T-3119

DYNAMIC NETWORK-BASED PLANNING TO DEFINE OPTIMAL PREPRODUCTION TIME COMPLETION

by

Jose M. Andujar

ProQuest LLC.
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106 – 1346

T-3119

A thesis submitted to the Faculty and the Board of Trustees of the Colorado School of Mines in partial fulfillment of the requirements for the degree of Master of Science in Mining Engineering.

Golden, Colorado
Date 8/11/86

Signed: _____
Jose M. Andujar

Approved: _____
Dr. Robert Cameron
Thesis Advisor

Golden, Colorado
Date: 8/13/1986

_____
Dr. Miklos Salamon, Head
Mining Engineering

ABSTRACT

This thesis shows how to define the optimal time for completion of the preproduction period of a new mining project utilizing a dynamic network-based planning approach. It allows three optimal preproduction schedules depending on the mine planner's objectives: such as, highest net present value of the entire mining venture, fixed preproduction cost budget, or fixed preproduction time completion.

The dynamic network-based approach utilizes the Out-of-Kilter algorithm (OKA) for the solution of a Critical Path Method (CPM) Cost Model. The OKA algorithm defines the optimum minimum time-cost trade-off function of the preproduction period.

A sensitivity analysis is used to determine the influence of different preproduction schedules on the mine planner's objectives, and defines the optimal schedule. The algorithm solution is applied to a very simple new mining project to demonstrate its use.

A computer program has been implemented using FORTRAN 77 on a Personal Computers (PC's). The program is limited to work for small-scale mining projects. The computer code is contained in the thesis.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I am thankful and grateful to several people which made this work possible.

Arthur and Evelyn Beynon, close friends of my family, first opened up the possibility of attending Colorado School of Mines and helped me find the support to carry through this intention. Roger Allen, who was the Financial Manager of Cia. Minera Del Madrigal, Peru, provided the additional support and encouragement needed.

Dr. Thys B. Johnson, former Head of the Mining Department and my first thesis advisor, introduced me to the topic of this thesis and gave both time and attention to my work. Prof. Matthew Hrebar provided me with understanding, support and friendship. I owe a debt of gratitude to Dr. Robert Cameron, my final thesis advisor for his services and help.

Dr. Robert King, of the Mining Department, and the Colorado School of Mines, supplied financial support which enabled me to finish my degree. I want to thank my final committe members, Dr. Miklos Salamon and Dr. Fun Den Wang, and all the members of the Mining Department for their aid and support in my graduate studies.

Finally, I would like to dedicate this thesis to my

parents, Jose and Maria Andujar, for everything that they have done for me, spiritually, emotionally, and financially; I owe them a greater debt than I can express. My wife, Maria Luz, I want to thank for the patience and understanding she gave during the time I have devoted to my studies.

CHAPTER 1

INTRODUCTION

The emphasis in most mining feasibility optimization studies has concentrated on the production phase of activity. This thesis is concerned with the activities that precede production, commonly defined as the "preproduction" period. The preproduction period is considered to end as soon as production begins.

Charles River Associates Inc. (1979) states: "Most owners do not realistically estimate the time required to bring a new mining project into production." If the amount of time to complete the preproduction period is not properly taken into account in project's financial feasibility analysis, positive cash flows are unlikely to be achieved on schedule, and a different investment decision can be reached.

The preproduction period consists of a range of activities, such as, the initial surface surveys, shaft sinking, and underground development work. The successful completion of all the activities involved in the preproduction period determines the beginning of the production period and hence, the positive cash flow of a project. The timing of the positive cash flow or the end of

the preproduction period, plays a major role in most valuation decision criteria.

The preproduction schedule can usually be shortened or extended, depending on the intensity of resource outlay a mining company is willing to expend. The preproduction or development period can therefore be viewed as a project whose time duration can be the target of optimization. In doing so, it is assumed that each activity within the preproduction period can be accomplished on different schedules depending on the amount of resources such as capital, manpower, and equipment assigned to each job. A range of time completion of each activity can then be defined. The range of time durations are generally defined by the two extremes and labeled as the normal and crash time durations.

The Critical Path Method (CPM) is used as a management aid to define the time completion of the whole preproduction period. CPM helps managers to define and monitor those activities critical to timely completion of a project and to define the activities whose start up and completion will not delay the project. A special form of CPM, Critical Path Method Cost Model formulates a Linear Program (LP) model to optimize the project cost by shortening its duration through accelerating the completion of critical activities. The Out-

of-Kilter Algorithm (OKA) can be used to solved the CPM Cost model by finding the optimum least time-cost trade-off function. The time completion of every activity within the preproduction period is scheduled by this technique. Then, any valuation criteria can be used for the final go/no-go investment decision.

The Net Present Value (NPV) has been chosen as the valuation criteria in this study. The present value of costs associated with each preproduction time completion and the present value of revenues, which vary depending on the preproduction time completion, are calculated. The NPV is then the difference between the two. Once the preproduction time completion has been properly factored into the project's financial feasibility analysis, the investors will be able to consider the effect of any change in the schedule of completion of this period.

This thesis demonstrates how to find the optimal time for completion of the preproduction period of a new mining project by establishing a dynamic network-based planning approach utilizing the Out-of-Kilter Algorithm. The optimal preproduction schedule may also be constrained by: (1) fixed preproduction cost budget or (2) fixed preproduction time completion. A sensitivity analysis can then be used to

measure the influence of these constraints and to define which schedule is optimal.

This thesis is divided into 7 chapters and 2 appendices. The following chapter, chapter 2 gives a literature review of the topic treated in this thesis. The development and formulation of the preproduction network is presented in Chapter 3 including network notation and terminology.

