have been affected. Figure 27 indicates similar results accompanying a decrease in the duration of Activity *B*, except that the lag value of Sequence Line *B*–*F* increases rather than

decreases, while the lag value of Sequence Line *F* *G* decreases rather than increases. The former has increased to six days while the latter has decreased to one day.

NETWORK INTERACTION LIMIT

Since the lag values of sequence lines can change, they may be converted from positive to zero values or from zero to positive values. It has already been established that changes are transmitted within the network by zero-lag sequence lines and are not transmitted by positive-lag lines. Therefore, the introduction or deletion of a zero-lag relationship has a special significance, since it alters the behaviour of the network in transmitting changes.

Consider the case of a further increase in the duration of Activity *B* of figure 26. If the duration is extended to 15 days, as shown in figure 28, the lag value of Sequence Line *B* *F*, which has been decreasing, reaches zero. At this point the network will begin to behave in a different manner should the duration of Activity *B* be increased further. The scheduling of Activity *F* will begin to be affected. The lag values of Sequence Lines *B* *F* and *F* *G*, which have been changing, will remain constant. The lag value of Sequence Line *C* *F*, which has been constant and zero, will become positive and begin to increase.

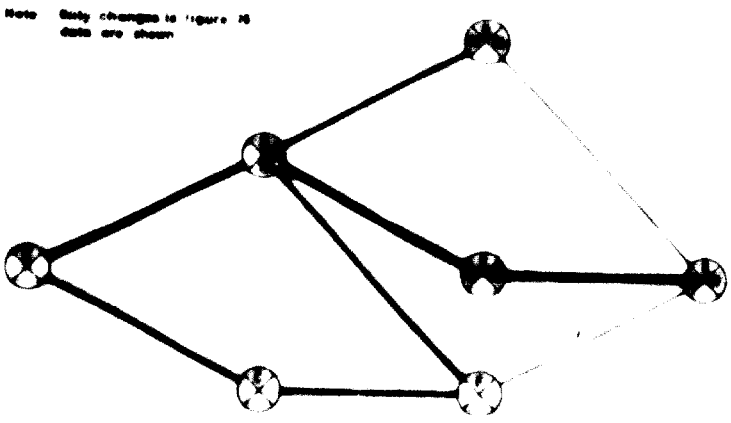


Figure 28. Updating for further duration increase

Another useful term that has been proposed (Fondahl, 1962) in connection with network mechanics is that of "network interaction limit", or time period associated with the change, in a specified direction, of the completion date of a specified activity. It is equal to the number of days of such change that will first cause a positive-lag sequence line to become a zero-lag line. For example, it could be stated for the conditions shown in figure 25 that the network interaction limit for an increase of the duration of Activity *B* is 5 days. Or, it could be stated that the network interaction limit for decreasing the duration of Activity *B* is 2 days.

The network interaction limit (NIL) indicates the number of time units that the completion time of a specified activity can be shifted in a specified direction before a basic change in the network functioning takes place. If a change is to be made that is greater than this limit, it is necessary to proceed in two or more steps, each step being limited by the NIL for the existing lag relationships at the conclusion of the previous step. At the conclusion of each step some positive-lag sequence line becomes a zero-lag line. This must be indicated on the diagram in order that subsequent changes will be transmitted through this line also. Often a new zero-lag sequence line will introduce a new critical path. This is the case when the duration of Activity *B* is reduced to 8 days and a second critical path is formed through Activities *A*, *C*, *F*, and *G*, as shown in figure 29.

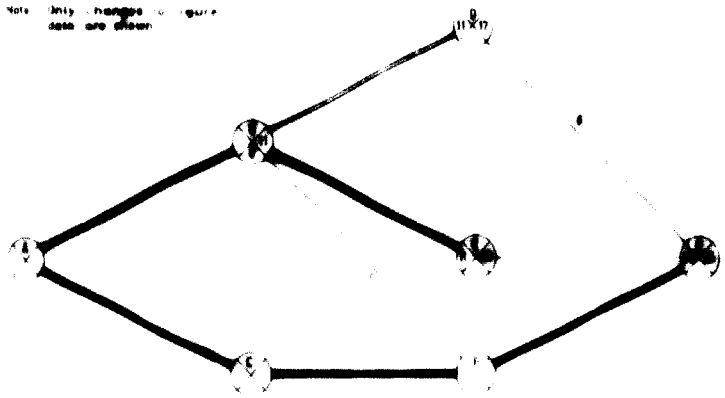


Figure 29 Addition of a critical path

A situation arises sometimes where a decreasing lag is indicated for a sequence line that already has a zero-lag value. Since negative-lag values are not permitted, this situation requires further investigation. For example, if the duration of Activity *B* of figure 29 were reduced further, the zero-lag sequence lines would transmit the change to Activity *G* but not to Activity *F*. This would indicate a shift of Activity *G* to an earlier date while Activity *F* remained fixed and, therefore, would indicate a decreasing lag value for Sequence Line *F* – *G*. However, this lag value has already reached zero. Analysis of the network in such cases will disclose that the following activity must stay fixed. Therefore, the zero-lag line that transmits the change to this following activity must be converted to a positive-lag line and cease to transmit the effects of the change. In the example it must be recognized that Sequence Line *E* – *G* of figure 29 will become positive when the duration of Activity *B* becomes less than 8 days. Figure 30 shows the updated network when the duration of Activity *B* is reduced to 5 days. Note that Activities *B* and *E* have become non-critical, leaving once again a single critical path but one that is different from the one in figure 25.

It will be noted that the definition of the network interaction limit assumes that a positive-lag value will become a zero-lag value, but it does not include

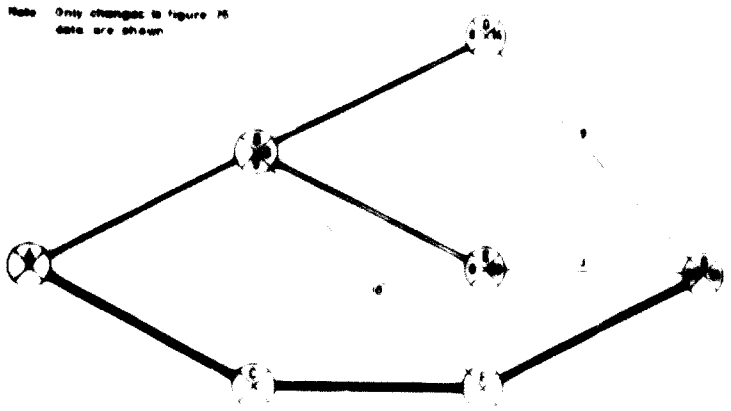


Figure 30 Elimination of a critical path

the case where a zero-lag value is converted to a positive lag. Both cases result in a modification of the network. However, the second case is automatically taken care of at the beginning or end of an updating computational cycle. The reason that this is true may be understood by analysing the two ways in which a zero-lag line can be converted to a positive-lag line. One way is for the following activity to remain fixed while the preceding activity shifts to an earlier date. This occurs only when there is at least one other zero-lag line merging into the following activity, and it results in the apparent negative-lag situation discussed in the previous paragraph. This situation is recognized at the very start of a computational cycle and is immediately remedied as already described. The second way for a zero-lag line to become positive is for the following activity to move to a later date while the preceding activity remains fixed. While the deletion of a zero-lag line should be taken into account for any subsequent updating cycles, it is not necessary to interrupt the cycle being commenced. The nature of the case indicates that affected activities are shifted to later dates and that the change involved does not affect the preceding activity of the sequence line under consideration. Therefore, whether the line is zero-lag or positive-lag has no effect on the transmission of changes in the updating cycle in process.

UPDATING OF ESSENTIAL DATA

Frequent updating of the schedules produced by network techniques is very important if the potential benefits from these procedures are to be obtained. With computer processing the simplest approach to updating is to change the items of data affected and then recompute the entire set of data for the network. It has generally been assumed that this is the simplest approach for manual processing also. This has placed practical limitations both on the frequency of

updating and on the size of network that can be handled by non-computer methods.

The network mechanics that have been described above can be applied to accomplish updating by a different procedure. Only the activities affected by a change need be updated. If in addition greater selectivity is exercised in determining the data for each activity that must be updated, manual procedures will have a greater range of application. In computer processing a requirement for twice as much data for each activity may involve a negligible increase in processing time. In manual processing, twice as much data will probably require twice as much time.

In the initial determination of the project schedule it is desirable to calculate a full set of data. This set would include earliest start and earliest finish times, latest start and latest finish times, and the total float and free float of each activity. At intervals during the progress of the work, it is usually desirable to repeat these calculations to produce a completely updated set of data. The most valuable updating, however, is not that which is performed periodically but rather that which is performed immediately whenever changes occur. This more frequent updating can be very effective without involving a complete updating of all data. The most important item of scheduling data for an activity is its earliest start time, or its scheduled start time if for some reason this is later. A record of the earliest or scheduled start times and of the current duration estimates for each activity should be continually maintained, and the critical activities should be correctly identified if possible, for this information is essential. It permits keeping those who are to perform in the future aware of changes in their scheduled start times, and it allows project management always to concentrate on the proper activities. For the application of the procedures to be described, it is also necessary to maintain a current record of lag values for each sequence line in the network. To make manual processing even more practical, updating may be limited to those activities in the immediate future, e.g., the next 30 days, rather than over the entire project duration. The use of subnetworks to be described later in this chapter will make this possible.

In summary, the essential data to be maintained on a continuous basis are activity duration estimates, earliest or scheduled start times, and the lag values of sequence lines.

If a subnetwork approach is adopted, this data need be maintained only for that portion of the project representing a limited time period ahead. All remaining data can be obtained if desired. Earliest of scheduled finish times are equal to earliest of scheduled start times plus durations. Free float can be determined from the lag values, since it is equal to the minimum lag value of those sequence lines leaving the activity. Of course, if only one sequence line bursts from the activity, activity free float will be equal to the lag value of this sequence line. Total float can be determined by a procedure, described later in this chapter, that does not necessitate the computation of other data. Latest start and finish times can be determined from the corresponding earliest start and finish times plus total float. Critical activities can be identified by noting the activities connected to the final project activity by chains of zero-lag sequence lines.

METHODS FOR UPDATING

Methods for manual updating of scheduling data may be developed from the principles of network mechanics that have been described. A change in one activity is conveyed to other activities by zero-lag sequence lines. By identifying these zero-lag lines on the diagram, the activities affected by a given change can be quickly determined. It is essential in applying this procedure that the lag values be kept updated at all times. When the affected activities have been identified, the sequence lines connecting these activities with unaffected activities can also be identified. These lines are the ones with changing lag values. By noting whether scheduling data is being shifted to earlier or later dates and whether the preceding or following activity of the sequence line is the one affected, it is possible to determine whether these lag values are decreasing or increasing. This permits updating of the lag values, addition or deletion of zero-lag sequence lines and a determination of network interaction limits.

Changes requiring updating involve changes in activity duration, changes in activity scheduling, deletion of existing activities, addition of new activities, or some combination of these. If the procedure for updating for duration change is understood, the procedures for the other changes will follow easily, since they involve only slight modifications. Therefore, this basic case will be described in some detail. It is assumed that a network diagram exists and that the zero-lag sequence lines are distinguished from the positive-lag lines, as shown in figure 25. It is also assumed that certain data records have been maintained. One of these contains the current duration estimate and earliest (or scheduled) start date for each activity. The second contains the current lag value for each sequence line. A third record that is recommended consists of a documentation of each change that is made including the nature of the change, the reason for the change, the magnitude and direction of the change, and a listing of the affected activities and sequence lines.

The activity whose duration is changed is marked on the diagram. A convenient method is to mount the diagram on sheet metal and to use a magnet of a certain colour (e.g., red) for marking activities. Then, all the activities following the changed activity and connected to it by zero-lag sequence lines are carefully and systematically marked in a similar way. The markers indicate the activities whose start dates need to be updated (except for the activity charged in duration whose start time remains fixed). Next, all the sequence lines having decreasing lag values are marked using a magnet of a different colour (e.g., green). If, for example, the change involves an increase in duration of the activity, sequence lines extending from marked activities to later unmarked ones will have decreasing lags. The next step is to mark all the sequence lines with increasing lags using magnets of still a different colour (e.g., blue). In the case of increased duration all sequence lines coming into marked activities from earlier unmarked activities (except for those coming into the activity whose duration is changed) will have increasing lags. Network interaction limit is determined next. This is accomplished by checking the current lag values of all sequence lines having decreasing lags and selecting the least of these values. Finally, the actual updating is performed.

If the change in activity duration is less than the NIL, the start times of all affected activities and the lag values of all affected sequence lines are changed by an amount equal to the duration change. As changes are entered on the records corresponding magnets may be removed from the diagram. When all have been removed the updating is completed. When the change in duration is greater than the NIL, the affected activities and sequence lines are updated by an amount equal to the NIL. The new zero-lag sequence line relationship is entered on the diagram, then additional cycles of updating are performed until the increment to complete the change is less than the NIL for the final cycle. More than two cycles are seldom required.

As an example involving most of the complications that will be encountered, assume that the duration of Activity *B* of figure 25 is decreased to 6 days. Markers would be placed on the diagram as shown in figure 31. A check of decreasing

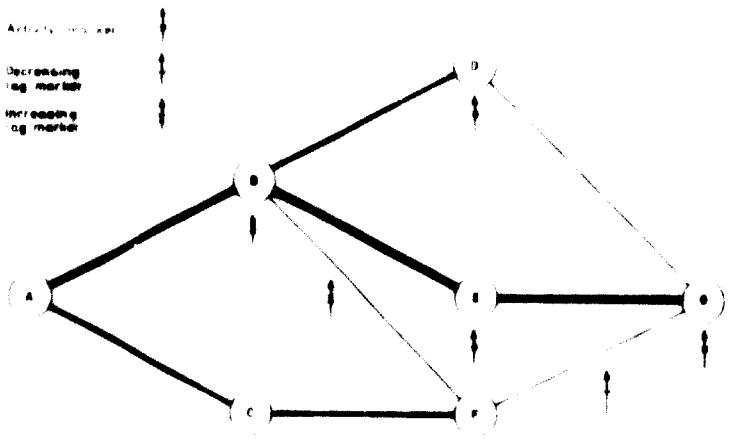


Figure 31. Initial cycle for duration change

lag lines (only $F-G$ here) would indicate a NIL of 2 days. The duration of Activity *B* and the earliest start times of Activities *D*, *E*, and *G* would be reduced by two days in the records. The lag values of Sequence Lines $B-F$ and $F-G$ would be changed to 7 and 0 respectively and the diagram altered to show the new zero-lag relationship. An attempt to perform a second cycle of updating would indicate a decreasing lag for Sequence Line $F-G$, which has already become zero. This requires recognizing that Sequence Line $E-G$ must be converted to a positive-lag line instead. Having made this alteration and once more marked the activities and sequence lines with changing values, the diagram would appear as shown in figure 32. Since there are no decreasing lags, the NIL is infinite and the remaining two days of updating can be completed. The duration of Activity *B* and earliest start times of Activities *D* and *E* are reduced by the final two days. The lag values of Sequence Lines $D-G$, $E-G$, and $B-F$ are increased by two days. Markers are removed as changes are recorded, and the updating is completed.

Updating involving changes in more than one activity can be accomplished in a single operation as long as the activities requiring updating are not con-

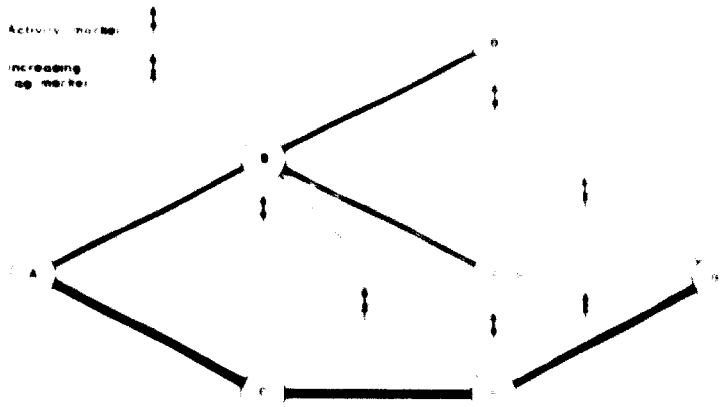


Figure 12 Final cycle for duration change

nected to one another by chains of zero-lag sequence lines and as long as the changes are in the same direction with respect to time. If the changes are different in magnitude, the change of least magnitude is accomplished concurrently for all the activities. Additional cycles of similar concurrent updating are performed for the remaining activities until the network has been updated for the activity having the change of greatest magnitude. However, if the activities whose durations are changed affect one another or if changes of both increased and decreased durations are involved, the updating for such changes must be performed in separate operations.

A change involving rescheduling an activity to a different date is handled in a very similar fashion to a change involving variations in duration. The only difference is that the start time of the activity being rescheduled is shifted to a different date as well as its finish time and, therefore, must also be changed. If sequence lines come into the rescheduled activity from earlier unmarked activities, they will have changing lag values in this instance.

A change involving the deletion of an activity from the network can be accomplished by reducing the duration of that activity either until it reaches zero or until the activity commences to develop free float, whichever occurs first. The activity will begin to have free float when all sequence lines departing from it to later activities have been converted to positive-lag lines. When either of these conditions has occurred, the activity can be deleted from the network. It is necessary to examine the sequential relationships between the preceding and following activities of the deleted activity to determine whether new sequence lines between them are required.

A change involving the addition of an activity can be accomplished by inserting the new activity and its necessary sequence lines into the diagram, the duration of the new activity having been temporarily set at zero. The duration of the new activity is then increased from zero to its estimated value using the same procedures as for changes in activity duration.

Other changes in project implementation programming and scheduling can be treated as combinations of those already discussed. For example, a change

in sequencing of two activities can be accomplished by the deletion of one of the activities and then by its addition in its new position.

TOTAL FLOAT COMPUTATION

The initial scheduling calculations for the entire network provide a complete set of data including latest start and finish dates and total floats. The latter provide a knowledge of the relative degree of criticality of all non-critical activities, and this is helpful in intelligently controlling the work. The updating procedures just described do not update this data for day-to-day control purposes this is not a serious disadvantage. As long as all critical activities are identified at all times and as long as the complete set of data is updated at periodic intervals to re-establish the awareness of relative criticality of non-critical activities, it is not essential to continually maintain current data on latest start and finish times and total float. But sometimes it is desirable to know the current value of total float of an activity, for example, when a decision on a rescheduling proposal is to be made. This information can be obtained for any given activity by the updating procedure already described for increasing the duration of an activity. The activity whose total float is to be determined is temporarily increased in duration until a zero-lag path is formed between it and the final project activity. In other words the number of days of delay to cause it to become critical is determined. This is its total float. This procedure has the advantage of also clearly showing the other activities that will be affected by the use of this float time. This information is generally required in reaching an intelligent decision regarding the proposed use of total float time, but it is not provided by the customary tabulations of updated values of total floats of all activities.