Chapter 4 develops the Out of Kilter mathematical formulation for the solution of the CPM Cost Model. The basic components of the solution methodology and a numeric example are contained in this chapter. Chapter 5 develops the solution algorithm for the optimal preproduction schedule.

A simplified analysis is developed in Chapter 6 on a small but representative hypothetical mine development project in order to demonstrate the solution technique. A computer program named LCS was written in FORTRAN 77 to find the optimal least time-cost trade-off. This program is limited to work for small-scale new mining projects. Appropriate details of the prototype computer implementation and execution information for the validation study are included.

Chapter 7 gives some conclusions and recommendations for further research. Appendix A gives an example of the hand computational procedure of the Network Flow Algorithm. Appendix B contains the LCS program code.

CHAPTER 2

LITERATURE REVIEW

The preproduction period is commonly discussed in mine valuation literature in a very general fashion. Detailed analysis of the activities within the preproduction period is seldomly viewed in sufficient detail as to determine its effects to the overall profitability of a proposed mining venture. Mine project optimization studies conducted by Wells (1978), Dowd (1976), and Gardner (1986), are typical of the literature in that they are generally confined to evaluating the effect of overall project parameters such as mine life, cut off grades, mine size and financing schemes.

Mathias and Redmon (1965) develop an optimal time-cost function using a case study for a potential mining property. Their work utilizes a computer program requiring 33 runs to obtain the full schedule compression and to develop the time-cost function. This algorithm is very computational inefficient and does not give detailed information which can be used for project management.

Gentry and O'Neil (1984) developed a feasibility study on a potential mining property. Details of the preproduction period such as equipment selection and manpower requirements, capital cost estimates, mine plans, and

project timing are considered but not given. Their analysis uses a simplified project activity schedule utilizing bar (Gantt) charts.

A study to compare the Gantt charts (developed by Gantt during World War I) and network diagrams was completed by Moder, Phillips, and Davis (1983). The bar chart lists the major activities comprising a project, their scheduled start and finish times, and their current status. This work demonstrates the major disadvantage of Gantt charts over network diagrams: Gantt charts do not explicitly show the dependency relationships among the activities.

Naftal (1964) introduces the Cerro Corp.'s Critical Path Calendar (CPC) which resembles bar charts. Its construction results in a visual model of the sequencing of individual jobs necessary for the completion of a mine or mill project enabling managers to better direct preproduction development. Network diagrams are a must, however, for complex projects.

Mike Nilsen (1981) measures the sensitivity of several general parameters on the project's financial feasibility. However, he does not consider the preproduction period.

Howard Wells (1978) develops a procedure to find a practical optimum size of mine for a particular deposit. He balances the positive cash flow and the negative cash flow

to optimize the return on investment capital. Wells' paper does not get into the concept of shortening the preproduction time completion.

Gentry and O'Neil (1984) looking at the investment, or negative cash flows, occurring during the preproduction period suggest it can be appropriately treated using the wealth growth rate (WGR) and the growth rate of return (GRR) criteria. However, in this thesis the simpler criteria, the net present value (NPV), is used for the accept/reject investment criteria. They too do not try to optimize this period.

Daellenbach and George (1978) are one of many general operations research texts that present the Critical Path Method Cost Model formulation which can be used to solve the minimum time-cost problem. Fulkerson (1961) shows that the Out of Kilter algorithm (OKA) generates the time-cost trade-off curve utilizing a network flow formulation of the CPM problem. Implementation of the OKA for solution of the CPM Cost Model in network form results in an answer containing all the advantages indicated by Naftal (1964).

Other systematic methods of generating optimal project time-cost functions have been carried out, such as the original CPM approach by Kelley (1961) and the heuristic

hand computational procedure by Siemens (1971). However, the Out of Kilter technique proved to be useful in easily generating the preproduction time-cost function.

CHAPTER 3

NETWORKING THE PREPRODUCTION PERIOD

This chapter is intended to clarify certain terms and concepts used in later chapters. It is very important to follow the Networking and the Critical Path Method algorithms in order to understand properly the solutions given as computer program output.

The Critical Path Method is a graphical process which management can use as an aid in planning and scheduling operations in a project. The accuracy and usefulness of a network is dependent mainly upon intimate knowledge of the project itself, and upon the general qualities of judgment and skill of the planning personnel (Moder, Phillips, and Davis, 1983).

There exist applications of this technique to projects in the mining industry, such as Mathias and Redmon (1965), Naftal (1964), and Hoffman (1964).

3.1 Development of the Network

The Critical Path Method divides the management function into two distinct phases: planning and scheduling. Planning is deciding what operations should be done, while scheduling is determining when.

The planning phase is initiated by making a detailed

listing of all the proposed jobs of the project. They are then represented graphically in a flow chart or network showing the precise sequence in which they must be executed. The planning phase is also the most time-consuming and difficult part of most critical path method applications, due primarily to the inherent analytical problems in any project planning effort.

The scheduling phase involves the identification of what operations actually control significant completion dates. It gives principal attention to these controlling operations.

3.1.1 Basic Terms

Several of the most common terms in networking are defined below. Terms associated with scheduling computations are explained in Section 3.2.

Activity--a well-defined amount of work to be accomplished which consumes time or resources and has a definable beginning and ending. Activities may involve labor, paper work, contractual negotiations, and machinery operations. Commonly used terms synonymous with "activity" are "task" and "job." In the arrow scheme of networking, activities are graphically represented by arrows, usually with descriptions and time estimates written along the arrow (Figure 3.1).

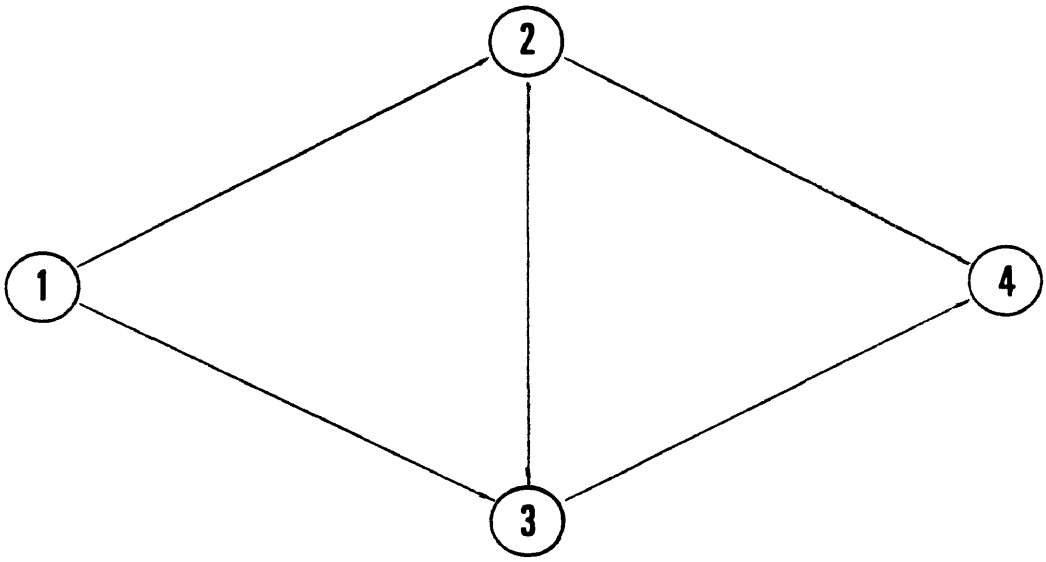


Figure 3.1 Example Network

Event--the beginning and ending points of activities. Theoretically, an event is an specific point in time. An event can represent the completion of either one or more than one activity, or the joint initiation of one or more than one activity. An event is often represented graphically by a numbered circle, called a node.

Network--a graphical representation of a project plan, showing the interrelationships of the various activities. When the results of time estimates and computations have been added to a network, it may be used as a project schedule.

Dummy Activity--an arrow representing merely a dependency of one activity on another. A dummy requires no time. The same arrow diagramming rules apply to dummy jobs just as they do real jobs, except that dummy jobs are usually represented by dashed-line arrows.

3.1.2 Arrow Diagramming--Basic Rules

An arrow network expresses relationships among operations. Every activity in a project is related to every other activity in one or more of the following ways:

1. What task(s) must be completed before this task can be started?
2. What task(s) cannot be started until completion of this task?

3. What task(s) can be done concurrently with this task?

The basic rules of network logic are:

Rule 1. Before an activity may begin, all activities preceding it must be completed.

Rule 2. Arrows imply logical precedence only. Neither the length of the arrow nor its orientation on the drawing have any significance.

Rule 3. Event numbers must not be duplicated in a network.

Rule 4. Any two events may be connected directly by no more than one activity.

Rule 5. Networks may have only one initial event (with no predecessor) and only one terminal event (with no successor).

For purposes of convenience in calculation, the node number at the tail of the arrow should have a lower value than that at the head of the arrow. These numbers do not necessarily have to be consecutive. When two or more jobs have common beginning and ending nodes, dummy jobs are inserted. This procedure insures that each job will have unique beginning and ending nodes. These requirements were applied to the program developed in Appendix B.

3.2 Basic Scheduling Computations

At this stage in the application of the critical path method, the project network plan has been completed and the performance times have been estimated for each activity as shown in Table 3.1 and Figure 3.1. The questions of how long the project is expected to take and when each activity may be scheduled are now considered. Answers to these questions are inferred from the network logic and the estimated durations of the individual activities.

All basic scheduling computations involve first a forward and then a backward pass through the network. Based on a specified project start time, the forward pass computations proceed sequentially from the beginning to the end of the project, giving the earliest start and finish times for each activity and the earliest occurrence time for each event.

From the specification of the latest allowable occurrence time for the completion of the project, the backward pass computations also proceed sequentially, but from the end to the beginning of the project. They give the latest allowable start and finish times for each activity and the latest allowable occurrence time for each event.

The path float is the amount of time by which the actual completion time of an activity on the path in

Table 3.1 Network Data

Activity	Predecessor Event i	Successor Event j	Duration D
1	1	2	3
2	1	3	4
3	2	3	2
4	2	4	5
5	3	4	6

question can exceed its earliest completion time without affecting the earliest start or occurrence time of any activity or event on the network critical path.

After the forward and backward pass computations are completed, the path float can be computed for each activity, and the critical path through the network determined (Section 3.2.4). The network computation procedures do not change if the unit of time changes.

3.2.1 Computation Nomenclature--Arrow Diagrams

The following nomenclature will be used in the formulas and discussion which describe the various scheduling computations. These definitions and subsequent formulas will be given in terms of an arbitrary activity designated as (i,j), i.e., an activity with predecessor event i and successor event j.

D(i,j) = estimated duration time for activity (i,j).

TE(i) = earliest occurrence time for event i.

TF(i) = latest allowable occurrence time for event i.

ES(i,j) = earliest start time for activity (i,j).

EF(i,j) = earliest finish time for activity (i,j).

LS(i,j) = latest allowable start time for activity (i,j).

LF(i,j) = latest allowable finish time for activity (i,j).

T = scheduled time for the completion of a project

3.2.2 Forward Pass Computations

To compute the early start and finish times for each activity and the earliest possible occurrence times for each event in the project, the forward pass computations are initiated by assigning an arbitrary earliest start time to the initial project event. The rules are summarized below.