With the capability of determining total floats comes the ability to obtain updated latest start and finish dates by simple addition of data already available. Therefore, the procedures discussed in the foregoing paragraphs allow updating of any data that are needed without a complete recomputation of data for the entire network.

SUBNETWORKS

Another important procedure that might be classified as an application of network mechanics is the development and use of subnetworks. The simplest approach is to isolate a portion of the larger network where that portion is tied to the remainder at only two points. The subnetwork can be removed for separate analysis and can be replaced in the main network by a single equivalent activity of a duration determined by the critical path of the subnetwork. Unfortunately, this does not occur frequently. It is more usual to expand a single activity in the master network into a network of its own for detailed analysis purposes.

There is another approach called the "dateline cut-off" method (Fondahl, 1962). Here the master network is divided into subnetwork by time periods.

Ordinarily only the subnetwork for the current time period (e.g., one or two months) would be developed. As work progresses the subnetwork for the next period is developed. When work moves into this next period the preceding subnetwork is dropped. This process continues until the final subnetwork for the last time period has been completed.

This method has two important advantages. First, it allows the use of a much smaller network; thus makes manual methods feasible where otherwise they would not be. For example, consider a project network of 600 activities that has a duration of two years. If subnetworks for one-month periods were developed, the average network might be expected to contain only about 25 activities. Assuming the largest network might contain twice as many activities as the average, it still would involve only 50 activities. Manual procedures can easily be applied to a network of this size. The second advantage is that attention can be concentrated on that portion of the work that is to occur in the immediate future. While over-all implementation programming and scheduling for the entire project is very important, it is not practical to carry it out in detail. It is a waste of time and effort to perform detailed updating or detailed resource levelling for activities that are to be performed, for example, a year hence since many more changes will inevitably occur before those activities can start. It is generally quite desirable, however, to give very detailed attention to the implementation plan and schedule for the next 30 or 60 days, for example.

It is important that updating be performed on a day-by-day basis so that participants scheduled to perform in the immediate future have the latest information concerning their expected start times. Greater efforts are justified in attempting to level resource requirements, and it becomes practical to consider resource schedules for resources other than the few crucial ones that were studied for the entire project.

For this subnetwork method to be useful, it is important that the effects of changes within the subnetwork be properly reflected by the project completion date. Also, it must be possible to determine and update all the required data, including knowledge of criticality, for the activities in the subnetwork.

A simple example will illustrate the principles of the date-time cut-off method. Consider the master network shown in figure 33. Assume that subnetworks for 30-day periods will be used. Scheduling calculations for the entire network are made initially. Results are shown in the tabulation of figure 34. The first

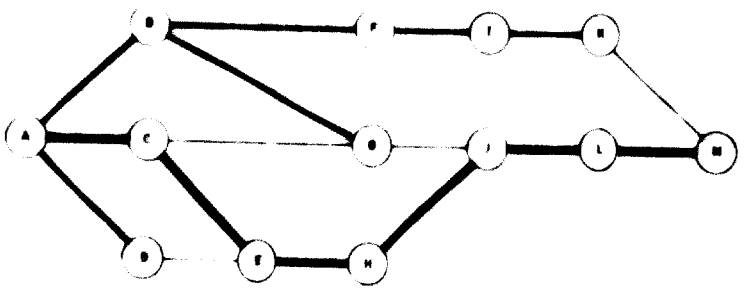


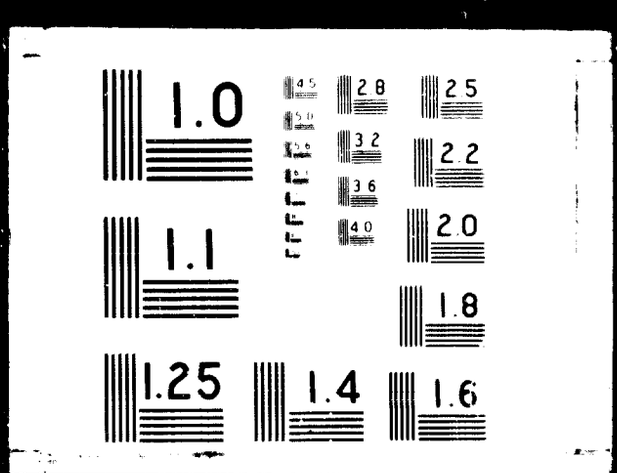
Figure 33. Master network

2. 8. 73

2 OF 2

DO

3001



Activity	Activity duration (days)	Critical activity	ES	EF	LS	LF	TF	FF
A	5	*	1	6	1	6	0	0
D	30		6	36	16	46	10	0
C	20	*	6	26	6	26	0	0
D	10		6	16	16	26	10	10
E	25	*	25	51	26	51	0	0
F	20		33	56	46	66	10	0
G	5		36	41	51	56	15	15
H	5	*	51	56	51	56	0	0
I	10		56	66	66	76	10	0
J	20	*	56	76	56	76	0	0
K	10		66	76	76	86	10	10
L	10	*	76	86	76	86	0	0
M	5	*	86	91	86	91	0	0

Figure 34. Updated scheduling data

subnetwork is constructed to include all activities commencing before or on the thirtieth day. An event representing project completion is added at the right end of the diagram, and artificial "tie" activities are added between this completion event and the interface activities. An interface activity is an activity in the subnetwork that has one or more sequence lines extending from it to activities in the portion of the master network that has been removed. The duration of each tie activity is equal to the number of days on the longest path between its preceding activity and the project completion event. This is determined most easily by the use of the late start and finish data that have been computed and shown in figure 34. The tie duration is equal to the "effective" latest finish time of its preceding activity subtracted from the project completion date. If the preceding activity has no following activities within the subnetwork, its effective latest finish time is the same as its ordinary latest finish time; this is true for Activities *B* and *E*. If the preceding activity does have following activities within the subnetwork, its effective latest finish time is equal to the earliest of the latest start times of its following activities that are outside the subnetwork; this is true for Activity *C*. The resulting subnetwork is shown in figure 35. An alternative method employing dual entries for the interface activity data eliminates the necessity for the tie activities (Fondahl, 1962).

The subnetwork of figure 35 does not show an impressive reduction in activities, but it would in the case of an actual project with a network of considerable size. The subnetwork would probably be expanded in detail after it had been developed initially. Note that the critical path through the subnetwork is always known. If a new critical path is formed, the resulting critical activities in the subnetwork will be identified, but those in the remainder of the project will not be known until later subnetworks are developed or the entire project

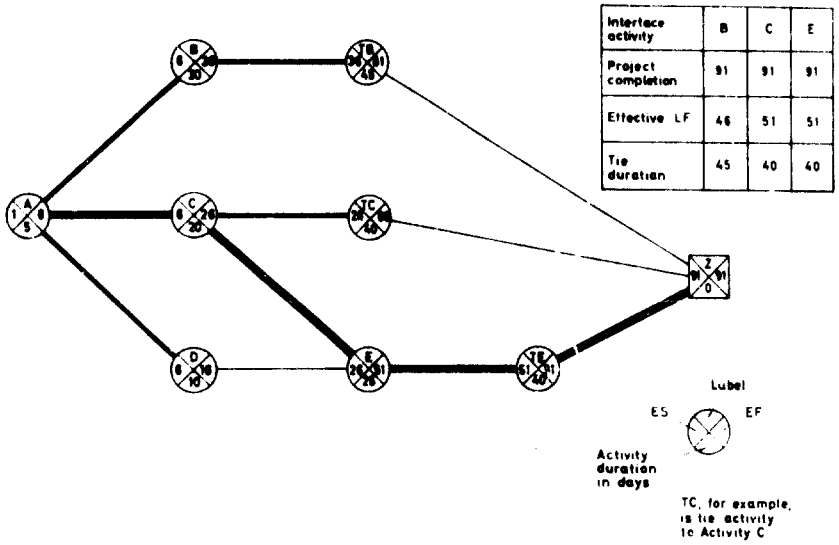


Figure 35. Subnetwork for first 30-day period

is updated. Note also that since the event representing final project completion is included in the subnetwork, the effect of changes occurring within the subnetwork will be reflected in it.

Figure 36 shows the subnetwork for the second 30-day period, assuming no changes during the use of the first subnetwork. This subnetwork is developed while work within the first subnetwork is under way. After work has moved into the period of the second subnetwork, the first is discarded and work on develcpment of the third-period subnetwork soon begins. This process continues until the final subnetwork covering the last 30-day period has been developed.

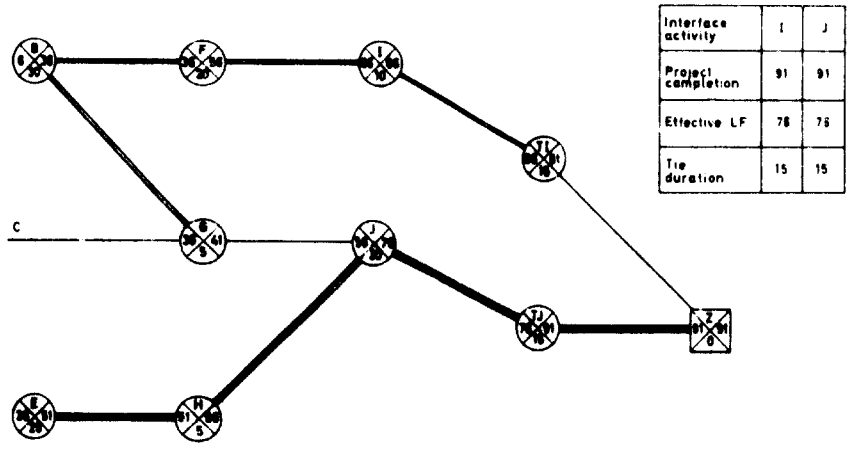


Figure 36. Subnetwork for second 30-day period

The application of subnetworks can be useful, especially for schedule updating and detailed resource levelling. There are certain shortcomings that

result from a lack of knowledge of the effects on activities and resource requirements in the project period following the subnetwork time span. There are also certain shortcomings in the time-cost trade-off analysis that result from inability to identify the critical activities that follow the subnetwork portion of the over-all diagram. But the advantages of the application of subnetworks outweigh these shortcomings.

Chapter 4

TIME-COST TRADE-OFFS

INTRODUCTION

For implementation programming purposes, network techniques require subdivision of a project into many separate activities. For implementation scheduling purposes, time estimates for the performance of each of these activities are required. Actually, each activity can generally be performed in a number of ways, and these ways have different time requirements. Since there are a number of activities and a number of variations for the performance of each, there exists an almost infinite number of possible schedules for the project. The programmer's objective is to develop the particular schedule that will provide the most favourable solution. Total cost is a major factor in judging the effectiveness of a solution. Therefore, the introduction of cost data provides a basis for choosing among the many scheduling possibilities. The time-cost trade-off technique discussed in this chapter is a means for applying these cost data in a systematic and logical manner. When a project completion date is already specified, time-cost trade-offs are applied to produce the most economical schedule that will meet the completion deadline. When the objective is to determine the most economical schedule, time-cost trade-offs are applied to develop such a schedule and to establish the corresponding completion date as well.

PROJECT TIME-COST RELATIONSHIPS

In discussing project time-cost relationships it is useful to distinguish between direct and indirect costs, since their patterns of variation are quite different. Direct costs are those associated with the activities into which the project is subdivided. These costs vary according to the method and manner of performance of the activities. For example, for physical construction they include costs such as those for labour, materials, and rent of equipment. Indirect costs are those associated with the project as a whole and vary mainly with the passage of time. They may include salaries of management and office personnel, interest or the cumulative project investment, insurance, and maintenance of utilities and services during construction. They may also include the loss of benefits for each time period during which the project has not been completed.

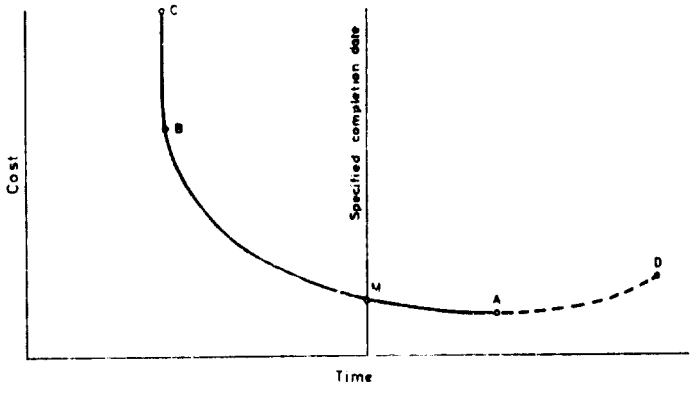


Figure 37. Project time-direct cost curve

The curve of figure 37 provides a basis for initial discussion of direct cost and time relationships. Point *A* represents a schedule that can be developed quite simply. It assumes that every activity in the project is being performed in the manner that results in the lowest direct costs. These are the costs that one usually supplies if the activity is being contracted. An estimator should be able to furnish these costs and the corresponding activity durations. The direct co-ordinate for Point *A* is simply the sum of the activity costs for all the project activities. The time co-ordinate for Point *A* is the project durations resulting from the conventional forward-pass calculations using the activity times furnished by the estimator. In general, there is no reason for developing the curve *AD* to the right of Point *A*. It would merely represent schedules that result in post-ponements of project completion without lowering project costs. In fact, cost increases usually result from inefficiencies that are generally associated with "dragging out" activity performance beyond the time that results in the achievement of lowest direct cost.

Point *C* represents another schedule that can also be developed quite simply. It assumes that every activity in the project is being performed by the fastest possible means, and therefore represents the "all-crash" schedule. An estimator should be able to furnish the activity costs and corresponding durations for such a schedule. The direct cost and the time co-ordinates for Point *C*, then, can be obtained in a similar manner to those for Point *A*. Points *A* and *C* establish the limits of the time range for all acceptable scheduling solutions. Another point, Point *B*, represents a schedule for the same minimum time limit as Point *C* but has a considerably lower cost in almost every case. Even with the fastest schedule possible for project performance, there are some activities that never become critical and, hence, do not affect project duration. The Point *C* schedule includes the cost of expediting, or "crashing", every activity whether it is critical or non-critical. The Point *B* schedule includes the costs of expediting only those activities that are effective in reducing project duration. Point *C* represents the customary approach of the manager faced with the necessity of a crash effort but lacking the information provided by network methods. It is the "all-crash" approach. Point *B* represents a much more economical solution for a crash effort and is based on the intelligent and selective expediting that

a knowledge of updated network data permits. The curve *AB* represents the most favourable scheduling solutions for project durations of intermediate durations. As in the case of Point *B*, these schedules are developed by selective expediting based on a knowledge of network data and associated cost estimates. The purpose of the time-cost trade-off technique is to offer a procedure for developing the scheduling solutions along the curve *AB*.

Common terminology for Point *A* describes it as the "normal" project-scheduling solution. Point *C* will be referred to as the "all-crash" solution and Point *B* as the "minimum-cost crash" solution. The curve *AB* shown in figure 37 is an idealized one in that it is smooth and continuous. Actually, owing to the tremendous number of possible scheduling variations available to the imaginative estimator, this becomes a reasonable representation. The shape of the curve has not been established yet, but it should appear intuitively correct. Starting at Point *A*, if there were reason for shortening project duration, those measures causing the least increase in cost would be taken first. As the most favourable opportunities for expediting were exhausted, more expensive measures would have to be adopted. As one progresses from right to left along the curve *AB* it seems logical that it will become more and more expensive to buy time until finally at Point *B* the only expediting opportunities serve merely to increase cost without reducing time.

If a project completion date were specified, a schedule represented by Point *M* would provide the most economical solution for completion on that date. It is possible, however, that a consideration of indirect costs might lead to a schedule for even earlier completion in order to reduce total costs. Therefore it is desirable to consider the indirect cost-time relationships as well as the direct cost curve.

The indirect cost curve would be one sloping upward to the right, since indirect costs tend to rise with the passage of time. Although it is frequently portrayed as a straight line, the curve is not necessarily linear but probably has sudden jumps and breaks in slope. It is not really essential to establish the entire curve or even to know the true value of its ordinates. Scheduling decisions require only a knowledge of the changes in indirect costs between different dates, and these dates would be in the time range between normal and crash performance. In fact, it is usually sufficient if the rate of change of indirect costs is known only at specified project durations. Then a decision can be based on whether the cost of expediting the schedule for that duration will cost more or less than the saving in indirect costs. Figure 38 shows the direct cost curve and the portion of the indirect cost curve for the range of schedules of interest, both plotted on the same time scale. It also shows the resulting total cost curve obtained by the addition of the above two curves. Since direct and indirect costs vary in opposite directions with respect to time, the total cost curve will have a minimum point. The schedule corresponding to the minimum point would be the most advantageous one unless other factors dictate a different completion date. Since the indirect cost curve, or at least the incremental indirect costs over a limited range of time, is not difficult to develop, the ability to make use of these time-cost relationships depends primarily on being able to develop

the direct cost-time curve. An understanding of the activity time-cost relationships is necessary first.

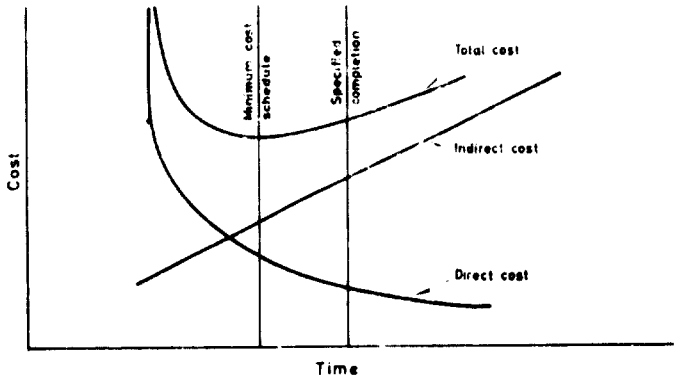


Figure 38. Project time-total cost curve

ACTIVITY TIME-COST RELATIONSHIPS

Each activity can usually be performed in a number of ways, as stated at the beginning of this chapter. There is one method of performance that will result in the lowest cost, and this is referred to as the "normal" solution. To expedite the activity from its normal performance will usually require measures that will increase costs. These might include overtime or shift work, inefficient use of larger crews, or a more expensive method. The limit that marks the shortest time for activity performance is called its "crash" performance.

The simplest time-cost curve for an activity is a single, continuous, straight line connecting normal and crash points. Accomplishment of an activity at any intermediate time between the normal and crash times is carried out at a corresponding cost. Since there are many activities and each one is a relatively minor element of the over-all project, such an approximation has been considered acceptable. Most time-cost procedures are performed using this assumption. It involves the least degree of complexity and requires the least amount of data per activity from the estimator. The data would be limited to normal and crash times and costs for each activity. Figure 39 (a) indicates this activity time-cost curve.