Rule 1. The initial project event is assumed to occur at time zero. Letting the initial event be denoted by 1, this can be written as:

$$TE(1) = 0$$

Rule 2. All activities are assumed to start as soon as possible; that is, as soon as all of their predecessor activities are completed. For an arbitrary activity (i,j), this can be written as:

$ES(i,j)$ = Maximum EF value for nodes preceding activity (i,j).

$TE(j)$ = Maximum TE for events preceding event j plus the time duration of activities ending at node j.

Rule 3. The early finish time of an activity is merely the sum of its early start time and the estimated activity duration. For an arbitrary activity (i,j) this can be written as:

$$EF(i,j) = ES(i,j) + D(i,j)$$

$$TE(j) = \text{Max } \{TE(i) + D(i,j)\} \qquad [i<j]$$

The above rules are applied to the simple network shown in Figure 3.1. In the forward pass section, the initial project event 1 is placed at 0 on the time scale according to the first rule. Starting with TE(1) = 0, the early start time for activities (1,2) and (1,3) is 0, and the early finish time of activity (1,2) by Rule 3 is merely

$$EF(1,2) = ES(1,2) + D(1,2) = 0 + 3 = 3$$
$$TE(2) = TE(1) + D(1,2) = 0 + 3 = 3$$

Since an event j cannot occur until all activities leading into node j have been completed, a crux of the forward pass computations occurs at the merge event 3, where it is necessary to consider the early finish times for predecessor activities (1,3) and (2,3) to determine the early start time for activity (3,4), i.e.,

$$EF(1,3) = ES(1,3) + D(1,3) = 0 + 4 = 4$$
$$ES(2,3) = EF(1,2) = 3$$
$$EF(2,3) = ES(2,3) + D(2,3) = 3 + 2 = 5$$

$$ES(3,4) = \text{Maximum of} \begin{matrix} EF(1,3) = 4 \\ EF(2,3) = 5 \end{matrix} = 5$$

or

$$TE(3) = \text{Maximum of} \begin{matrix} TE(1) + D(1,3) = 0 + 4 = 4 \\ TE(2) + D(2,3) = 3 + 2 = 5 \end{matrix} = 5$$

Finally, the early possible occurrence time for the final event 4 is

$$TE(4) = \text{Maximum of} \begin{matrix} TE(2) + D(2,4) = 3 + 5 = 8 \\ TE(3) + D(3,4) = 5 + 6 = 11 \end{matrix} = 11$$

Thus, the early expected finish time for the entire project, corresponding to the earliest occurrence time of the project terminal event 4, is denoted by TE(4) = 11.

3.2.3 Backward Pass Computations

The purpose of the backward pass is to compute the latest allowable start and finish times for each activity and the latest allowable times for each event in the project. These computations are precisely a "mirror image" of the forward pass computations. First, the term "latest allowable" is use in the sense that the project terminal event must occur on or before some arbitrarily scheduled time, which will be denoted by T. Thus, the backward pass computations are initiated by specifying a value for T.

The latest allowable finish time for activities other than the final activity are then determined from network logic, which dictates that an activity must be completed

before its successor activities are started. The rules are summarized below.

Rule 1. The latest allowable finish time for the project terminal event (s) is set equal to either an arbitrary scheduled completion time for the project, T, or else to its earliest occurrence time computed in the forward pass computations (the zero-float convention).

$$TF(s) = T \text{ or } TE(s)$$

Rule 2. The latest allowable finish time for an arbitrary activity (i,j) is equal to the smallest, or earliest, of the latest allowable start times of its successor activities.

$LF(i,j)$ = Minimum LS of activities directly following activity (i,j)

$TF(j)$ = Minimum of {TF events succeeding event i minus the time duration of activities starting at node i}

Rule 3. The latest allowable start time for an arbitrary activity (i,j) is merely its latest allowable finish time minus the estimated activity duration time.

$$LS(i,j) = LF(i,j) - D(i,j)$$

$$TF(i) = \text{Min } \{TF(j) - D(i,j)\} \qquad [i<j]$$

As a result of applying the zero-float convention, the float along the critical path is zero, while the float along

all other paths is positive. The zero-float convention is adopted in the illustrative example shown in Figure 3.1, where the terminal event 4 is placed at time 11, i.e.,

$$TF(4) = TE(4) = 11$$

Next compute the latest allowable start times of activities (3,4) and (2,4) by applying Rule 3 for the backward pass:

$$LS(3,4) = LF(3,4) - D(3,4) = 11 - 6 = 5$$
$$TF(3) = \text{Min } \{ TF(3) - D(3,4) \} = 5$$
$$LS(2,4) = LF(2,4) - D(2,4) = 11 - 5 = 6$$

The crux of backward pass computations occurs at event 2. Here, the computation of the latest allowable finish time of activity (1,2) requires consideration of its two successor activities. Applying Rule 2 above for backward pass gives:

$$LF(1,2) = \text{Minimum of} \begin{matrix} LS(2,4) = 6 \\ LS(2,3) = 3 \end{matrix} = 3$$

or

$$TF(2) = \text{Minimum of} \begin{matrix} TF(4) - D(2,4) = 11 - 5 = 6 \\ TF(3) - D(2,3) = 5 - 2 = 3 \end{matrix} = 3$$

Finally, $LS(1,2) = LF(1,2) - D(1,2) = 2 - 2 = 0$. This

Table 3.2 Event Occurrence Times

Event i	Earliest TE(i)	Latest TF(i)
1	0	0
2	3	3
3	5	5
4	11	11

result can be used as a check in the computation when the zero-float convention is followed. If TF(4) = TE(4) = 11 for the terminal event, then TF(1) = TE(1) = 0 must result for the initial event. The event occurrence times of Figure 3.2 are shown in Table 3.2.

3.2.4 Critical Path Identification

Path float is the total float associated with a path. For a particular path activity, say (i,j), it is equal to the difference between either the latest and earliest start or the latest and earliest finish times. Thus, for activity (i,j), the path float is given by

$$F(i,j) = LS(i,j) - ES(i,j)$$

or

$$F(i,j) = LF(i,j) - EF(i,j)$$

The critical path is the one with the least path float. If the zero-float convention of letting TE(s)=TF(s) for the terminal network event is followed (Section 3.2.3), the critical path will have zero float. This is equivalent to not causing any delay in the completion of the project.

For the network in Figure 3.1, the critical path includes events 1-2-3-4 and tasks 1-3-5 (see Table 3.3).

Table 3.3 Earliest and Latest Start and Finish Times for Activities

Task	Events i	Events j	D	Earliest ES	Earliest EF	Latest LS	Latest LF	Slack	Critical Path
1	1	2	3	0	3	0	3	0	*
2	1	3	4	0	4	1	5	1	
3	2	3	2	3	5	3	5	0	*
4	2	4	5	3	8	6	11	3	
5	3	4	6	5	11	5	11	0	*

CHAPTER 4

LINEAR COST-DURATION FUNCTION

The main purpose of this chapter is the development of a procedure to determine activity schedules (i.e., generate project time-cost curves) to reduce the project duration time with a minimum increase in the project direct costs, by buying time along the critical path(s) where it can be obtained at least cost.

The Out Of Kilter Algorithm (OKA) is the procedure used for this purpose. OKA is a mathematically proven technique that generates the optimal (least expensive) trade-off time-cost curve (Fulkerson, 1961). In the presentation no attempt is made to describe the mathematical model in all its detail and implications. The presentation is limited to only what is necessary to provide a rationale for the procedure of the algorithm.

The various topics treated in this chapter will be illustrated using the simple network shown in Figure 4.1 and tabulated in Table 4.1 (Ford and Fulkerson, 1962). The example considers a single slope cost from normal to crash time completion for each activity, as shown in Figure 4.2. However, a real case of a nonlinear nonincreasing function between both time completions for an activity is explained

Table 4.1 Tabulated Figure 4.1

No. of Nodes = 4		No. of Arcs = 5		
Arc No.	i	j	Time Crash	Time Normal	Slope Cost
1	1	2	1	3	3
2	1	3	2	4	1
3	2	3	0	2	1
4	2	4	2	5	1
5	3	4	1	6	3

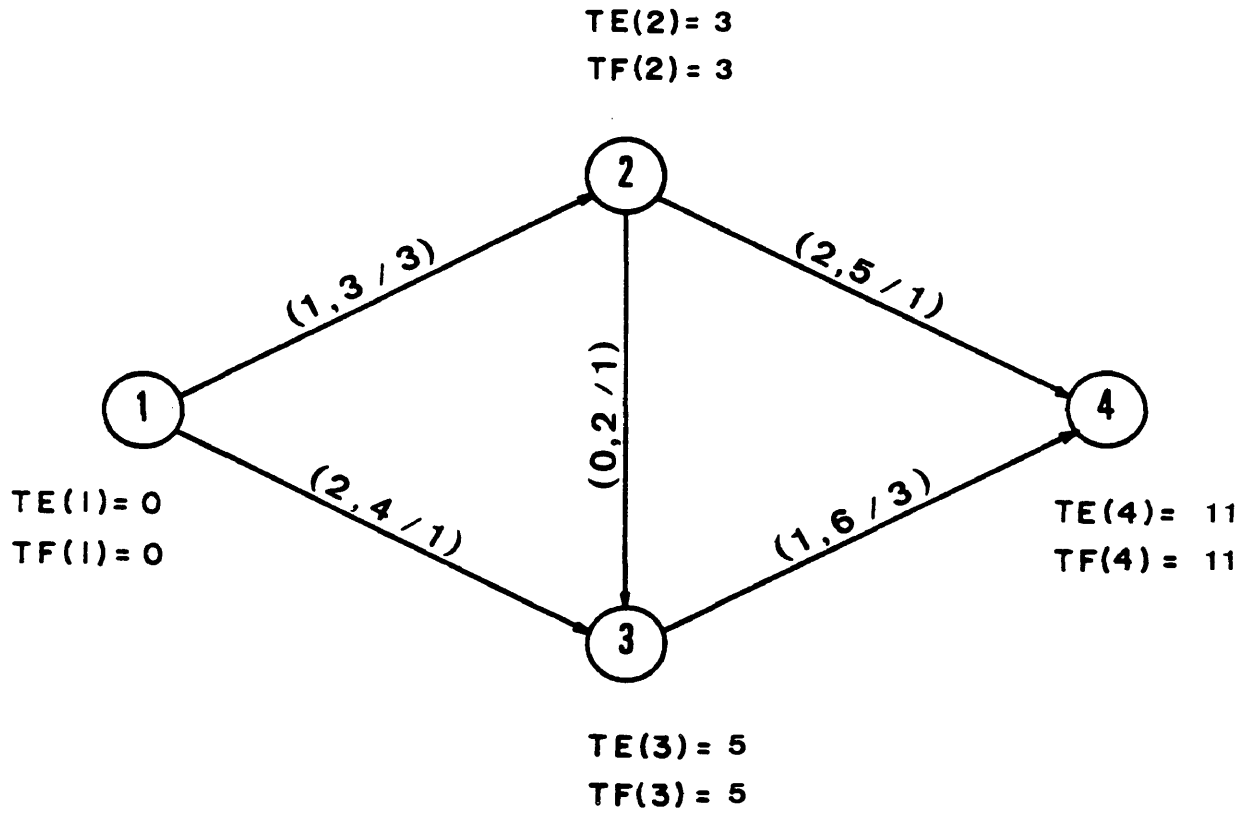


Figure 4.1 Simple Network

in detail at the end of this chapter.