A more detailed time-cost curve might be similar to that shown in figure 39 (b). It has the intuitively correct shape discussed for the project time-cost curve, i.e., that each increment of expediting effort requires a higher expenditure per time unit because the most economical measures are taken first. Such a curve may be used if the straight line approximation, figure 39 (a), is not suitable. Such a curve is not difficult to apply in isolated cases, but its general application has usually been considered unjustified because so much additional data would be required. Its application might be justified in the case of an activity that is expanded into a subnetwork for analysis purposes. If a time-cost curve were developed for the resulting "subproject", this curve would become

an activity time-cost curve of the general shape of figure 39 (b) for the original activity.

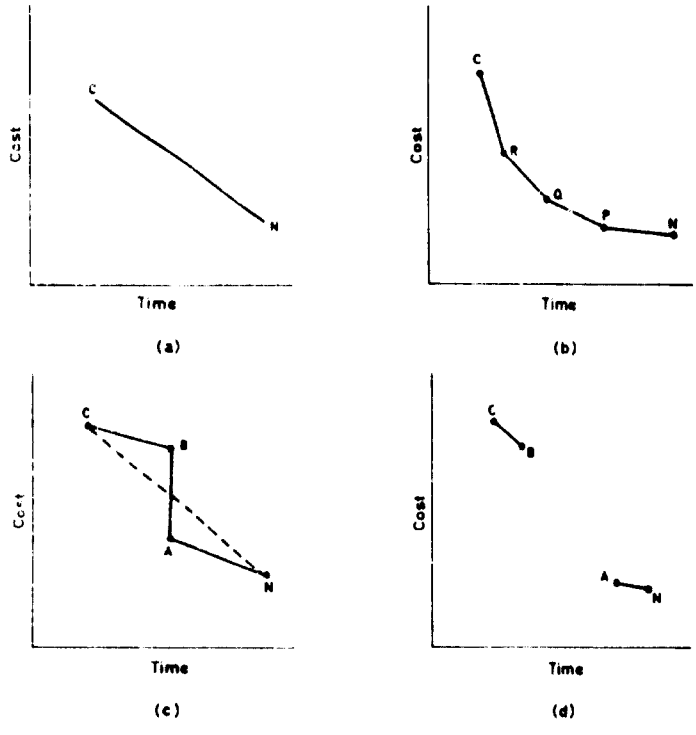


Figure 39. Activity time-cost curves

Another activity time-cost relationship that may often exist is shown in figure 39 (c). Segment *NA* represents one method of performance and accompanying expediting by such measures as assigning more manpower or overtime work. Segment *AB* represents switching to an alternative method of performance. In other words, segment *AB* represents the increase in cost effected from using the new method. Segment *BC* represents expediting of this new method by such measures as assigning more manpower or overtime work. This general type of curve, i.e., one that is not convex upward over its entire range, is very difficult to apply in trade-off procedures. Theoretically, it can be used, but even large capacity computers using the mathematical methods involved can only solve very trivial-size networks in a practical length of time. Fortunately, a straight line approximation between normal and crash points, as shown by the broken line in the figure, is usually an acceptable representation of such a curve.

Finally, curves of the type of figure 39 (d) can also occur. All of the activity time-cost relationships are probably more accurately represented by discrete points rather than continuous curves. There are far fewer scheduling combinations than is the case with project data that produce a nearly continuous curve. Sometimes the activity time-cost schedules have very definite discontinuities with wide gaps between successive solutions, as in figure 39 (d). This

is often the case with activities associated with supply where there are limited alternatives and perhaps each is subject to expediting over only a very short range of time. When such activities are critical and offer definite possibilities for economical project-shortening, it may become necessary to make a separate analysis of the project time-cost curve based on using first one discontinuous portion of the curve and then another. The most favourable portions of the two resulting project time-cost curves can be used for scheduling.

DEVELOPMENT OF THE PROJECT TIME-COST CURVE

A procedure based on the principles of network mechanics already discussed and practical for manual application is offered for developing the project time-direct cost curve. It is assumed initially that the estimator furnishes time and cost data for every activity and that straight line activity time-cost curves, similar to that of figure 39 (a), are acceptable. These are the assumptions of most computer procedures using much more sophisticated linear programming methods. Later in this chapter definite improvements over these initial assumptions that simplify manual procedures, make them more effective, and allow them to use more realistic data than permitted by computer procedures will be indicated.

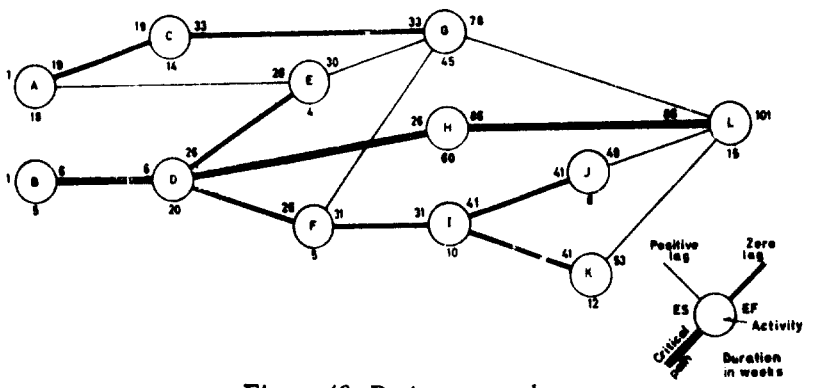


Figure 40. Project network

The project network of figure 40 and the data of figure 41 illustrate the proposed method. The estimator determines first the most economical manner of performing each activity and the corresponding activity durations. These are shown as Normal times and Normal costs in figure 41. The estimator estimates the fastest possible performance time and the corresponding cost for each activity. These result in the Crash time and Crash cost data of figure 41. The difference between crash cost and normal cost is divided by the difference between normal time and crash time to obtain the slope in cost per time unit of the activity time-cost curve (cost per month, per week, per day etc.). Thus the assumption of a linear, continuous curve between two limiting points is made a basis for the procedures that follow. The cost slopes for each activity have been computed in figure 41.

Activity	Normal time (weeks)	Normal cost (dollars)	Crash time (weeks)	Crash cost (dollars)	Time difference (weeks)	Cost difference (dollars)	Cost slope (dollar/week)
A	18	200,000	10	240,000	8	40,000	5,000
B	5	80,000	5	80,000	0	0	NONE
C	14	15,000	10	16,000	4	1,000	250
D	20	20,000	15	21,500	5	1,500	300
E	4	42,000	2	54,000	2	12,000	6,000
F	5	60,000	4	64,000	1	4,000	4,000
G	45	80,000	30	125,000	15	45,000	3,000
H	60	42,000	45	64,500	15	22,500	1,500
I	10	72,000	5	78,000	5	6,000	1,200
J	8	18,000	5	21,000	3	3,000	1,000
K	12	45,000	8	53,000	4	8,000	2,000
L	15	81,000	15	81,000	0	0	NONE
Totals		755,000		898,000			

Figure 41. Project time-cost data

Figure 42 shows a summary sheet for entering data that will produce the project time-direct cost curve. The curve development must start at a point that can be determined. Methods for obtaining the co-ordinates for Points *A* and *C* of the project curve of figure 37 have been discussed earlier. Point *A*, the "normal" schedule, will serve as a logical starting point. The curve will be developed from this point by cycles of computation, each cycle producing an additional segment as the project is expedited from its normal schedule. The normal project duration is determined by forward-pass calculations using the activity normal times given in figure 41. A period of 100 weeks (earliest finish of final Activity *L* is the beginning of the 101st week) has been chosen for the project. The normal project direct cost is determined by totalling the activity normal costs to obtain $755,000, as shown in figure 41.

In each cycle of computations two questions must be answered: "Which activities should be expedited to achieve project-shortening with the least in-

Cycle	Activities shortened	Possible shortening (weeks)	NIL (weeks)	Shortening (weeks)	Cost/week (dollars)	Cost increase (dollars)	Total direct cost (dollars)	Project duration (weeks)
0							755,000	100
1								
2								
3								
4								
5								

Figure 42. Summary sheet

crease in cost?" and "By how many time units may the project duration be expedited in this cycle?" The answer to the first question must be based on three criteria: the activity (or activities in the case of multiple critical paths) chosen must be critical if it is to affect project duration; it must be one that the estimator has indicated can be shortened and that has not already been expedited to its crash limit; and it must have the lowest possible cost slope in order to produce the most economical expediting possible for the existing schedule. To facilitate the selection of the activities for expediting, the tabulation of figure 43 can be used. This is not absolutely essential but will aid in illustrating

Activity	Cost slope (dollars/week)	Critical activities	Finish	Possible shortening (weeks)						
				Cycle 0	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6
C	350			4						
D	300	0		5						
J	1,000			3						
I	1,200			5						
H	1,500	0		15						
K	2,000			4						
O	3,000			15						
F	4,000			1						
A	5,000			8						
E	6,000			3						
B	NONE	0	0	0						
L	NONE	0	0	0						

Figure 43. Selection sheet

the thought process involved. The activities are arranged in ascending order of cost slopes. If cost slopes are identical, the earlier activity is listed first. The third column indicates whether the activity is critical. Since additional activities will probably become critical as expediting continues, the entry in the third column indicates the cycle in which the activity became critical. A blank in the fourth column indicates that the activity can be shortened; an entry in this column shows the cycle in which the activity has been expedited to its crash duration and indicates that no further expediting of this activity is possible. If the estimator determines that an activity cannot be expedited, an entry is made in column IV at the beginning of the procedure; this is the case with Activities *B* and *L*. The remaining portion of the selection sheet provides space to record the amount of time by which an activity can be expedited and space to update this data. To answer the first of the two principal questions, "Which activity should be shortened?", the highest activity on the sheet (least expensive to expedite) that has an entry in column III (critical) and does not have an entry in column IV (can be shortened) is chosen. The activity is entered in column II of the summary sheet, figure 42. For multiple critical paths the selection procedure is somewhat more complicated and will be explained later.

The answer to the second question, "By how much time?", can be established by two criteria. The first is the number of time units by which the selected activity can be shortened. For multiple activities, the activity with the least

amount of possible shortening governs. The second criteria is the network interaction limit for the proposed duration change. When an activity is shortened by an amount equal to its NIL, the cycle must be terminated in order to update the lag relationships. Possibly a new critical path will be formed and will require the selection of different activities for further project-expediting. The use of the NIL is important because it eliminates trial and error methods for determining the effective project-shortening of each proposed change. The NIL can be determined by the marking procedure discussed in Chapter 3 for updating when activity durations are changed; lag values must be computed and updated with each change. Figure 44 shows a tabulation of lag values and space for updating them.

Preceding activity	Following activity	EF-preceding	ES-following	Lag	Revised lags									
					Cycle 1		Cycle 2		Cycle 3		Cycle 4		Cycle 5	
A	C	19	19	0										
	E	19	26	7										
B	D	9	9	0										
C	G	33	33	0										
D	E	20	20	0										
	F	30	30	0										
	H	30	30	0										
E	G	30	33	3										
F	G	31	33	2										
	I	31	31	0										
G	L	70	66	0										
H	L	99	96	0										
I	J	41	41	0										
	K	41	41	0										
J	L	49	66	37										
K	L	53	86	33										

Figure 44. Lag sheet

The procedure for each cycle of computation necessary for developing the project time-direct cost curve may be outlined as follows:

(a) Using the selection sheet (figure 43), choose the activity to be expedited; it will be the first activity that has an entry in column III and none in column IV.

(b) Enter this activity in column II of the summary sheet (figure 42). Determine its possible shortening from the selection sheet and enter this figure in column III of the summary sheet. Determine its cost slope from column II of the selection sheet and enter this figure in column VI of the summary sheet.

(c) Using the network diagram (figure 40), place the appropriate markers to indicate the affected activities and the sequence lines with decreasing and increasing lags. Using plus or minus signs, mark on the lag sheet (figure 44) the sequence lines that have changing lag values. Check the

lag values for those lines that have been marked with a minus sign to determine the least value that gives the NIL. Enter the NIL in column IV of the summary sheet.

(d) Complete the entries for the cycle on the summary sheet. The amount of shortening for the cycle is entered in column V and is equal to the minimum of the figures entered in columns III and IV. The Cost increase in column VII is the product of the figures in columns V and VI. The new value for the Total direct cost is the sum of the Cost increase and the Total direct cost of the preceding cycle. The new value for the Project duration is the amount of shortening from column V subtracted from the Project duration for the preceding cycle.

(e) Update the lag sheet (figure 44) using the amount of shortening for that cycle as determined from column V of the summary sheet. Add this amount to the previous lag values of sequence lines marked with a plus sign and subtract this amount from the existing lag values of sequence lines marked with a minus sign.

(f) If any lag values are changed from positive to zero (as will be the case when the NIL governs the amount of cycle-shortening) or from zero to positive, modify the project network (figure 40) accordingly by showing the new relationships for the lines involved.

(g) Update the selection sheet (figure 43) by changing the Possible shortening figures for the activities expedited. If an activity has reached its crash limit, make an entry in column IV to indicate the cycle in which this occurred. If a new zero-lag relationship has produced a new critical path on the project network, make entries in column III of the selection sheet indicating these new critical activities and the cycle in which they became critical.

(h) Repeat this procedure starting again with step 1.

(i) Continue the procedure until:

(i) The entire curve to the project crash duration is developed. This will occur when all the activities on one critical path reach their crash limits. The corresponding project time can be determined in advance, as an independent check, by forward-pass calculations using activity crash durations from the data sheet (figure 41).

(ii) The direct cost per time unit for expediting project performance, as shown in column VI of the summary sheet, begins to exceed the estimated indirect cost per time unit that is saved by earlier completion.

(iii) The project duration has been reduced to correspond to a specified completion date even though the cost of expediting to that date is greater than the saving in indirect costs.

In general, criteria (ii) would appear to be the most logical basis for termination and would usually require only development of part of the total curve. Figures 45 to 56 illustrate this procedure step by step. Figures 54, 55 and 56, however, show also the summary, selection and lag sheets respectively at the conclusion of the development of the entire project time-direct cost curve for

Cycle	Activities shortened	Possible shortening (weeks)	NIL (weeks)	Shortening (weeks)	Cost/week (dollars)	Cost increase (dollars)	Total direct cost (dollars)	Project duration (weeks)
0							755,000	100
1	D	5	7	5	300	1,500	756,500	95
2								
3								
4								
5								

Figure 45. Summary sheet (Cycle 1)

Activity	Cost slope (dollars/week)	Critical activities	Finish	Possible shortening (weeks)						
				Cycle 0	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6
C	250			4						
D	300	D	1	5	0					
J	1,000			3						
I	1,200			5						
H	1,500	H		15						
K	2,000			4						
G	3,000			15						
F	4,000			1						
A	5,000			8						
E	6,000			2						
B	NONE	B		0						
L	NONE	L		0						

Figure 46. Selection sheet (Cycle 1)

the example. Had the indirect cost per week been estimated at $1,500, for example, the calculations could have been terminated following Cycle 2. Note that for project crash performance the total direct cost is $804,000 (figure 54). This sum corresponds to the cost for Point *B* of figure 37 and is substantially below the cost of $898,000 for the all-crash performance of figure 41. Note also that at the end of Cycle 2 a second critical path was produced. This necessitated a slightly more involved selection process for activities to be expedited in subsequent cycles. Either activities from both critical paths had to be expedited concurrently or an activity common to both paths had to be selected. In such cases it helps to list the two paths side by side, showing the portion of each path included between their common points. The most economical activity in each such sub-path can be circled. For the example at the beginning of Cycle 3, the listing would be as follows (Activity *L* is excluded from the list, since it is common to both paths and moreover cannot be shortened):

Preceding activity	Following activity	EF-preceding	ES-following	Lag	Revised lags									
					Cycle 1		Cycle 2		Cycle 3		Cycle 4		Cycle 5	
A	C	19	19	0										
	E	19	26	7	-	2								
B	0	6	6	0										
C	0	33	33	0										
0	E	26	26	0										
	F	26	26	0										
	H	26	26	0										
E	0	30	33	3	.	8								
F	0	31	33	2	.	7								
	I	31	31	0										
0	L	70	86	6	-	3								
H	L	96	96	0										
I	J	41	41	0										
	K	41	41	0										
J	L	49	86	37										
K	L	53	86	33										

Figure 47. Lag sheet (Cycle 1)

Cycle	Activities shortened	Possible shortening (weeks)	NTL (weeks)	Shortening (weeks)	Cost/week (dollars)	Cost increase (dollars)	Total direct cost (dollars)	Project duration (weeks)
0							795,000	100
1	D	5	7	5	300	1,500	756,500	95
2	H	15	3	3	1,500	4,500	761,000	92
3								
4								
5								

Figure 48. Summary sheet (Cycle 2)

Activities *B* and *D* from the original critical path have been struck out, since they have reached their crash limits. When Cycle 3 starts, Activity *C* would be the first candidate for shortening. The above tabulation would indicate that Activity *C* is in a critical path having a parallel critical path and that Activity *H* should be shortened concurrently. The combined cost is $1,750 per week. A further check down the selection sheet tabulation indicates that the next best selection (besides Activity *H*, which produces the same combination of Activities *C* and *H*) is Activity *G*, which costs $3,000 per week (and also requires concurrent shortening of Activity *H*). Investigation can be terminated at this point,

Activity	Cost slope (dollars/week)	Critical activities	Finish	Possible shortening (weeks)						
				Cycle 0	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6
C	250	2		4						
D	300	0	1	8	0					
J	1,000			3						
I	1,200			5						
H	1,500	0		16		12				
K	2,000			4						
G	3,000	2		18						
F	4,000			1						
A	5,000	2		5						
E	6,000			2						
B	NONE	0		0						
L	NONE	0		0						

Figure 49. Selection sheet (Cycle 2)

Preceding activity	Following activity	EF - preceding	ES - following	Lag	Revised lags									
					Cycle 1		Cycle 2		Cycle 3		Cycle 4		Cycle 5	
A	C	19	19	0										
	E	19	26	7	-	2								
B	D	8	8	0										
C	G	33	33	0										
D	E	26	26	0										
	F	36	26	0										
	H	26	26	0										
E	G	30	33	3	-	8								
F	G	31	33	2	-	7								
	I	31	31	0										
G	L	78	86	8		3	-	0						
H	L	86	86	0										
I	J	41	41	0										
	K	41	41	0										
J	L	49	86	37			-	34						
K	L	53	86	33			-	30						

Figure 50. Lag sheet (Cycle 2)

since even if there were a single activity common to both paths (such as Activity *L*) the cost would be greater than the $1,750 figure already achieved. If there are more than two critical paths, a similar analysis can be made as long as each path is listed with one of its parallel paths between their common points. Each pair of such listings must be investigated and satisfied when an activity is considered for expediting. Investigations can be terminated when it becomes obvious, as above, that no combination can produce a better solution than the one already obtained.