4.1 Linear Programming Formulation of the CPM Time-cost Trade-off Problem

The procedure to generate an optimal project time-cost function consists of reducing time completion of activities in the critical path where it can be obtained at the cheapest cost. It is interesting to note that every time the project time completion is reduced, new critical paths can be added to the original one. Further reductions in the project are still possible, but now one must determine the economical way to cut all critical paths simultaneously.

Solutions have been carried out by Kelley (1961) (a linear programming approach) and Siemens (1971) (a hand-computed heuristic). The procedure used in this thesis, namely, the Out of Kilter Algorithm, which combines linear programming and network flows, will prove useful in easily generating project time-cost functions of any type.

4.1.1 Data Description

The numbers appearing along each activity in Figure 4.1 give three non-negative values [L(i,j),U(i,j)/C(i,j)], with L(i,j) <= U(i,j).

For example, [L(1,2) , U(1,2) / C(1,2)] = (1 , 3 / 3) indicates that for activity (1,2), the crash and normal performance times are 1 and 3 time units, respectively, and

the slope of the linear time-cost trade-off curve is 3 monetary units per time unit. In the applications, the time and monetary units are chosen so that the L(i,j)'s, U(i,j)'s and C(i,j)'s are all integers.

4.1.2 Activity Cost

The cost of doing job (i,j) is thus some known linear function

$$C(i,j) = K(i,j) - [C(i,j)*Y(i,j)] \qquad (4.1)$$

as shown in Figure 4.2, where K(i,j) is a constant representing the C(i,j) axis intercept for Y(i,j)=0, where Y(i,j) is the activity duration time, assuming such an activity duration were admissible. This function is defined over the range of Y(i,j), which must be somewhere between the crash time and the normal time. That is,

$$Y(i,j) >= L(i,j) \qquad (4.2)$$

$$Y(i,j) <= U(i,j) \qquad (4.3)$$

For dummy activities, L(i,j), U(i,j), and C(i,j) are all set to zero.

From Equation (4.1) it is seen that the activity cost equation is linear, and varies inversely with time, since C(i,j) is non-negative.

4.1.3 Total Project Direct Cost

For a given amount of time T in which to complete a project, the problem is to choose for each activity (i,j) a

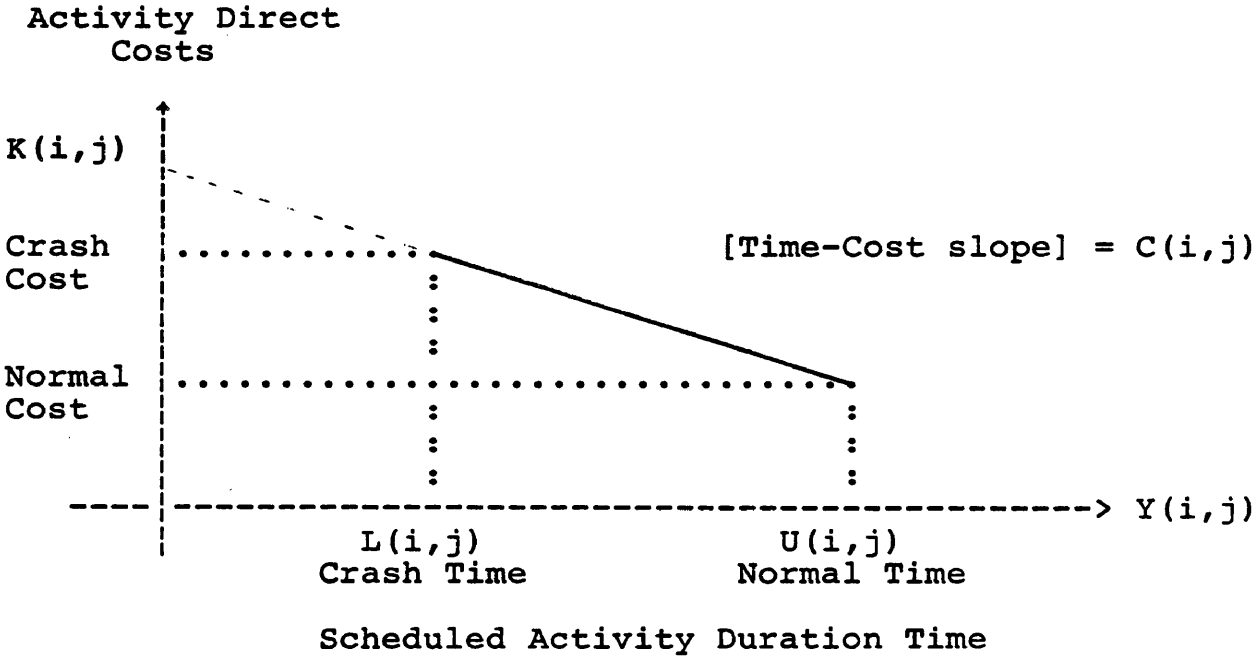


Figure 4.2 Time-Cost Trade-Off Curve Nomenclature

time Y(i,j), satisfying definitions (4.2) and (4.3), in such a way that the resulting project total direct cost

$$\text{SUM } C(i,j) = \text{SUM } [\ K(i,j) - C(i,j)*Y(i,j)\ ] \qquad (4.4)$$

is minimized.

Equation (4.4) can be expressed as

$$\text{SUM } C(i,j) = \text{SUM } K(i,j) - \text{SUM } [C(i,j)*Y(i,j)] \qquad (4.5)$$

From this it is obvious that SUM C(i,j), the project total direct cost, can be minimized by maximizing the subtractive term. SUM [C(i,j)*Y(i,j)] is always positive since C(i,j) >= 0 and Y(i,j) >= 0. The curve of Equation (4.5) is convex rather than linear since the C(i,j) in the sums have different values, as shown in Figure 4.3.

Thus, Equation (4.5) is minimized by maximizing the function

$$Z = \text{SUM } [C(i,j)*Y(i,j)] \qquad (4.6)$$

4.1.4 Precedence Constraints

If TE(j) represents the earliest possible time of the realization of event j, the the precedence constraints imply that

$$TE(i) + Y(i,j) \ <= \ TE(j) \qquad (4.7)$$

$$TE(1) = 0 \qquad \text{for } j=2,3,\ldots,n$$

which simply means that, if event i precedes event j, then the latter cannot be realized before Y(i,j) time units after

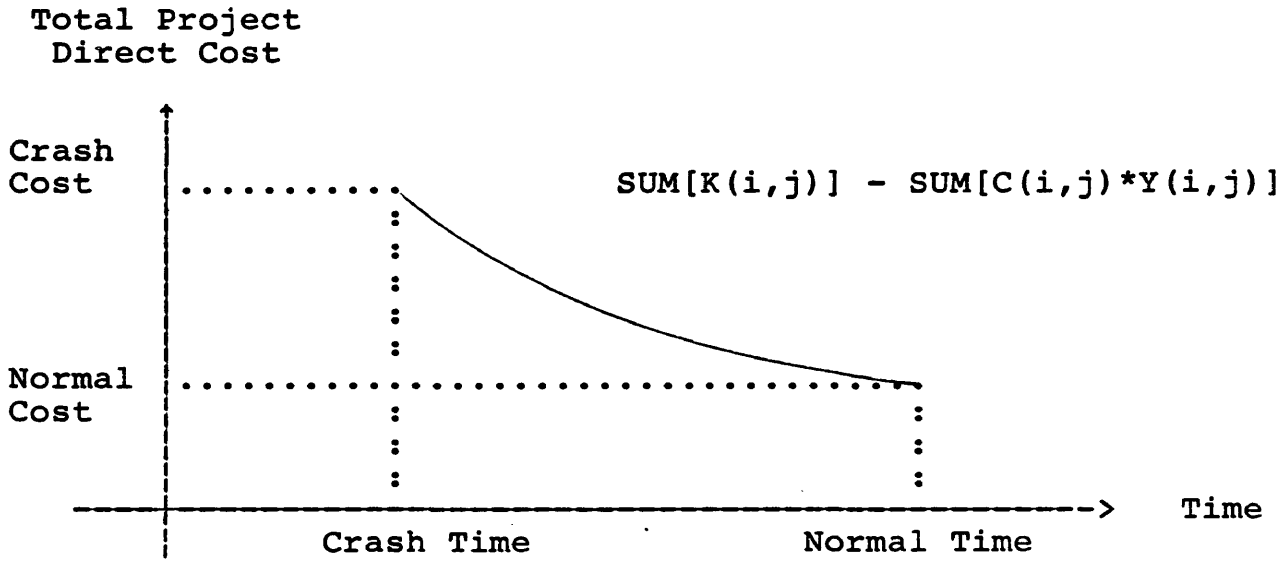


Figure 4.3 Project Cost Curve

event i has been realized.

4.1.5 The Primal Formulation

The problem of optimal activity durations to accomplish the project in a specified time T can be stated as follow. Determine activity durations {Y(i,j)} and the earliest possible node-realization times {TE(j)} that maximize

$$F[Y(i,j)] = \text{SUM}\ [C(i,j)*Y(i,j)] \qquad (4.6)$$

Subject to the following constraints. (The dual variables shown will appear in the next formulation)

			Dual Variables	
TE(i) + Y(i,j) - TE(j)	<=	0	f(i,j)	(4.7)
TE(End)	<=	T	v	(4.8)
Y(i,j)	>=	L(i,j)	g(i,j)	(4.2)
-Y(i,j)	>=	-U(i,j)	h(i,j)	(4.3)

Inequality (4.7) expresses the precedence constraints. Inequality (4.8) expresses the requirement that the project duration must be less than or equal to some desired duration of the project T. Inequalities (4.2) and (4.3) re-express the range of Y(i,j) as described above.

The linear programming formulation for this problem has been written out in full in Table 4.2. While this problem could be solved using the simplex method to find the schedule Y(i,j)'s which satisfy all of the constraints and

Table 4.2 Primal Problem (Max F[Y])

	Y(1,2)	Y(1,3)	Y(2,3)	Y(2,4)	Y(3,4)			
F[Y] =	3	4	2	5	6			
Subject to:								
	1	1	0	0	0	=	1	W1
	-1	0	1	1	0	=	0	W2
	0	-1	-1	0	1	=	0	W3
	0	0	0	-1	-1	=	-1	W4
				All	Y(i,j)	>=	0	

at the same time maximize F[Y(i,j)], a more efficient network flow algorithm will be given.

4.1.6 The Dual Formulation

From the well known Primal-Dual algorithm for linear programs, it can be shown that to every primal formulation there corresponds a related linear programming problem which is called the dual formulation.

The dual solution to the above maximizing problem will now be considered. The dual problem is:

Min Z = T*V + SUM [U(i,j)*g(i,j)] - SUM [L(i,j)*h(i,j)]

subject to the constraints:

$$f(i,j) + g(i,j) - h(i,j) = L(i,j) \qquad (4.9)$$

$$\text{SUM}\,[f(i,j)-f(j,i)] = \begin{cases} v, & i=1 \\ 0, & 1<i<n \\ -v, & i=n \end{cases} \qquad (4.10)$$

Note that all of the dual constraints are equalities, since all of the primal variables are unrestricted in sign, and the dual variables are all constrained to be non-negative, since the primal constraints are all inequalities.