Cycle	Activities shortened	Possible shortening (weeks)	Mil (weeks)	Shortening (weeks)	Cost/week (dollars)	Cost increase (dollars)	Total direct cost (dollars)	Project duration (weeks)
0							755,000	100
1	D	5	7	5	300	1,500	756,500	95
2	H	15	3	3	1,500	4,500	761,000	92
3	C and H	4 and 12	7	4	1,750	7,000	768,000	88
4								
5								

Figure 51. Summary sheet (Cycle 3)

Activity	Cost slope (dollars/week)	Critical activities	Finish	Possible shortening (weeks)						
				Cycle 0	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6
C	250	2	3	4			0			
D	300	0	1	5	0					
J	1,000			3						
I	1,200			5						
H	1,500	0		16		12	8			
K	2,000			4						
G	3,000	2		15						
F	4,000			1						
A	5,000	2		8						
E	6,000			2						
B	NONE	0	0	0						
L	NONE	0	0	0						

Figure 52. Selection sheet (Cycle 3)

The procedure outlined in this section is not mathematically perfect. Under circumstances that occur quite rarely, the optimum expediting combination of activities may not result from a procedure that is based exclusively on shortening these activities. The optimum solution can involve simultaneous shortening of some activities and lengthening of others. This situation can occur only when there are three or more critical paths and then under an unusual combination of conditions. In its simplest form, there would be three critical paths, *A*, *B* and *C*. A critical activity on Path *B* would have been previously expedited. Then there would exist a combination of one activity common to paths *A* and *B* and one activity common to paths *B* and *C* that could be shortened concurrently, producing two time units of shortening for Path *B* for each one for Paths *A* and *C*. This would permit extending the duration of the previously shortened activity on Path *B* by one time unit for each unit of shortening of the two selected activities. The combination of the costs of expediting the two selected activities minus the saving realized by "selling back" time for the activity

Preceding activity	Following activity	EF-preceding	ES-following	Lag	Revised lags				
					Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5
A	C	18	18	0					
	E	19	26	7	2				
B	D	6	8	0					
C	G	33	33	0					
D	E	28	28	0					
	F	26	26	0					
	H	26	26	0					
E	G	30	33	3	8		4		
F	G	31	33	2	7		3		
	I	31	31	0					
G	L	78	86	0	3	0			
H	L	88	88	0					
I	J	41	41	0					
	K	41	41	0					
J	L	48	86	37		34	30		
K	L	53	86	33		30	26		

Figure 53. Lag sheet (Cycle 3)

Cycle	Activities shortened	Possible shortening (weeks)	NIL (weeks)	Shortening (weeks)	Cost/week (dollars)	Cost increase (dollars)	Total direct cost (dollars)	Project duration (weeks)
0							755,000	100
1	D	5	7	5	300	1,500	756,500	95
2	H	15	3	3	1,500	4,500	761,000	92
3	C and H	4 and 12	7	4	1,750	7,000	768,000	88
4	G and H	15 and 8	26	8	4,500	36,000	804,000	80
5	NONE							

Figure 54. Summary sheet (Cycle 4)—final

whose duration could be extended might be less than the most favourable costs based on combinations involving expediting only. The more sophisticated linear programming techniques will mathematically determine such optimum combinations. However, they occur very infrequently, the cost difference is not usually significant, and the programmer who is aware of such possibilities can usually not arrive at the optimum solution unless there are many parallel critical paths with complex interrelationships. This is not an important shortcoming for the manual approach proposed here.

Activity	Cost slope (dollars/week)	Critical activities	Finish	Possible shortening (weeks)						
				Cycle 0	Cycle 1	Cycle 2	Cycle 3	Cycle 4	Cycle 5	Cycle 6
C	250	2	3	4			0			
D	300	0	1	5	0					
J	1,000			3						
I	1,200			5						
H	1,500	0	4	15		12	8	0		
K	2,000			4						
G	3,000	2		18				7		
F	4,000			1						
A	5,000	2		9						
E	6,000			2						
B	NONE	0	0	0						
L	NONE	0	0	0						

Figure 55. Selection sheet (Cycle 4)—final

Preceding activity	Following activity	EF-preceding	ES-following	Lag	Revised lags									
					Cycle 1		Cycle 2		Cycle 3		Cycle 4		Cycle 5	
A	C	19	19	0										
	E	19	26	7	-	2								
B	D	8	6	0										
C	G	22	33	0										
D	E	26	26	0										
	F	26	26	0										
	H	38	26	0										
E	G	30	33	3	-	8				4				
F	G	31	33	2	-	7			-	3				
	I	31	31	0										
G	L	78	86	0	-	3	-	0						
H	L	86	86	0										
I	J	41	41	0										
	K	41	41	0										
J	L	49	86	37			-	34	-	30	-	22		
K	L	53	86	33			-	30	-	26	-	18		

Figure 56. Lag sheet (Cycle 4)—final

EFFECTIVE APPLICATION OF TIME-COST TRADE-OFFS

The procedures outlined in the preceding section can be considerably simplified by manual application. The manual approach allows a degree of judgement and realistic data to be used that computer procedures based on sophisticated mathematical techniques do not permit. This is an advantage that far outweighs the possibility of obtaining a sub-optimum solution under the rather rare circumstances discussed in the last section. The following points will explain these claims:

(a) The procedure discussed so far in this publication and mathematical solutions in general are based on the assumption that the activities in a project are independent of one another except for the sequential relationships shown by the network diagram. From a practical standpoint this is often not the case. The measures taken to expedite one activity may affect others and in turn produce secondary cost effects that should be considered. Sometimes these cost effects need to be considered in the time-cost trade-off cycle being performed, and other times they should be considered when future cycles are being performed. In either case an analysis of the interdependencies is necessary during the intermediate steps of developing the time-cost curve. The manual, cycle-by-cycle calculation has a very real advantage because it permits a continual application of judgement and modifications of data throughout the process of developing a solution.

For example, a decision to work overtime or shift work to expedite a particular activity may have an effect on concurrent activities. It may become necessary from a practical labour relations viewpoint to put other crews on the same schedule. The resulting cost and time effects must be considered together with the original proposed change. Or, as a slightly different example, a large piece of equipment, such as a power crane, may be moved on to the job as a means for expediting a particular activity. It is very likely that this equipment, once it is made available, can be used on certain following activities also. The fact that its "move in" costs have been absorbed by the first activity means that cost slopes for expediting the later activities may be reduced. Sometimes in order to justify such a measure, the effects on more than one activity must be considered to arrive at the initial decision.

A closely related objection to any procedure that requires a complete set of input data is that it is unfair to expect an estimator to furnish such data because he does not have enough information. This applies in particular to estimates of costs for crashing each activity. As indicated in figure 41, the estimator is asked to determine the cost of crashing each activity, but he does not have any idea at what stage it will become attractive to expedite any individual activity. An activity may be one of the first selected for expediting at a point when the rest of the job is being performed on a five-day, single-shift workweek with a minimum of expensive equipment. Or, it may become attractive to expedite only after many other activities have been expedited and, perhaps, the rest of the project is on a seven-day, three-shift workweek and many pieces of specialized equipment have been moved in to speed other work. The cost of crashing the particular activity under consideration will be quite different in these two situations. Yet the estimator is asked to furnish the crash costs without knowing the project environment under which crashing is to take place. Such a demand is unrealistic and the data resulting are necessarily unrealistic. By not requiring a complete set of data, as discussed further under the next point, and by permitting modifications after each expediting cycle, the procedures of this publication do not demand or depend on unrealistic data.

(b) One of the major problems from a practical standpoint in applying existing time-cost trade-off procedures is that the amount of data required is

excessive. At least two time and cost estimates for each activity in the project have been considered a minimum requirement. This was the case with the data presented in figure 41 and in general is the requirement for all computer procedures. One major advantage of the manual procedure is that a complete set of data is not required. The only cost-slope data needed at a specific stage of the procedure are data for the activities that are critical at that point. These generally include a small proportion of the total activities. A competent estimator can usually eliminate the majority of these critical activities from detailed consideration by a rapid examination and then concentrate on a more detailed analysis of the few remaining ones. For example, in the network of figure 40 there were initially only four critical activities: Activities *B*, *D*, *H* and *L*. Two of these, *B* and *L*, were of such a nature that the estimator could eliminate them from consideration at a glance. Therefore, it would be necessary to examine and develop data for only two activities at the beginning of the time-cost trade-off procedure for the previous example. When later in the procedure Activities *A*, *C* and *G* became critical, these activities would need to be considered. Again the estimator would probably postpone a detailed analysis of Activities *G* and *A*, since he could tell very quickly that Activity *C* would offer the best possibilities for expediting. The importance of requiring only a minimum of data for effective application of time-cost trade-offs cannot be overemphasized. It is a tremendous advantage in implementing these potentially powerful procedures and it is an inherent advantage of the manual, cycle-by-cycle method of calculation.

(c) It was mentioned earlier in this chapter that the activity time-cost curve is often of the same general shape as the project time-cost curve. This is logical, since many activities can be expanded into networks of their own and be considered as projects. Such a curve was shown in figure 39 (b). For each activity with this type of curve there is a period of possible expediting at a cost slope considerably less than the cost slope determined by the two terminal points of the curve. If many activities are not expedited during the application of time-cost trade-off procedures because their cost slopes as determined by normal and crash performance appear to be excessive, many favourable opportunities for project cost improvement will have been overlooked. To achieve the greatest economies it is very important to base expediting decisions on the low cost-slope ranges of the critical activities rather than on the customary normal-crash cost slopes. This is seldom practical in computer procedures, since the data requirements for multi-segment activity time-cost curves would be prohibitive. As discussed under point *(b)*, the requirements for only two time-cost points per activity is often a serious obstacle to practical implementation. With manual, cycle-by-cycle calculations, however, there is no problem in taking into consideration the low-slope initial portions of activity time-cost curves. Generally a much more favourable solution will result from limited expediting of a number of activities instead of expediting a few all the way to their crash limits.

(d) Because of the almost infinite number of scheduling combinations the project time-cost co-ordinates may actually approach an almost smooth, continuous curve, but this is not so true for activity curves. They are often a series of discrete points that represent a limited number of alternatives. In some cycles

of the development of the project time-cost curve the amount of activity-shortening will be determined by the network interaction limit. If from a practical standpoint the activity must be shortened by an amount of time greater than the NIL in order to achieve project-shortening equal to the NIL, the true cost slope will be higher than that based on the assumption of a continuous curve. There may prove to be more attractive solutions based on expediting some other activity or based on the performance of that same activity by an alternative method producing an amount of shortening more nearly matching the NIL. This alternative method might not ordinarily be considered because it appears to have a steeper cost slope.

To illustrate, consider the conventional two-point activity time-cost curve *NC* of figure 57. The cost slope is apparently \$50 per day. Assume that the NIL for shortening this activity is 2 days. Therefore, expediting to the crash limit would produce only two days of project-shortening if the new zero-lag sequence line created a new critical path that would cause this activity to become non-critical. However, the cost increase would be \$400. The effective cost slope is \$200 per day. Other activities may offer more economical solutions for two days of project-shortening. Or there may be other discrete scheduling solutions for this activity, such as that corresponding to Point *A*. Point *A* represents a method of performance which, though its cost slope is twice that of Point *C*, provides two days of project-shortening for \$100 less (an effective cost slope of \$150 per day).

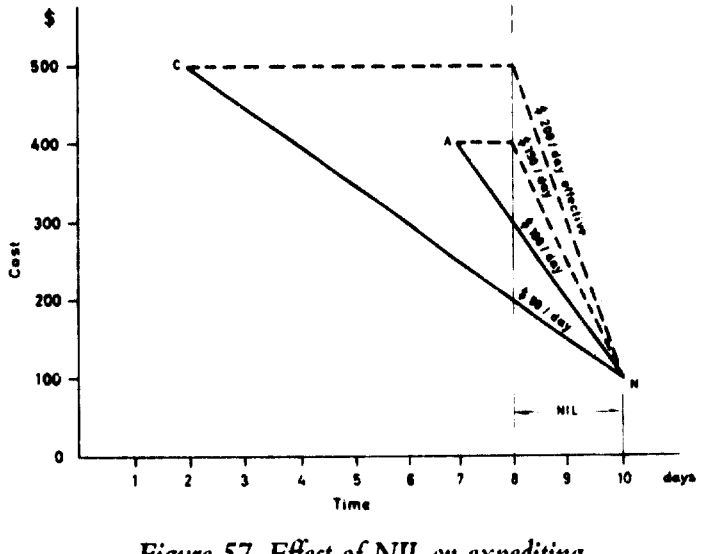


Figure 57. Effect of NIL on expediting

If curves are not truly continuous, cost slopes offer a rather poor criteria for selection of activities to be expedited. The manual, cycle-by-cycle method permits the programmer to calculate the NIL for a proposed change and to consider specific alternatives with relation to their respective NIL's. This will permit more intelligent time-cost trade-off decisions.

TIME-COST TRADE-OFFS FOR SUBNETWORKS

Time-cost trade-off analysis is ideally performed using the entire project network. Although it is possible to take advantage of the fact that only limited data for selected critical activities are required for application of the manual procedure, there are practical limitations on the size of network that can be analysed. The use of the dateline cut-off subnetwork, discussed in Chapter 3, offers an imperfect but helpful approach to larger networks.

This subnetwork approach is imperfect because not all of the opportunities for project-expediting occur within the subnetwork, and therefore some of them are not considered when decisions are made. It is both possible and probable that a more economical method of expediting is offered by activities outside the subnetwork, since the subnetwork represents a small portion of the over-all project. If there are multiple critical paths, the optimum expediting solution may involve the joint expediting of an activity within the subnetwork and another in a later portion of the project. By confining consideration only to the work within the period of the subnetwork, these optimum solutions are not obtained.

Imperfect solutions that result in improved, more economical schedules are better than no solutions at all. By establishing a value per time unit (e.g., dollars per day, per week or per month) for earlier project completion, a target is available for judging whether any proposed change will result in an improved schedule. If the change shortens the project at a lower cost than the amount saved by the earlier completion, even though it may not be the optimum improvement possible, it is advantageous. The chances of achieving near-optimum solutions are greatly increased by the ability to consider the low cost slopes of the initial segments of each critical activity time-cost curve. This increases the number of opportunities for expediting within each subnetwork and tends to reduce the differences between expediting opportunities within and without the subnetwork. It seems certain that an analysis based on a consideration of these low-slope segments but limited to activities within the current subnetwork will produce better results than an analysis based on a consideration of the entire network at one time but utilizing the assumption of two-point, single-slope activity time-cost curves.

Chapter 5

SINGLE-PROJECT RESOURCE ALLOCATION

Introduction

The execution of most project activities requires the use of various types of resources. Resources may consist of classifications of labour, such as management personnel, professionals and workers. Each of these categories could be subdivided; for example, workers could be classified as carpenters, qualified welders, or equipment operators. Resources may also consist of types of equipment, such as power shovels, tractors, cranes or trucks. In some instances resources may be types of material, such as concrete, or structural iron and steel. The money required to pay for work completed might even be treated as a resource in the procedures to be discussed. In most developing countries the resources required for the performance of project activities are usually in short supply. Sometimes these limitations are not serious, but often they impose restrictions that need to be considered if a realistic and intelligent project schedule is to be developed.

So far this publication has not dealt with resource requirements. A project implementation plan has been developed based only on satisfying physical restraints. Sequential relationships have been based on defining those activities that must be completed before others can be started. Activity scheduling has been based on these sequential relationships and routine calculations that also require estimates of activity durations. At no point has the question been asked, "Are the resources required for the performance of this activity available?" and, "If the required resources are not available, what action should be taken?" The answers to these questions are just as important in producing a realistic schedule as is adherence to physical sequencing restraints. It is true that certain very clearly defined resource restrictions may be taken into account while developing a project implementation plan even though these questions are not formally asked. The programmer may recognize, for example, that he has only a single pile-driving rig available and take this into account as he sequences the various activities that require the use of this equipment. But most resource problems are more subtle ones. They may involve a substantial number of units of a resource as well as a number of activities requiring this type of resource that can be scheduled concurrently. To determine a favourable scheduling for these activities that will not cause resource-availability limits to be exceeded is

frequently a difficult task. The development of procedures for this purpose is desirable.

The first step is to define the problem. The development of resource schedules for each important resource will clearly reveal the resource-allocation conflicts that exist. A resource schedule is a list of the units of that resource required during each time unit for the span of the project duration. To obtain resource schedules the estimator must furnish data giving the resource requirements of each activity. These requirements should be based on the method of performance that corresponds to the estimated duration of the activity. It is also necessary to have a schedule for the timing of the performance of each activity. A schedule based on starting each activity at its earliest start date offers a logical initial schedule for purposes of problem definition. Having resource requirements of each activity and a scheduling for each activity, the development of resource schedules involves a time-unit by time-unit examination of those activities in progress, the total of the number of units of each resource required during each time unit, and the entry of this total in the proper resource schedule.

As a sample problem, consider the project network shown in figure 58. Assume that there are three resources required for the performance of this project; these resources are in short supply and therefore scheduling restrictions on project activities may arise. These resources are designated as Resources *X*, *Y* and *Z*, and the requirements for each resource by each activity are shown in the diagram under the node symbols. Scheduling data for this network has been calculated, and the results are shown in the tabulation in figure 59. Project duration without consideration of resource restrictions is 34 weeks. A schedule

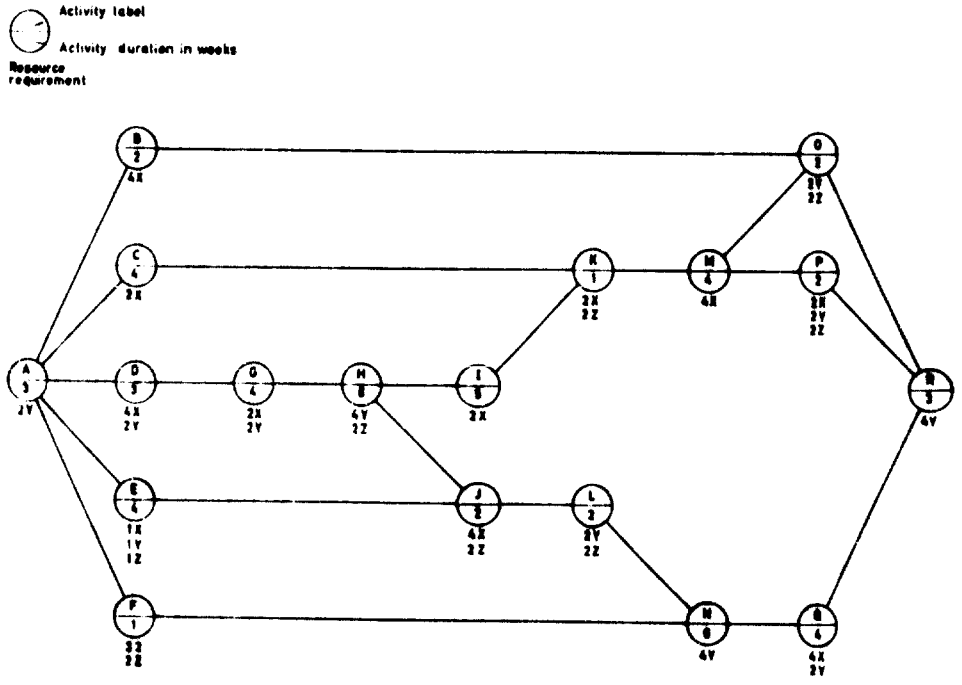


Figure 58. Sample network with resource requirements

Activity	Activity duration (weeks)	Critical activity	ES	EF	LS	LF	TF	FF
A	3	*	1	4	1	4	0	0
B	2		4	6	27	29	23	21
C	4		4	8	20	24	16	14
D	3	*	4	7	4	7	0	0
E	4		4	8	13	17	9	9
F	1		4	5	21	22	17	17
G	4	*	7	11	7	11	0	0
H	6	*	11	17	11	17	0	0
I	5		17	22	19	24	2	0
J	2	*	17	19	17	19	0	0
K	1		22	23	24	25	2	0
L	3	*	19	22	19	22	0	0
M	4		23	27	25	29	2	0
N	6	*	22	28	22	28	0	0
O	3		27	30	29	32	2	2
P	2		27	29	30	32	3	3
Q	4	*	28	32	28	32	0	0
R	3	*	32	35	32	35	0	0

Figure 59. Scheduling data for sample network

based on starting each activity at its earliest start time is shown in figure 60, and the resulting resource schedules for the three resources are developed in this figure. These resource schedules can be visualized more easily in the graphic representation shown in figure 61. The resulting schedules are not attractive ones owing to excessive peak requirements and to large, frequent variations in the requirements. This is especially true for Resource *X*, which has a peak requirement of 14 units, but a few weeks later the requirements drop to zero.

Two principal problems exist with regard to resource utilization. One occurs when the demand for resource units exceeds the supply. Since such a demand cannot be satisfied, one or more of the activities responsible for the demand will have to be rescheduled. In other words, the peak requirements have to be reduced until they no longer exceed availability limits. The principal objective is to accomplish this in such a manner that the project duration is not

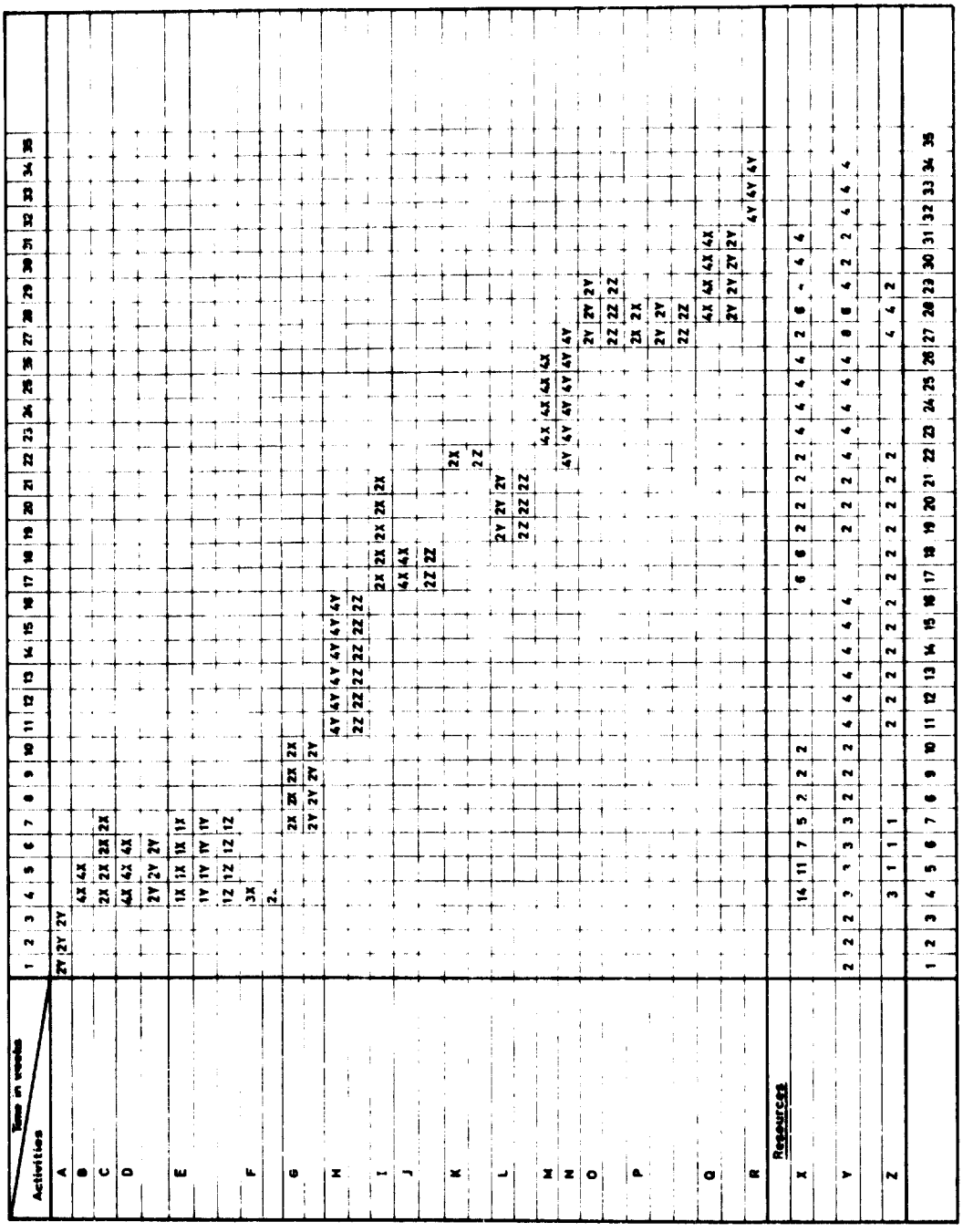


Figure 66. Development of resource schedules based on earliest start times

increased or, if this is not possible, is increased by the absolute minimum. The second problem involves the number and magnitude of the variations in resource requirements. In general, these variations cost money, and the larger they are the more they cost. A decrease in resource requirements followed by an increase represents an expense. In the case of labour, it involves laying off men with the risk of losing them permanently and later incurring the expense of obtaining and retraining new men. The alternative is to retain them on the payroll without efficiently utilizing their time. In the case of equipment, there is the cost of

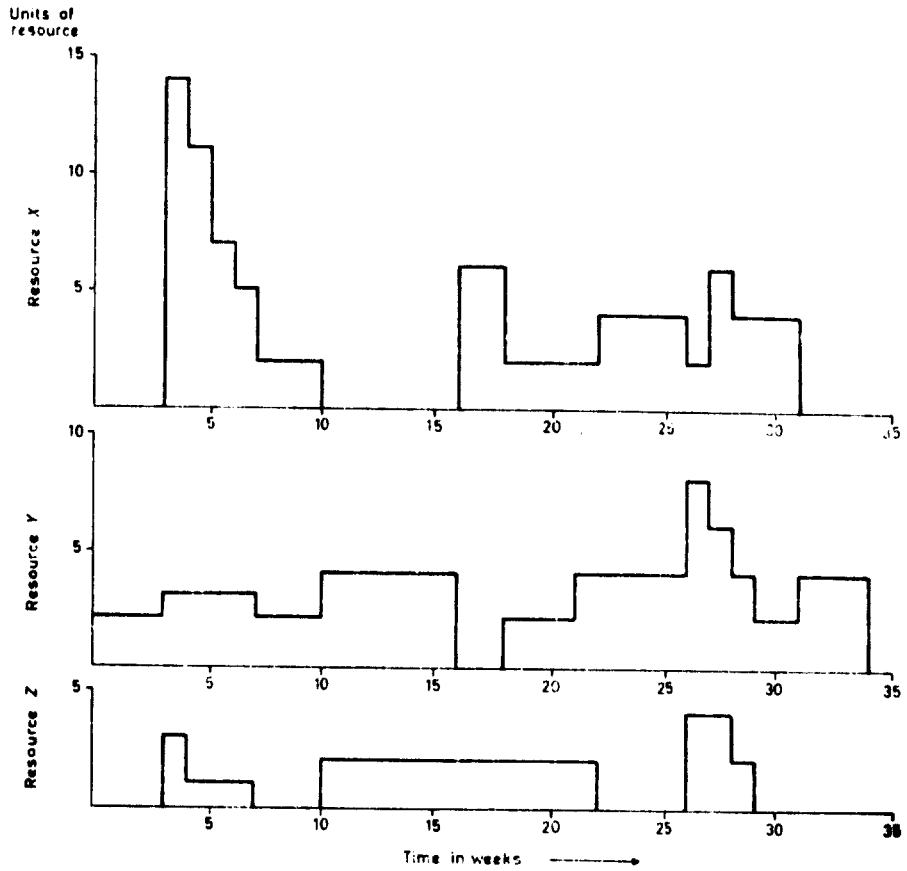


Figure 61. Resource schedules in chart form – early start

moving it off the project site and then moving it back when it is needed again. The alternative here is to absorb the ownership or rental costs while the equipment stands idle. All these measures tend to increase costs. The ideal resource schedule would be represented by a curve that built up and tapered off gradually, was constant over its intermediate range, and never exceeded availability limits.

GENERAL APPROACHES TO RESOURCE ALLOCATION

Developing a schedule to satisfy resource restrictions can be a very complex problem. Even for small networks and for processing by large capacity computers, there are so many scheduling combinations possible that it is not practical to apply procedures that theoretically would guarantee an "optimum" solution. There are a number of very sophisticated resource-allocation programmes that require large computers and considerable running time and which, hopefully, provide near-optimum solutions. Many of these programmes are proprietary ones. The problem is complicated because of the amount of data involved, the probability that a considerable amount of these data will have to be updated as a result of the interactions that occur when activities are rescheduled, and the almost limitless number of possible rescheduling combinations. Therefore, it

is generally claimed that computer approaches to this problem are essential because of the magnitude of the data-handling operations required.

Consider again the resource schedules developed in figure 60 for the sample network. The development of such schedules is not a new concept, nor is it dependent on network techniques. Such schedules can be developed from bar charts as long as resource requirements are provided by the estimator. Careful programmers have developed resource schedules for important projects for many years, but it has not been a common practice. The reasons apparently are that a great deal of data must be furnished and much time and effort expended. These are not particularly good reasons for not developing resource schedules, since the time and effort are generally profitably spent. But the fact remains that project management often chooses to proceed quite blindly with respect to resource problems because of the trouble involved in developing even a single set of resource schedules.

If the initial resource schedules were developed and if scheduling changes were subsequently proposed, each such change might require extensive modifications. If a critical activity were rescheduled or if a non-critical activity were rescheduled in its interfering float range, other activities would also have to be rescheduled. These activities may involve a number of different resources. The original change that was aimed at a specific improvement in one portion of one resource schedule may produce many changes in other portions of that resource schedule as well as changes in several other resource schedules. Some of these changes may be unfavourable and may even nullify improvements that have been previously achieved by considerable effort. In view of these problems it seems unlikely that project schedules providing good solutions for meeting resource limitations can be developed successfully by a manual procedure. Yet the purpose of this chapter is to show that such is the case.

The fact that most resource programming at present is accomplished by non-computer methods – in some cases poorly but in other cases satisfactorily – provides evidence that such methods are practical. At one extreme there are very elaborate computer-oriented resource-allocation programmes, but these account for a very small percentage of total resource programming. At the other extreme there is a complete lack of advance programming, and problems are solved as they arise. For example, on an ordinary construction project there are certain units of equipment on the site and a certain number of men report for work each morning. If there is insufficient equipment or an insufficient number of men to perform all the work that is ready to be done, some work must be postponed. Choosing what to postpone is often left to the craft foreman, and decisions are based on his judgement and intuition rather than on any precise knowledge of criticality and float times for activities. If there is more equipment or a larger number of men than needed, it is the foreman who decides which equipment to use or, in the case of the men, whether to lay them off. No matter how crude this process may be, it constitutes a resource-allocation technique.

Both extremes of practice have certain advantages and disadvantages. The computer can handle the large amount of data and computational steps that are

routinely required and can apply mathematical procedures that are completely prohibitive in manual processing. On the other hand, it can only consider implementation programming and scheduling modifications that it has been programmed to perform and for which it has been supplied data. It is impossible, in practice, to programme a computer to consider all alternatives that might be possible and desirable in specific situations or to furnish it with all the data that it might possibly make use of. Another disadvantage in some locations is that resource-allocation programmes require very large capacity computers that are not always readily available.

The disadvantages of the other extreme of practice are that decisions are made at the last moment, when the opportunity for favourable solutions may already have been lost; they are made without benefit of data that could be helpful; and they are often made by a level of management that may not be the best qualified to make them. One advantage of a lack of advance planning is that decision-making does not require computer equipment that is not available. However, regardless of the availability of computer facilities, an even greater advantage is that the exercise of judgement is permitted. There are many possible ways to resolve resource conflicts. One is to reschedule activities. This is the principal, and often the only, basis for most formal procedures. Other measures, however, can be just as effective. Activities can be replanned to change their resource requirements. When an estimator is required to furnish resource data, he must provide a list of resource requirements in specific numbers. Generally he can alter these requirements or even eliminate them by adopting a different method of performance. Sometimes activities can be interrupted temporarily or resource units can be borrowed from other activities without completely interrupting them. Sometimes resource units can be shared by concurrent activities. The clever foreman who is faced with a shortage of resource units will often consider all these alternatives and a few more. He has an ability that cannot be programmed into a computer and that allows him to obtain acceptable results even though, on analysis, the method appears crude.

This chapter proposes a middle course of action. It is assumed that computer facilities are not available and therefore a manual procedure is required. This manual procedure should take advantage of basic network scheduling data and the application of network mechanics as far as possible. Such an approach cannot be as simple as pushing a button and awaiting a printout of a solution; it requires a considerable amount of experimentation, calculation, and analysis. When a prolonged effort is to be expended in the implementation and management of an important project of long duration, this effort is justified. When it is necessary to programme many projects of very short duration, only computer processing can provide practical solutions to resource-allocation problems. This publication, however, is concerned with single projects or programmes involving a limited number of projects that extend for months or years. Project management, therefore, can afford to take the time required by the procedures this publication proposes. Procedures for three cases, each of increasing complexity, are presented below.

MANUAL RESOURCE-ALLOCATION PROCEDURES - PREPARATORY STEPS

An initial step for these procedures is to develop resource schedules for one specific project schedule which, except for resource requirements, would be a feasible one. This step is not essential in the procedure outlined for Case II below, but it is helpful there also. The simplest schedule is that based on the earliest start times of each activity, as shown in figure 60. In the procedures presented here activities are rescheduled, or other changes are made that may require rescheduling of activities, in an attempt to eliminate unacceptable resource requirements and improve resource schedules. As a result, total requirements for each time unit are frequently changed.

If the mechanics of physically making changes in the data tabulations are simplified, the possibilities for improving schedules will be expanded. A method for accomplishing this is to use a worksheet consisting of a large piece of cross-section paper mounted on sheet metal. A time scale is shown horizontally across the sheet, with each column of the grid representing one time period in the time units chosen. Each activity is represented by a bar cut from a strip of magnetic plastic. Such plastic is inexpensive and comes in rolls of stripping, in widths as narrow as one quarter of an inch; it may be painted with enamel paint. A wax pencil or a pen using water-soluble ink is used to mark resource requirements on each activity bar. These entries should be positioned to match the horizontal scale of the grid sheet so that a single entry will appear in each time-unit column over the time span of the activity. It is not essential to cut the plastic bars to match the duration of the activity. A module, such as five time units, is chosen and all bars cut the same length from the rolls of stripping material. The bar for any activity is made by placing these basic modules end to end and leaving any extra space on the final length blank. It is also helpful, if the number of different types of resources to be analysed is not too great, to use a different colour bar for each resource. The magnetic strip material can be painted in at least six to eight different pastel shades. This will permit simple marking with pencil or pen and will enable the origins of the various resource requirements to be easily seen. If an activity requires more than one resource, a different colour bar for each resource is placed one above another to form a composite bar for the span of the activity. This group of bars is moved as a unit when rescheduling of the activity is performed. Strips of magnetic plastic are mounted across the bottom of the worksheet to receive entries of the totals for each time period. If colours are used to represent different resources, one strip of each colour spanning the entire project duration is used for the totals. Entries are made in a manner similar to those for the activity bars, i.e., with grease pencil or water-soluble ink. As activities are rescheduled these entries can be easily changed by carefully wiping out the previous figure and entering the new total. This arrangement permits making changes in a more practical manner than by making entries directly on the cross-section paper with subsequent extensive erasing.

An additional refinement to the above procedure to allow experimentation with changes that involve shifting several activities is possible. Sometimes a decision concerning the desirability of a proposed change cannot be reached until

rather extensive changes have been made. In this case an extra blank strip for each resource is placed across the bottom of the worksheet adjacent to the strip on which totals per time unit are entered. The bars for each activity involved in the proposed change are left in their original position and merely turned over to eliminate them temporarily from consideration. New bars are placed in the new proposed positions (pieces of magnetic strip can be laid on top of portions of the turned-over strips, so extra vertical space need not be provided). The new totals are entered on the extra blank strips at the bottom of the worksheet after the proposed change has been completely processed and all affected activities have been scheduled in their new positions. These figures can then be compared with the previous totals, which have not been destroyed, and are directly above them. If the change does not prove desirable, the new data are removed and the turned-over bars restored to their former positions. If the change proves to be an improvement, the turned-over bars are removed, the entries on the original totalling strips on the bottom of the worksheet changed to match those on the spare strips, and the spare strips blanked out for reuse.

Case I—Limited resource restrictions

Sometimes the resource restrictions are not severe and the elimination of major peak requirements can be achieved by a relatively simple shifting of activities, mostly within free float ranges. This shifting can be accomplished by moving the magnetic bars representing the activities to a new position and changing the totals affected at the bottom of the worksheet. If an activity is shifted within its free float range, there are no chain effects, so no other activities need be shifted. This is the simplest possible type of change and yet can be quite effective in many situations. Of course activities can also be shifted in their interfering float ranges if the affected activities are also shifted and all necessary totals corrected. If the chains of affected activities are short ones, not much effort is required, and such changes may also be practical to process. Critical activities can also be shifted and other activities shifted beyond their float limits if the improvement produced is worth extending project duration. However, unless such changes occur near the end of the network diagram, there are likely to be many activities affected and a large amount of data modification required. Therefore, scheduling changes for solving resource problems by procedures for Case I are limited primarily to changes involving the use of free float time and changes that do not require an excessive amount of activity-shifting.

Figure 62 shows a rescheduling of the activities of the project network of figure 58. Only activities with free float have been rescheduled from their earliest start times, and rescheduling has been limited to the float ranges. The vertical lines opposite each activity indicate the possible time span for rescheduling by marking the earliest start time and the end of the activity's free float time. In this rescheduling, project duration has not been changed, of course, and no activities have been interrupted or replanned to change resource requirements. The improvement, nevertheless, is impressive. The maximum requirement for Resource *X* has been lowered from 14 units to 6 units. Those for Re-

sources *Y* and *Z* have been reduced respectively from 8 to 6 and from 4 to 2. The six-unit requirement for Resource *Y* could easily be reduced further to 4 units by extending project duration by one week as is apparent from a quick examination of the worksheet. The continuity of the use of resources has also been improved. Once Resource *X* is required, it is used continuously, in varying amounts, until it is no longer needed. Resource *Z* builds up to its maximum, which then remains constant except for a single four-week break, as compared to two breaks and a less uniform requirement in the initial schedule. The schedule of figure 62 is definitely an improvement over the schedule of figure 60 from a resource standpoint, and yet the schedule modifications were very simple to make.

It is not necessary or desirable to develop strict rules or a definite procedure for rescheduling of this type, since the previously suggested trial and error juggling of data on magnetic strips can be accomplished so easily. Whether the best possible answer is obtained depends on the skill and imagination and luck of the person in charge, but usually an initial schedule can be considerably improved. If the programmer considers measures such as replanning activity performance, borrowing resources temporarily from concurrent activities, varying resource use during the performance of an activity, interrupting activities, sharing resources between activities, and a host of other possibilities, he can probably greatly improve the original schedule. For example, in the schedule shown in figure 62 it might be possible to replan Activity *I* for performance in 3 weeks using 4 units of Resource *X* per week. This would increase the resource-weeks of effort from 10 (5×2) to 12 (4×3) to allow for some inefficiency in the alternative method. But it would improve the resource requirements for the period from the seventeenth through the twenty-first week to give a 4—4—4—4—4 schedule instead of a 6—6—2—2—2 schedule. Such changes can be made frequently if the need for doing so is brought to the attention of the programmer and if he is familiar with the method of performance of the work involved.

Case II—Different resource restrictions

Satisfying resource restrictions is sometimes more difficult than in the previous example. Rescheduling activities within their free float ranges may not accomplish the necessary reductions. Other rescheduling often involves shifting several activities, and if many changes are necessary an experimental, trial-and-error procedure is not very practical. Since making changes that involve several activities is difficult with a manual procedure and imposes practical limitations on the number of scheduling improvements that may be attempted, the problem of improving project schedules that are unacceptable and that are subject to severe resource restrictions arises. A method of approach different from that of Case I is called for. Instead of starting with an initial schedule and attempting to reschedule activities within it, the programmer starts at the beginning of the project network and gradually builds an acceptable project schedule activity by activity. Rescheduling of activities is confined, at any moment, to

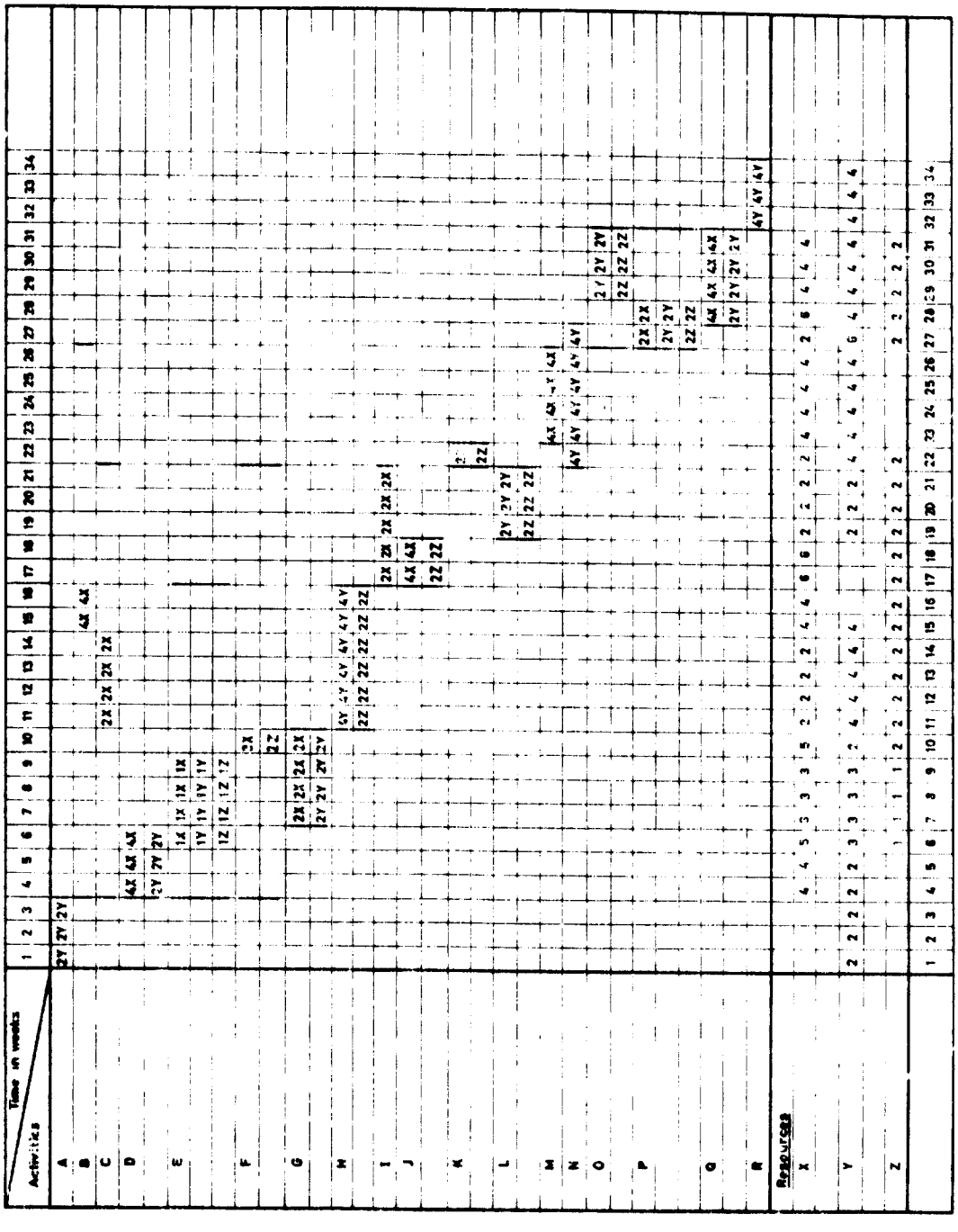


Figure 62. Rescheduling within free float ranges

the activity then being scheduled and possibly to a very limited number of activities that are in conflict with it for available resource units. In this procedure an activity is not scheduled for performance until resource requirements can be satisfied. If this requires postponing performance beyond its float period, project duration is extended. It is not uncommon to resort to this measure; when resource restrictions are severe, project completion usually must be delayed. The objective of the procedure becomes that of minimizing the extension of

project duration while satisfying all resource requirements within the limits of resource restrictions.

In this procedure three principal questions must be answered:

(a) What limits should be established for resource availabilities?

(b) What is the basis for arranging the order of activities preparatory to scheduling them?

(c) By what method should conflicts over resources be resolved?

The question of establishing levels of resource availabilities may be considered first. The procedure for Case I involved starting with an existing schedule and attempting to improve it until resource requirements are considered acceptable. What "acceptable" means is often somewhat flexible and may depend on the amount of effort required to make further improvements. In any case, the decision on final resource limits can be made after a certain amount of rescheduling has been accomplished. In the procedure for Case II resource limits must be established at the very beginning of the process, before any activities are scheduled. One basis for establishing these limits is simply to state how many units of each resource are actually available. Theoretically this is a difficult figure to define, since there are almost always some measures possible that will produce additional resource units if funds are available. If a criterion of "normal maximum" is used, limits can be established and the development of a schedule can commence. Later, if undesirable delays in project completion become necessary because of insufficient resources of a specific type, this resource limit can be reviewed and the premium costs of extra units examined. If a decision is made to increase the limit beyond that which originally appeared reasonable, it may or may not be desirable to repeat the scheduling procedure up to that point. This would depend on the extent to which the project has already been delayed because of the resource limit that is being revised.

The procedure for Case II does not "level" resource requirements within the maximum limits. The scheduling of activities is based only on satisfying established limits rather than on eliminating peaks and valleys in the resource-schedule curves. By progressively lowering the limits and repeating the procedure, effective levelling will be accomplished. The problem with this approach is that each scheduling pass requires much effort, and iterative methods are not particularly suitable for manual processing. It is often desirable, therefore, to set original limits that are more severe than those based on "normal maximums" in order to force additional levelling and to keep resource peaks as low as practical. If such limits are set a good possibility should exist that they can be met without excessive delays in project completion. In general, the tighter the resource limits are, the longer the project duration will be. The extreme limit on the tight side would be to set limits equal to the maximum requirement of any single activity that uses the resource whose limit is being set. The project cannot be performed unless there are at least as many resource units as are required by any single activity. This method of establishing resource limits will be applied to the sample project network of figure 58. Resource limits of 4, 4, and 2 will be set for Resources *X*, *Y*, and *Z* respectively, since these are the maximum number of units required by single activities in the project. For example, Activity *B* requires

4 units of Resource *X*, and Activity *H* requires 4 units of Resource *Y* and 2 units of Resource *Z*. This will offer a problem that has the tightest resource restrictions possible.

Another approach is to develop the resource schedules for only the critical activities based on the earliest start time schedule. The limits can then be set equal to the maximum requirements in each of these schedules. This provides a mathematical possibility of meeting the limits without extending project duration, although this possibility is remote. In the example being used, this approach would produce the same 4, 4 and 2 limits determined above. One other approach offers limits of intermediate severity and is sometimes useful. It uses the resource schedules that have been determined for the project schedule based on the earliest start times for all activities. The total number of resource units per time unit can be determined for each resource over the entire span of project duration. This total can be divided by the number of time units in the period in which the resource is used (including brief periods of zero use) to obtain average requirements. Then the average requirements can be multiplied by a suitable, arbitrary factor (greater than 1) to establish working resource limits. For example, in figure 60 there are a total of 99 resource unit-weeks required for Resource *X* over a period from the fourth to the thirty-first week. This is an average requirement of 99/28, or 3.25 units per week. Using a factor of 1.5, a total of 5 resource units ($3.25 \times 1.5 = 4.88$) would appear to be a reasonable limit, provided that it is at least as large as the requirement by any single activity and does not exceed the limit set by a normal maximum figure. In a similar manner limits of 5 ($(108 \times 1.5)/34 = 4.77$) for Resource *Y* and 2 or 3 ($(40 \times 1.5)/26 = 2.31$) for Resource *Z* might be established.

Once resource limits for each resource have been determined by one of the foregoing methods, the second problem is to establish an order for the scheduling of the activities. One possibility is to proceed a time-unit at a time and for each time-unit to consider all the activities available for scheduling. This is a good procedure, but it is a more tedious one to apply manually than a procedure that schedules on an activity-by-activity basis. If activities are to be scheduled in some systematic order, it seems logical to schedule those that are more critical before scheduling those that can also be started at the same time but that have greater scheduling flexibility. Activity latest start times or latest finish times can furnish a basis for selecting activities according to both criticality and a progressive ordering from start to finish in the project network. Latest start times have been chosen for the procedure for Case II. This favours scheduling the activity with the longer duration first, since it becomes more difficult to schedule the longer duration activities after others have been scheduled.

Figure 63 shows the activities listed in ascending order of latest start times. They may be listed in any order as long as they are scheduled in the order according to their latest start times. Extension of project duration will not affect this order as long as scheduling progresses in the order established by the original latest start times. Although latest start times change when project completion is changed, all activities yet to be scheduled will have their latest start times shifted by the same amount; hence, the order is not affected. As activities are

about to be scheduled, it is helpful to mark their possible scheduling range on the worksheet. The initial limit is determined by the completion times of the preceding activities that have already been scheduled. The final limit is determined by the updated latest finish time, which is equal to the original latest finish time plus the delay in project completion that has been incurred up to that point. These date limits have been marked on figure 63 just prior to the scheduling of the corresponding activity (note the pairs of vertical lines opposite each activity). The activity is then scheduled at the earliest date at which its resource requirements will not cause the totals at the bottom of the sheet to exceed the resource limits that have been previously set. If an opportunity for better resource scheduling or avoidance of a delay in project completion can be achieved by interrupting an activity, the programmer should investigate the method of activity performance to judge whether this is possible. If in order to schedule the activity it is necessary to complete it at a date later than its latest finish time, the delay in project completion is noted in column 3 of the worksheet as shown in figure 63. This allows updating of the latest finish times of subsequent activities and also allows later analysis of the reasons for the final resulting increase in project duration. This analysis may lead to decisions to change resource limits and repeat the procedure.

The final problem in the procedure for Case II is that of resolving resource conflicts. When an activity cannot be scheduled at its earliest start time because of lack of available resource units, it can be postponed. As long as it has float time this is not a serious matter; but when it must be scheduled to finish after its latest finish time, project duration is affected and a more serious problem has arisen. The simplest procedure would be to schedule each activity in order at the earliest date possible regardless of when this occurs and proceed to the next activity. This procedure will furnish a final schedule that meets all resource restrictions and with a minimum of effort. But a study of alternative methods of resolving resource conflicts as they arise, if they do involve the project completion date, can produce a more efficient schedule. Some of the activities already scheduled that require the same resource may be rescheduled instead of the activity under current consideration. If this will lead to less delay in project completion, it is a better solution; other alternatives may also be considered.

It would be possible to establish a formal set of rules for considering the activities in conflict for the scarce resources and determining which should be rescheduled and how. This would result in a mechanical procedure that could also serve as the basis for a computer programme. The major advantage of a manual procedure is that it offers the possibility of utilizing the full scope of the programmer's judgement, a possibility that cannot be programmed into a computer. So a formal procedure will not be presented here for resolving resource conflicts. An examination of the data may suggest rescheduling one or more activities previously scheduled rather than postponing the activity currently being scheduled. It may suggest interrupting and rescheduling a portion of an activity, or it may suggest replanning certain activities to change their resource requirements (and probably their durations). Sometimes it may be possible for the current activity to borrow resource units temporarily, and hence

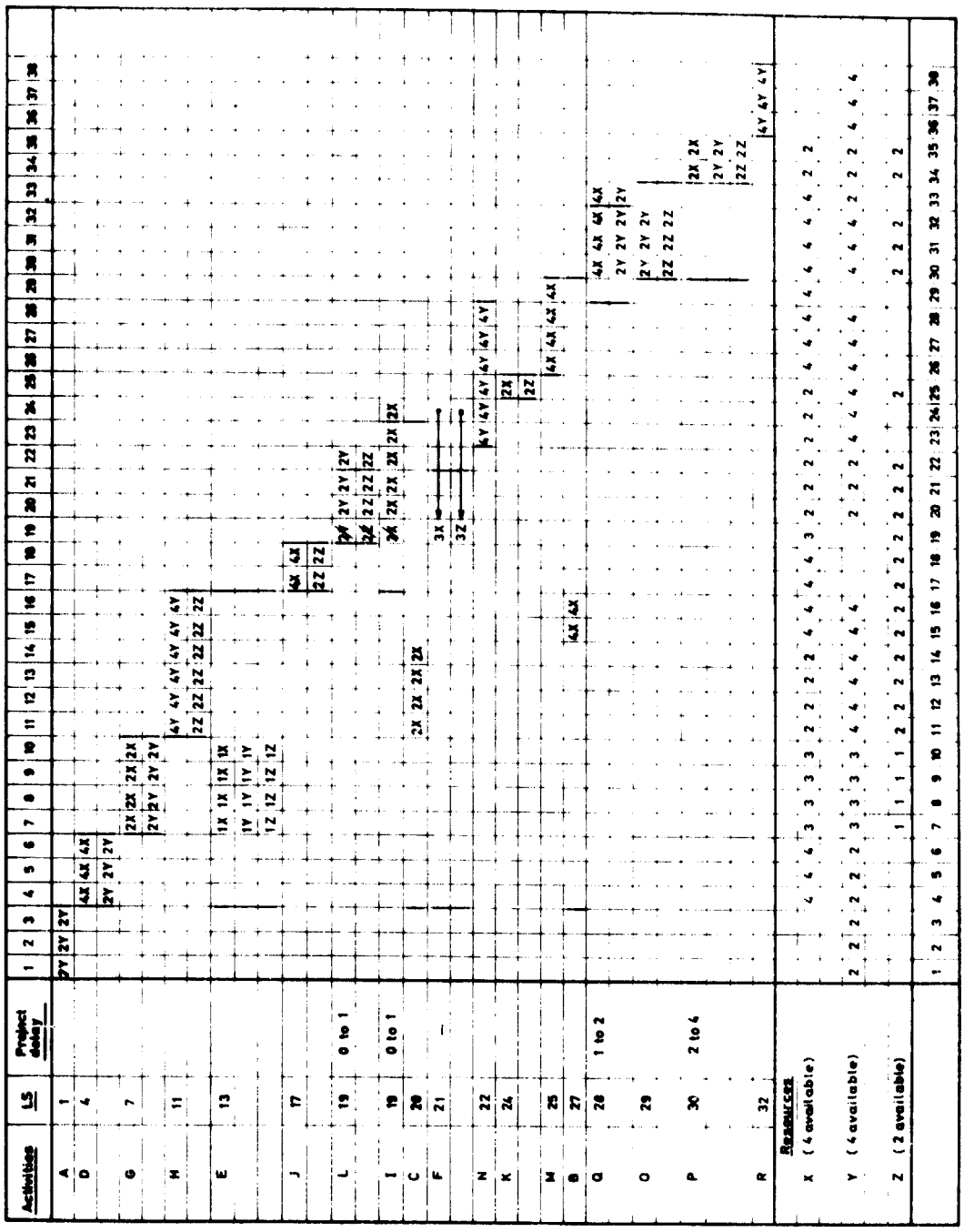


Figure 63. Rescheduling to meet resource restrictions

non-uniform resource requirements over the duration of certain activities can be considered. Sometimes an on-the-spot decision can be reached to change the resource limits; there are many possibilities. If no better alternative can be found than scheduling the current activity at the earliest date that will satisfy resource restrictions, then this can always be done. It is true that this procedure for Case II is not mathematically satisfactory, since it is not based on a definite set of rules. But from a practical standpoint a skillful programmer may often be able to obtain better results than those provided by the most elaborate resource-allocation computer programmes.

In the example shown in figure 63, when scheduling has progressed to a consideration of Activity *F* it is found that the earliest that Activity *F* can be scheduled is the twenty-fourth week, three weeks later than its latest finish date. An examination of the schedule already developed indicates that by rescheduling Activities *L* and *I*, which had previously been scheduled to start on the nineteenth week, one week later, Activity *F* can be scheduled on the nineteenth week. The net result is to delay project completion by one week instead of three. Later scheduling conflicts involving Activities *Q* and *P* cannot be resolved by means other than scheduling those activities at the earliest available date permitted by resource limits. The final project duration required to satisfy resource restraints is 38 weeks. If all activities had been scheduled at the earliest date at which resource units were available (Activity *F* is the only one in this example that was not), the project duration would have been 39 weeks. Figure 64 shows in graphic form the results obtained by the procedure for Case II that were presented in figure 63. Minor improvements can still be achieved if desired. For example, the discontinuity in use of Resource *Z* on the thirty-third week could be eliminated by scheduling Activity *O* a week later. A comparison of figures 61 and 64 indicates the improvements that have been achieved.

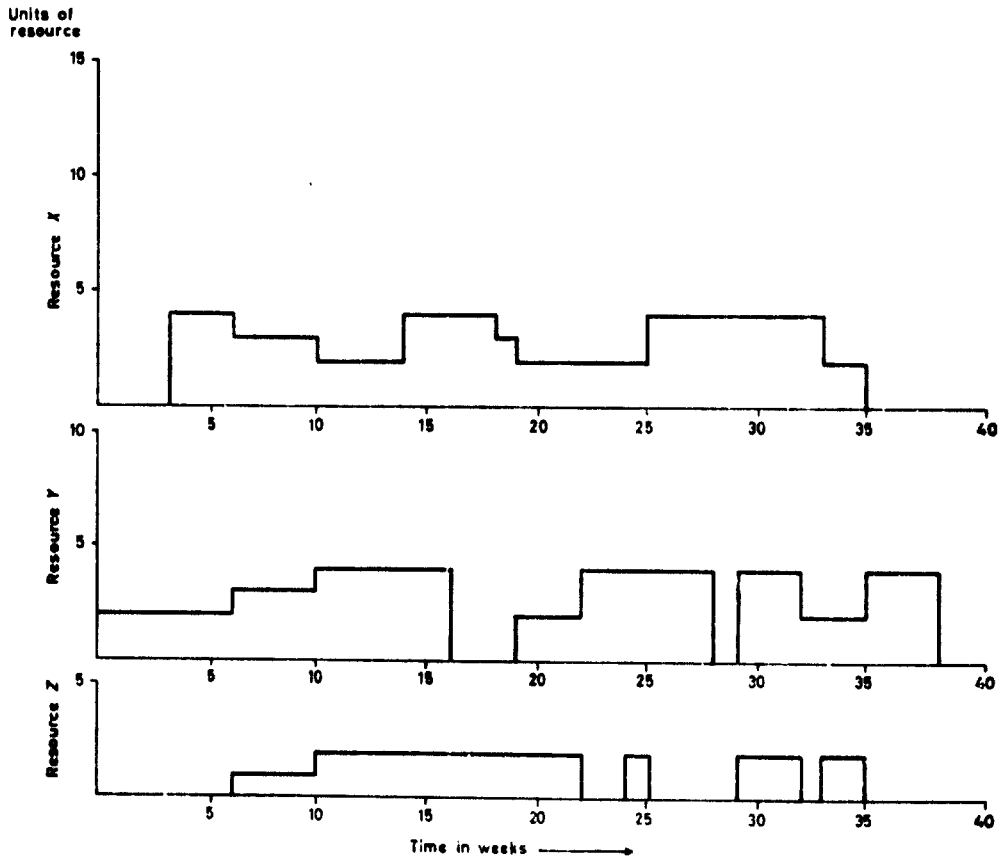


Figure 64. Final resource schedules in chart form

Case III – Projects of long duration

Projects of long duration require more effort for resource-allocation scheduling principally because of the many time units involved and the many entries to be made and updated. Very detailed resource programming for that portion of a project to be performed in the distant future is of questionable value. Such programming will undoubtedly have to be repeated because of changes in the performance of the earlier portion of the project. A combination of the procedures for Case I and Case II together with the use of subnetworks created by the dateline cut-off method discussed in Chapter 3 offers a means of resource programming for projects of long duration.

Initially the resource requirements of the entire project should be analysed to locate major problems and to determine measures for meeting them. Only those resources should be considered that are expected to offer definite limitations either because they are scarce or expensive or because the project activities make extensive use of them in relation to their levels of availability. For these activities it is generally worth while to develop resource schedules based on earliest start times. An added short cut that can often be used effectively is to base the resource schedules on a larger time unit, for example, weeks instead of days, months instead of weeks, or two or more months instead of one month. This preliminary study will help to establish reasonable resource limits to use in the more detailed scheduling procedures that follow, and it may also indicate the need for some immediate reprogramming of the project to eliminate major resource problems. If such reprogramming is necessary it should be done at this time, since it will probably extend project duration. If a reasonably valid project completion date can be determined in the preliminary analysis, subsequent effort to maintain in the detailed analysis a project duration that will eventually prove to be completely unrealistic will be avoided.

After the initial analysis has been completed, detailed schedule development using the procedure for Case II can be performed on subnetworks that cover a limited time period in the immediate future. In these studies additional resources can be considered because subnetworks can accommodate the additional data to be handled. A shorter time unit may be used as the basis for developing the schedule. When advantageous, the network activities can be further subdivided to expand the project plan in more detail. The effects on project completion of rescheduling any activity will be known by the amounts by which updated activity latest finish times must be exceeded. The effects on the preliminary resource schedules for the period ahead of the current subnetwork will not be known until subnetwork development has progressed to the period of interest. If extensive changes are made during the programming of subnetworks and there is reason for concern about major effects on the over-all project schedule, the procedures for this over-all study may be repeated. The major effort once subnetwork programming is commenced is confined to very detailed programming for a limited period ahead. This type of programming is practical and profitable with this subnetwork approach, but this would not be true if it had to be accomplished using the entire project network.

The procedures outlined in this chapter should permit effective resource allocation for any major project of extended duration even though these procedures have been restricted to non-computer methods capable of being applied at the project management level.

Chapter 6

MULTI-PROJECT RESOURCE ALLOCATION

THE MULTI-PROJECT PROBLEM

An industrial development programme may involve more than one project, and these projects may be in progress at the same time. Because resource limitations can affect the scheduling of activities in these projects, resource-allocation techniques should be applied. Where each project draws its resources from its own resource pools, the single-project resource-allocation procedures of Chapter 5 can be applied. If the projects draw their resources from a common pool, a co-ordinated effort is necessary to resolve conflicts for scarce resources in a manner that benefits the over-ail programme. A procedure for multi-project resource allocation is in this case desirable.

The multi-project resource-allocation problem is basically the same as the single-project problem, since project networks can be treated as subnetworks and combined into a single master network for the entire programme. The resulting problem is more complex if only because of its much larger size. However, there are other factors that add to its complexity. A principal one is that the different projects generally have different priorities. It may be much more serious to the effective implementation of the over-all programme to delay one project rather than another. Priorities must be taken into account if there are valid reasons for having them.

Another factor that is sometimes involved is that of mobility of resources between projects. There may be complete interproject mobility of resource units without problems, or each move of resource units from one project to another may represent considerable effort, time and expense. Which situation exists will usually depend on the relative geographical location of the projects and the degree of mobility of the resource itself. For example, one extreme occurs when multi-project resource-allocation techniques are applied to the scheduling of the engineering and pre-construction phases of several projects within a programme. The resources may consist of various classifications of professional and skilled personnel, often employed under the same roof. In this case there is complete mobility of the resource units. A shift to a new project may only involve exchanging one set of plans for the next set. The other extreme, however, may arise in the construction phase of the same projects if the projects are situated at widely separated locations and each may require certain types of heavy equipment that are in very short supply. Moving units of equipment

from site to site involves a major expense as well as time, and this must be considered in scheduling the activities of the projects that require these moves.

There are many other ways in which multi-project resource allocation can reach still higher levels of complexity. Some project resources may be drawn from common resource pools of the over-all programme while others come from pools available to a single project. Even more complicated combinations arise where projects within a programme share pools for certain resources with some other projects and share different pools with others. Similarly, certain resources may have interproject mobility with respect to some projects but not with respect to others.

It is the purpose of this chapter to suggest modifications of the single-project resource-allocation procedures of Chapter 5 that will cover these additional complexities. It is beyond the scope of this publication to develop detailed procedures for the multi-project case or to present a detailed treatment of its problems.

PROCEDURE FOR MULTI-PROJECT RESOURCE ALLOCATION

The projects in a programme that draw resources from a common pool should be considered jointly to determine the most favourable resource allocations for the entire programme. The basic scheduling data for each project (earliest and latest start and finish times) can be computed separately. If not all the projects are to begin at the same time, the starting times of their initial activities should each be related to the over-all programme time schedule. For example, if Project *A* is to start 90 days after the beginning of the first project in the programme, the earliest start time of its initial activity should be set equal to 91 before the forward-pass calculations are begun.

If the projects have different priorities in the opinion of the programme management, these priorities should be expressed in a quantitative manner, even though it is not always simple, nor perhaps diplomatic, to do so. There will be conflicts for available resources, which must be resolved, otherwise a resource-allocation analysis would not be needed. Resolving conflicts will often require extending the duration of at least some projects; the higher the priority of a project, the less desirable it is to extend its duration. One logical basis for expressing priorities is to determine the costs incurred per time unit of added project duration. The resulting figure is the same as that used in the time-cost trade-off procedure of Chapter 4 to determine when the most favourable schedule has been achieved. It is equal to the indirect costs for project performance during the time range following its earliest possible completion and also includes those costs representing lost income or extra expense as a result of not placing the completed project in operation. If such figures can be determined, they give an ideal quantitative basis for project priorities and can be used directly. A less precise alternative is to assign priorities arbitrarily on the basis of the judgement of the programme management.

The basic procedures of Chapter 5 can now be applied. Assuming that the resource restrictions are sufficiently severe to require the procedure for Case II, modifications of this procedure will be discussed in further detail. Resource-availability limits must be initially set and can be determined by one of the methods suggested in Chapter 5; then activities should be ordered in latest start sequence. Because of the large number of activities, there are advantages in listing the activity data on cards and using one card for each activity instead of listing the activity labels on the worksheet, as proposed in Chapter 5. Since it is generally not possible to assign one row on the grid of the worksheet to each activity, a band of rows should be assigned to each project. The magnetic bars that carry the activity resource data may also include the activity label; then non-concurrent activities in the same project can be placed in the same grid row. The band of vertical space required for each project need only be large enough to contain the maximum number of activities that can be performed concurrently. Hence, this is a space-saving measure.

Another reason for listing activities on cards is that the order of activities will change during the scheduling procedure. If the completion time of one project is extended, the latest start times for activities in that project are all increased by an amount equal to the increase in project duration (though the latest start times of activities in other projects are unaffected). Since latest start times are the basis for ordering the activities for scheduling, reordering is necessary. With the activities placed on cards this reordering is simple. The original latest start time of each activity is listed in the upper corner of its card, and the cards are arranged in order based on these figures; scheduling of activities commences with the activity on the first card. When a change in project duration becomes necessary in order to resolve resource conflicts, the magnitude of the change is listed in an assigned column of the worksheet in the horizontal band of that project in a manner similar to the listing of delays in figure 63. As each card is examined preparatory to scheduling its activity, the total delay of its project is noted from the worksheet. The latest start time on the card is updated if necessary, and the card is reordered according to its updated latest start time. If the card needs to be reordered it will not be scheduled at that time; the next card will be examined and the process continued.

When resource conflicts do occur during the scheduling process, their solution is complicated by project priorities. If there are no priorities and the only objective is to complete the over-all programme in the least possible time, then all projects can be combined into a single network. They can be tied together at the beginning by an event node representing programme commencement and at the end by an event node representing programme completion. The problem is reduced to one of single-project resource allocation (although the network may be quite large) with the exception that some attention may need to be given to interproject mobility in shifting resource units. In most cases priorities for completion of the individual projects within the programme do exist and need to be considered.

As each activity is scheduled, its magnetic strips are placed on the grid sheet. It is helpful to enter the activity label on the upper left corner of the

bar assembly and its latest finish time in the upper right corner. Latest finish time is preferable to latest start time for this purpose, since the activity may be interrupted during the scheduling process. This figure will show when the project is being extended in duration and, during the scheduling of other activities, will permit a rapid determination of the float time of any previously scheduled activity. The original latest finish time value should be used. It can be quickly updated by glancing at the column in which project delays are tabulated and adding the amount of any delay already incurred by its project. The float time of an activity can be determined easily by subtracting the date on which the activity is scheduled for completion (the time unit of the column in which the activity bar terminates) from its updated latest finish time. As activities are scheduled the initial attempt is made to start them immediately after the latest of their preceding activities have been completed. The identity of the preceding activities can be found by referring to the appropriate network diagram. It is also helpful to have each activity's preceding activities listed on its card. The bars of the preceding activities are observed on the worksheet to determine the earliest possible start date for the activity being scheduled.

Owing to resource limitations, it may not be possible to schedule an activity at the earliest date following completion of its preceding activities. In this circumstance either postponement of this activity, rescheduling of other activities, or some other measure is required. The entire group of concurrent activities requiring the limiting resource, including activities previously scheduled and the activity currently being scheduled, are considered together. The first objective is to try to confine any rescheduling to float ranges if possible. The second objective is to observe priorities if there are alternative solutions, rescheduling those activities with as low a priority as possible. Ideally, activities should be rescheduled in their free float ranges, but it is usually not practical to provide the data necessary to allow differentiation between free and interfering floats during the progress of these frequent investigations.

If the above rescheduling cannot be accomplished within float ranges, then activity rescheduling will cause project delays. The least unfavourable solution should be sought on project priorities and the amount of delays necessary. If equal amounts of delays in different projects are produced by alternative solutions, the solution delaying the project with the lowest priority should be selected. If unequal amounts of delay are involved, the delays should be weighted by use of the priority values in arriving at the solution. For example, suppose that Project *A* has a priority of 500 (based on a cost of $500 per day if it is delayed) and Project *B* has a priority of 300. Suppose further that in order to resolve a resource conflict Project *A* would have to be delayed one day or Project *B* could be delayed two days. In this case it would appear more desirable to extend the duration of the higher priority project, since 500×1 is less than 300×2. Once the least unfavourable solution involving rescheduling that causes project delays has been determined, alternatives involving other measures should be considered. These might, for example, require reprogramming of activities to change their resource requirements or they might be decisions to alter resource availability levels. The best solution that can be developed is finally applied, and then the

next activity in order is considered; the scheduling process continues in this manner.

The matter of interproject mobility of resource units should also be considered during the scheduling process, but formal rules are difficult to outline. If there are costs associated with interproject shifting of resources, then these costs should be determined in order to assist in scheduling decisions. In general, if resource units will be required again on a particular project, an alternative to shifting them to another project is to retain them where they are until they can be used again. This may cause delays in other projects. If so, it becomes a matter of whether the savings realized by not transferring the equipment to and from the other project is greater than the cost of the delays caused. A difficulty in reaching decisions on this matter is that it is not practical to look very far ahead, since the scheduling of future activities has not been determined at the time that a transfer of resource units is first considered. The most effective approach is probably that of viewing the problem in retrospect. If an activity requires resource units that were previously on that project but transferred to other projects, it may be possible to reconsider and revise the scheduling that has already been performed. As long as the time period to be reanalysed does not extend very far back, it is often possible to develop an alternative scheduling that will profitably eliminate certain resource transfers without resulting in excessive rescheduling effort.

As scheduling progresses and project delays become necessary, these delay amounts are listed on the worksheet as already described. In addition it is advisable to keep a record of the specific activities and resources that were responsible for each delay. This will permit a more intelligent analysis of the final results and furnish better insight in determining whether certain resource levels should be increased or whether certain portions of the programme should be reprogrammed.

DISCUSSION OF A SAMPLE PROBLEM

A two-day workshop on "Multi-project Scheduling for Highway Programmes" sponsored by the Automotive Safety Foundation was held in Washington, D.C., in December 1963. The programme was devoted primarily to a discussion of methods for multi-project resource allocation and, in particular, to two of the most sophisticated computer programmes available (both are proprietary programmes of private organizations) for this purpose. A sample problem was presented by a representative of the conference sponsor (Automotive Safety Foundation, 1963). It involved a programme consisting of twelve projects that could be performed concurrently, each of which used six different resources drawn from a common pool. The resources consisted of 32 men in professional and semi-professional classifications associated with the pre-construction engineering effort for the twelve projects. The classifications included planning, traffic engineering, surveying, and highway design, and the projects were typical highway projects. Each of the twelve projects had a different priority. There was complete interproject mobility of resources. The resource

restrictions were relatively tight, with limits of 2 units for two resources, 3 units for another, and 4 units for another. The result was that programme duration had to be extended considerably with relation to the duration of the longest project in the programme, which was 48 days. This problem was submitted for solution by each of the two computer programmes mentioned, and the results were compared and analysed. One of the two programmes was able to schedule the over-all programme in 96 days and did so without interrupting any activities. The other programme succeeded in scheduling the over-all programme in 88 days and interrupted 9 activities.

The techniques proposed in this chapter were tested on this same sample problem. On the one hand, the scheduling by the manual procedure required a full day's work. Probably the computer produced its solution in a matter of minutes once the data were prepared and submitted to it. On the other hand, the results of the manual procedure were better, and the effort appeared well worth while. The programme was scheduled in 88 days without an interruption in any activities; thus, the best results of the two computer programmes were combined. Of greater importance, the priority requirements of the twelve projects were much more fully satisfied, and the three top priority projects were not delayed at all. In one computer programme the top priority project was delayed seven days; and the second priority project was delayed seven and nine days, respectively, by the two computer programmes. In five of the twelve projects the manual method met priority requirements better than either programme; in two others it was better than one and as good as the other; in one it was the same as both; and in no case did it do as poorly as one of the two programmes. The most important advantage of the manual solution was that the person who performed the work gained a clear understanding of the problem and the reason for the project delays. He probably could make much more intelligent suggestions for improving programme scheduling with respect to resource requirements than the person who could only study the final output of the computer programmes. In practice, some of these proposed changes would probably have been made in the early stages of schedule development. Actually the manual procedure in this case was applied with unrealistic handicaps in order to compare its results with the other solutions. The only changes permitted were rescheduling ones. Had a person with a knowledge of the work to be performed been allowed to reprogramme certain activities or consider any of the many other measures available besides simply rescheduling activities, it is quite certain that the schedule could have been improved much further.

A fair question is whether a day's effort or several days' efforts are justified to develop a schedule for any given programme. If computer capability does not exist, the alternative appears to be the omission of advance programming and the facing of problems as they arise. If this is the choice and if the programme is one of importance, the benefits obtained by advance programming ought to justify the effort required. If computer capability does exist, it is still probable that a manual analysis will be worth while for a programme of importance. Both methods of attack could be used and, indeed, would prove useful in supplementing one another.

A programme involving industrial development is often of importance to the national economy, will involve considerable expense and effort over a long period ot time, and will require the commitment of scarce resources that could be used for other worth-while purposes. Such programmes justify hard work in programming both prior to their commencement and throughout their implementation. If any method has a good probability of producing good results even though it requires the expenditure of much time by a key member of the programme management, the method should be applied. A scheduling procedure permitting the maximum degree of judgement control while utilizing network information and mechanics would appear to be such a method.

Chapter 7

COMBINING TIME-COST TRADE-OFFS AND RESOURCE ALLOCATION

Feasibility of independent applications

Procedures for the application of time-cost trade-offs have been presented in Chapter 4 and for resource allocation in Chapters 5 and 6. These topics have been treated independently. This is the general approach followed in other treatments of these topics, and it implies that these are independent techniques and can be applied separately. Is this true in a practical sense?

It would seem more feasible to apply resource-allocation analysis independently than to apply time-cost trade-offs separately. Actually, most resource-allocation applications involve certain time-cost trade-off decisions, at least in the broad sense. Resource-availability limits must be established. Unless these limits are determined by absolute maximums, which is almost impossible, they involve a type of time-cost trade-off. The decision to use a particular limit is a compromise between the cost of providing additional resource units and the degree of restriction that is placed on the scheduling of activities requiring that resource. The latter is almost invariably reflected in project time requirements. Following establishment of resource limits, activities are scheduled to meet them. As long as this scheduling is confined to float ranges, project duration is not affected, but most of the other alternatives involve some form of time-cost trade-off.

If project duration is extended because of scheduling of activities beyond their float ranges, this is a time-cost decision. Project duration has been increased to avoid the expense of increasing certain resource levels. Or conversely, if resource levels are increased during resource-allocation procedures, this is a decision that involves expense but either shortens the project duration or prevents it from being increased. If activities are replanned to change their resource requirements and duration, a cost is usually involved. The purpose of this is generally to influence project duration, so a time-cost trade-off is involved. In Chapter 4, time-cost trade-offs involved changes in the method of performace of activities that affected their time and cost requirements and were applied in a rather formal manner to produce changes in project duration. It is fair to state that resource-allocation procedures can be applied independently of time-cost trade-offs of this nature. If a broader definition is applied to time-cost trade-

offs to include any measures that balance costs against changes in project duration, such as changes in resource-availability levels or in resource requirements of activities, then resource-allocation procedures would be difficult to apply independently of time-cost trade-offs.

Can time-cost trade-off methods be applied without resource-allocation analysis? This can be done, but in much the same way that an ostrich, according to popular belief, hides its head in the sand to avoid danger. A programmer can allow himself to be blind to existing conditions and take actions that seem to offer a good solution but actually are very ineffective. Unless resource availabilities are higher than any maximum requirements that might occur and there are no expenses related to the variations in resource requirements with respect to time, time-cost trade-offs cannot be realistically accomplished without consideration of resource requirements. Any time-cost trade-off must of necessity affect the duration of one or more activities. Whether resource requirements of those activities are changed or not, the duration change will produce changes in resource schedules. Practically any change in resource schedules must affect costs. Changes in costs should be considered in establishing the cost slope on which the time-cost trade-off is based. If they are not considered, the basic data will be invalid and may lead to faulty conclusions. Some changes that would be made may actually be impossible for lack of available resources. Hence, time-cost trade-offs should not be applied independently of resource-allocation analysis.

JOINT APPLICATION

Regardless of whether resource-allocation and time-cost trade-off procedures can or cannot be applied independently, the use of both is desirable to produce a good project implementation plan and schedule. For example, a knowledge of resource schedules is not only advisable in determining valid time-cost trade-offs, but it opens up a whole new area of possibilities for additional favourable trade-offs. The knowledge of the existence of idle resources at specific times permits changes to be proposed that utilize these idle resources. Such changes may be possible to accomplish at zero or very low cost slopes. If a piece of equipment is idle during a certain time period, it may be possible to use it in the performance of activities scheduled during this time period to reduce their costs even though such use could not be justified if the equipment were not idle. Or, if men are idle during a given time period, it is often possible to employ them on activities in progress even though the resulting crew sizes are larger than the ideal sizes for maximum efficiency. In both cases performance times can be reduced at little or no extra cost.

Resource-allocation procedures may require postponing critical activities or scheduling non-critical activities beyond their float ranges, particularly if resource levels are difficult to satisfy. In either case, gaps in the critical path will be created. If any critical activity (i.e., any activity that has zero total float) is not scheduled at its earliest start time, its preceding activities will have float

time and therefore cease to be critical. Then there will be no single critical path across the entire project network. After extensive resource-allocation scheduling there may be very few critical activities remaining, in the sense in which critical activities have been defined so far. This has a rather drastic effect on the application of the time-cost trade-off procedures outlined in Chapter 4, since those procedures require the use of critical activities. Unless the definition of critical activities is broadened, there may be little opportunity for the application of time-cost trade-offs because of a lack of critical activities. If a critical activity is considered to be any activity whose performance has a direct effect on project completion, new opportunities for time-cost trade-offs are presented. Those activities whose resource requirements cause the scheduling that produces the gaps in the critical path can also be considered critical. Methods for changing their performance and resource requirements become effective measures for changing project duration and thus many new time-cost trade-off possibilities are introduced. Or, as pointed out earlier, the whole question of revising resource limits can be treated as a time-cost trade-off problem.

PROCEDURE FOR COMBINED APPLICATIONS

A joint application of resource allocation and time-cost trade-offs has been shown to be desirable. This joint application will require modifications of the procedures already developed for independent applications. Therefore, it is appropriate to conclude this publication with the following proposed procedure for combining these techniques:

(1) Develop a project implementation plan and reduce it to a network diagram. Attempt to minimize the amount of activity breakdown and detail while arriving at a reasonably realistic representation of the project implementation plan. Base network relationships primarily on physical sequencing requirements and avoid sequencing based on resource restrictions unless the programmer has no doubt concerning their necessity.

(2) Obtain time estimates for each activity in the network. It is desirable that these estimates be based on those methods of performance that involve the least direct costs. This will allow an orderly development of the time-cost curve from a starting point at the "normal" solution. (It is not essential that this starting point be used. A "conventional estimate" starting point has also been proposed (Fondahl, 1962) when actual performance is known to be required in considerably less time than "normal" performance.)

(3) Perform the basic scheduling calculations to provide earliest and latest start and finish dates and a knowledge of critical activities.

(4) Establish a cost figure that represents the value per time unit of any reductions in project duration. This will serve as a standard in determining whether proposed time-cost trade-offs should be made. Then perform any time-cost trade-offs that can be accomplished at zero cost or at low cost slopes relative to the limiting figure just established. After critical activities have been identified, it is often possible to expedite them by more detailed planning or

to break them down further to allow additional overlapping in their performance, all with little or no cost increase. It is not necessary to develop either normal cost data or crash time and cost data for any of the activities. Decisions can be made solely on the basis of changes in time and cost of the critical activity being considered and the resulting change in project duration. It is true that these time-cost trade-offs will be made blindly with respect to the effects on resource schedules that have not yet been developed. Once resource schedules have been developed, time-cost trade-offs involving analysis of the entire network will require excessive data-handling to revise those schedules. If time-cost trade-offs are made at this stage of the procedure and are limited to those involving low cost slopes, they will probably still be justified after the additional opportunities furnished by a knowledge of resource schedules are known. Moreover, those changes that will prove ineffective owing to unfavourable resource demands can be reversed later, and generally the cost can be recovered by the "sell-back" of the time.

(5) Develop resource schedules for selected resources that are known to be scarce or expensive or which seem to be needed in such quantity in relation to their availability levels that they are likely to create problems.

(6) Examine the resulting resource schedules and the project duration for over-all reasonableness. It is to be expected that resource schedules can be improved by measures to be taken and, if resource restrictions are not unduly severe, that project duration can be decreased by additional expenditures. If the existing results are so poor as to indicate that the basic implementation plan is probably not workable and that major reprogramming is advisable, it is desirable to accomplish this before further detailed analysis is undertaken.

(7) The procedure from this point will depend on the severity of resource restrictions and the complexity and duration of the project. Assume a difficult situation involving low resource availability limits and a complex project extending over a considerable period of time.

(8) Working with the entire network, attempt to develop solutions for eliminating resource peak requirements that are well above availability limits. Owing to the amount of data involved, such solutions will be largely limited to rescheduling within free float ranges or in interfering float ranges where chains of affected activities are short, or to other measures that have limited chain effects. Further time-cost trade-off analysis is generally not practical owing to the amount of data that would have to be updated.

(9) Establish the resource-availability limits for further detailed levelling.

(10) Develop a subnetwork for the initial portion of the project using the dateline cut-off method. Use a time period for the subnetwork that produces a small enough network (after some further expansion if a more detailed breakdown is advantageous and after the addition of any other resources that deserve consideration also) to allow practical application of time-cost trade-off and resource-allocation techniques. Do not use a longer time period ahead than that which warrants detailed programming.

(11) Apply the procedures for Case II discussed in Chapter 5 to develop a schedule that meets resource-availability limits. Whenever conflicts arise that

would require project extension, consider the application of time-cost trade-off measures, such as replanning activities to reduce resource requirements, expediting activities to avoid concurrent performance that produces resource conflicts, or changing the resource limits.

(12) Having developed a schedule for the subnetwork activities that meets resource restrictions, attempt to develop further time-cost trade-offs that appear to shorten project duration. Whether they actually do so will depend on the ability to solve resource problems in subsequent subnetworks. It is possible that some expenditure will be made uselessly; but they will produce opportunities for project-expediting, and most of these opportunities will probably lead to successful results. In making time-cost trade-offs, take advantage of the periods when idle resources are available. This will offer additional time-cost trade-off opportunities and will provide chances to level resource use.

(13) Investigate any other changes that will help to level resource requirements. Since variations in these requirements, even though they are below availability limits, generally represent expenses, smoothing these curves is an effective cost-reduction measure.

(14) As work progresses during the subnetwork period, keep the time schedule and the resource schedules updated and continue to work on them. Reprogramme the work whenever changes make it necessary to do so, when new opportunities develop or increased knowledge of project conditions is gained, or whenever time can be devoted to further detailed analysis. Begin development of the following subnetwork well before the work covered by the current subnetwork is completed.

(15) Whenever there is reason to believe that changes in the subnetwork may have produced major effects on the over-all time or resource schedules, another general analysis of the remaining over-all plan, as in steps 3 to 9, should be performed.

It should be emphasized that the techniques proposed in this publication have been offered as reasonably simple, non-computer procedures that can be used at the project level. Problems that are considered complex and difficult to solve even with large capacity computers have been the primary subject matter of this publication. The methods developed are not sophisticated mathematically, and it is not claimed that they will produce optimum solutions. They depend to a considerable extent upon the application of the knowledge and judgement of the user. It is believed that the ability to apply this judgement throughout the application of the procedures will often provide better solutions than more formal, mathematically oriented techniques. Finally, these procedures are not intended for use only during the original programming stage of the project. The principal reason for attempting to make them project-level tools, and for believing that they can be truly effective, is that they are expected to be used for continual updating and reanalysis throughout the performance of the project; they are truly dynamic tools and must be used in this manner.

BIBLIOGRAPHY

ALPERT, L. and D. S. ORKAND (1962) *A time-resource trade-off model for aiding management decisions*, Operations Research Inc. (Technical Paper No. 12)

ANON. (1961a) "Critical-path scheduling", *Plant Administration and Engineering* (October).

ANON. (1961b) "Teaching PERT project network techniques", *Training Directors* (December).

ANON. (1962a) "CPM moves into the specifications", *Engineering News-Record* (December 6).

ANON. (1962b) "Critical-path scheduling", *Chemical Engineering* (April 16).

ANON. (1962c) "Planning, implementation and appraisal through PERT", *Business Budgeting* (January).

ANON. (1962d) "Time-scale simplified CPS diagrams", *Plant Administration and Engineering* (March).

ANON. (1965) "Contractors shift from arrow to precedence diagrams for CPM", *Engineering News Record* (May 6), 32–33.

ARCHIBALD, R. D. (1962) *Experience using the PERT cost system as a management tool*, Institute of Aerospace Sciences, Los Angeles (June 21).

AUTOMOTIVE SAFETY FOUNDATION (1963) *Multi-project scheduling for highway program; proceedings of a two-day workshop*, Washington, D.C.

BAKER, B. N. and R. L. ERIS (1964) *An introduction to PERT/CPM*, Irwin, Homewood, Ill.

BATTERSBY, A. (1964) *Network analysis for planning and scheduling*, St. Martin's Press, New York.

BEUTEL, M. L. (1963) "Computer estimates costs, saves time, money", *Engineering News-Record* (February 28).

BIGELOW, C. G. (1962) "Bibliography on project planning and control by network analysis: 1959–1961", *Operations Research* 10 (5), 93–104.

BOEHM, G. A. W. (1962) "Helping the executive to make up his mind; (decision theorists come to rescue with CPM and PERT linear programming etc.)", *Fortune* 65 (April), 128–131.

BOOSE, J. A. (1963) *A non-intuitive decision-making method for configuration of a complex system*, International Business Machines Corporation, Space Guidance Center, Oswego, N.Y.

CHARNES, A. and W. W. COOPER (1962) "A network interpretation and a directed subdual algorithm for critical path scheduling", *Journal of Industrial Engineering* 13 (4), 213–218.

CLARK, C. (1960) *A discussion of an action planning and control technique*, International Minerals and Chemical Corporation, Freeport, Texas.

CLARK, C. (1961) "The optimum allocation of resources among the activities of a network", *Journal of Industrial Engineering* 12 (1).

ECKMAN, D. P. (1961) *Systems research and design*, Wiley, New York.

FAZAR, W. (1962) Planning implementation and appraisal through 'PERT', *Business Budgeting* (January).

FONDAHL, J. W. (1962) *A noncomputer approach to the critical path method for the construction industry*, 2nd ed., Department of Civil Engineering, Stanford University, Stanford, Calif.

FONDAHL, J. W. (1964) *Methods for extending the range of noncomputer critical applications*, Department of Civil Engineering, Stanford University, Stanford, Calif. (Technical Report No. 47).

FULKERSON, D. R. (1961) "A network flow computation for project cost curves", *Management Science* 7 (2), 167–179.

HALL, A. D. (1962) *A methodology for systems engineering*, Van Nostrand, Princeton, N.J.

HAMDY, M. H. A. (1967) *Application of network procedures in the implementation of industrial projects in developing countries*, paper presented at the International Congress for Project Planning by Network Analysis, Vienna, Austria, 21–27 May 1967.

HAMDY, M. H. A. (1969) *Problems encountered in the application of network analysis techniques in project implementation in developing countries and pertinent recommendations*, paper presented at the International Congress for Project Planning by Network Analysis, Amsterdam, the Netherlands, 6–10 October 1969, published in *Project Planning by Network Analysis* (Proceedings of the Second International Congress, Amsterdam, the Netherlands, 6–10 October 1969), pages 54–70.

KELLEY, J. E. and M. R. WALKER (1959) "Critical path planning and scheduling", in *Proceedings of the Eastern Joint Computer Conference, Boston, 1–3 December*, pp. 160–173.

KELLEY, J. E., Jr. (1961) "Critical path planning and scheduling: mathematical basis", *Operations Research* 9 (3), 296–320.

LEVIN, R. I. and C. A. KILPATRICK (1966) *Planning and control with PERT/CPM*, McGraw-Hill, New York.

MARSHALL, A. W. and W. H. MECKLING (1959) *Predictability of the costs, time, and success of development*, RAND Corporation (Report P-1821).

MILLER, R. (1962) "How to plan and control with PERT", *Harvard Business Review* (March–April), 93–104.

MILLER, R. W. (1963) *Schedule, cost and profit control with PERT; a comprehensive guide for program management*, McGraw-Hill, New York.

MODER, J. J. (1963) "How to do CPM scheduling without a computer", *Engineering News-Record* (March 14), 30–34.

MODER, J. J. and C. R. PHILLIPS (1966) *Project management with CPM and PERT* 3rd ed., Reinhold, New York.

MOSHMAN, J. J. JOHNSON and M. LARSEN (no date) "RAMPS: a technique for resource allocation and multiproject scheduling", in *Proceedings 1963 Spring Joint Computer Conference*, Books Incorporated, New York.

MUNDORFF, C. T. and W. BLOOM (1960) *Managing a development programme*, Bureau of Research and Development, FAA, General Precision Inc., New York.

MUTH, J. F. and G. L. THOMPSON (1963) *Industrial scheduling*, Prentice-Hall, Englewood Cliffs, N.J.

NORDEN, P. V. and F. J. O'REILLY (1960) *Life cycle method of project planning and control*, International Business Machines Corporation, Data Systems Division, Poughkeepsie, N.Y.

PEARLMAN, J. (1960) "Engineering program planning and control through the use of PERT", *IRE Transactions on Engineering Management* 7, 125–134.

SANDO, F. A. (1964) "CPM, what factors determine its success?", *Architectural Record* 155 (April/May) 202–204..., 211–216.

SHAFFER, L. R., J. B. RITTER and W. L. MEYER (1965) *The critical-path method*, McGraw-Hill, New York.

SIMMS, T. J. G. (1961) "The critical-path method; a new approach to planning", *Engineering and Contract Record* (June).

STEINFIELD, R. C. (1960) "Critical-path saves time and money", *Chemical Engineering* (November 28), 120–152.

VAN KRUGEL, E. (1964) "Introduction to CPM", *Architectural Record* 136 (September), 337.

VAN SLYKE, R. M. (1963) "Monte Carlo methods and the PERT problem", *Operations Research* 11 (September), 838–860.

VILLERS, R. (1959) "The scheduling of engineering research", *Journal of Industrial Engineering* (November/December).

WALDRON A. J. (1963) *Fundamentals of project planning and control*, 2nd ed., Waldron, West Haddonfield, N.J.

WALKER, M. R. and J. S. SAYER (1959) *Project planning and scheduling*, Du Pont de Nemours, Wilmington, Delaware (Report No. 625 a).

WHITE, G. L. (1963) "An introduction to computerized CPM", *The Constructor* (April) 53–55.

WIEST J. D. (1964) "Some properties of schedules for large projects with limited resources; critical-path method, PERT and related techniques", *Operations Research* 12 (May), 395–418.

Printed in Austria

69-383—June 1971—4,300

Price: $U.S. 2.00
(or equivalent in other currencies)

United Nations publicatio

Sales No.: E. 70. II. B. 1

ID/SER. L


2.8.73