The flow in the arc (i,j) is f(i,j), and it is measured in units of cost per unit time [the same as the C(i,j)'s]. Equations (4.10) are flow equations, and they express the fact that flow must be conserved at all nodes of the network

Table 4.3 Dual Problem (Min G[W])

	W1	W2	W3	W4		
G[Y] =	1	0	0	-1		
Subject to:						
	1	-1	0	0	>=	3
	1	0	-1	0	>=	4
	0	1	-1	0	>=	2
	0	1	0	-1	>=	5
	0	0	1	-1	>=	6

Infinite >= Wi >= Infinite

i= 1,2,3,4

except at the starting and ending events of the project, where the flow out of the initial event (node) and the flow into the terminal event (node) is v.

Applying the dual theorem of linear programming to the network in Figure 4.1 leads to the dual formulation given in Table 4.3.

4.2 The Network-Flow Algorithm

This algorithm was developed by Fulkerson (1961); it closely follows primal-dual arguments and procedures. In essence, start at the longest possible project duration and proceed to shorten that duration in such a way that the results are always primal and dual feasible. Appendix B shows the hand computational procedure of this algorithm applied to Figure 3.1.

4.2.1 Start--Initial Schedule Determination

The initial project schedule is computed by finding TE(i) and TF(i) for each event i as defined by Equations (4.1) and (4.2), where the activity durations are taken at their normal level, Y(i,j) = U(i,j).

This gives the project event times and these times define the project normal duration and schedule. Then Critical Paths, Critical Activities, and Critical Events are defined.

4.2.2 Labeling Procedure

The labeling procedure specifies the method used in attempting to increase the flow in the project network, in keeping with the flow maximizing requirements of the mathematical model. The purpose of this procedure, however, is to find minimum-cost activities.

In the labeling procedure, events j are assigned labels of the form (i,E). The i indicates that event j was labeled from event i. E is the largest cost flow possible along the path from the starting event of the project (node 1) to j.

The labeling procedure begins with node 1 , which is labeled (-,infinite), indicating that infinite flow is available at the starting event of the project. The initial flows [F(i,j)] through any arc are zero. For any labeled node check all unlabeled nodes connected to it for which arc(i,j) is a Critical Activity, and either:

a. $F(i,j) < C(i,j)$ Label node j with [i,E(j)], where $E(j) = \text{Min } [E(i), C(i,j)-F(i,j)]$.

or

b. $F(i,j) > 0$ Label node i with [j,E(i)], where $E(i) = \text{Min } [E(j), F(i,j)]$.

Continue the labeling until either a breakthrough or nonbreakthrough occurs. Breakthrough indicates that the final event (Sink Node) will have been labeled.

Nonbreakthrough indicates that it has not been possible to label the final event.

If there is a breakthrough, go to the Flow Change Procedure, Section 4.2.3. If not, go to Project Shortening procedure, Section 4.2.4.

4.2.3 Flow Change Procedure

If, as a result of the labeling procedure, breakthrough has occurred, two conditions can happen:

E(Sink Node)=Finite

Breakthrough to final event with a label of finite flow. This procedure defines the minimum-cost activity along one CP. As long as breakthrough is possible, the flow change procedure saturates the capacity of at least one activity along a CP. The set composed of these saturated minimum-cost activities represents the ones whose duration must be reduced, and the cost of such reduction is guaranteed to be minimal.

Then change the flow along the newly discovered CP by the amount E(Sink Node). Either reduce or increase all residual capacities [C(i,j)-F(i,j)] by a factor E(Sink Node) for forward or reverse arcs, respectively. Continue the algorithm by discarding the old labels and going back to the Labeling Procedure (Section 4.2.2).

E(Sink Node)=Infinite

Breakthrough to final event with a label of infinite flow. There exists a CP through the project network all of whose arcs have infinite capacity. The arcs have infinite capacity only when they are at crash duration. Therefore a CP of activities at crash duration exists and the algorithm terminates. The project duration is now at its crash duration (shortest possible) and no further shortening of the project is possible.

4.2.4 Project Shortening Procedure

As a result of the labeling procedure, the final event has not been labeled. Therefore it is not possible to increase the flow in the network with the current schedule in effect, so a new shorter schedule is required.

When a nonbreakthrough condition is obtained, it must be that the CPs have been exhausted. The project shortening procedure then reduces the total project duration by Delta (the biggest possible reduction). Delta is limited by either: (a) The fact that another path becomes a CP, or (b) an activity has been reduced to its lower bound, L(i,j). The nodes of the network can be divided into two mutually exclusive and totally exhaustive subsets: the labeled nodes and the unlabeled nodes. Single out the following subsets of arcs:

Z(1)={Arc(i,j): i labeled, j unlabeled, any arc}

Z(2)={Arc(i,j): i unlabeled, j labeled, critical arc}

Define:

Delta(1) = Min [T(j) - T(i) - K(i,j)], [Z(1),Non-CP]

Delta(2) = Min [T(j) - T(i) - L(i,j)], [Z(1),CP]

Delta(3) = Min [K(i,j) + T(i) - T(j)], [Z(2),CP]

Delta = Min [Delta(1),Delta(2),Delta(3)]

Change TF(i) and TE(i) for Critical Events (or just TF(i) for noncritical ones) for unlabeled nodes by subtracting Delta from them. Leave the event times of labeled events as they are. This provides the new project schedule.

The next activity durations [Y(i,j)], are calculated based on the schedule changes made by this step, and new CPs will become part of the network.

Whenever the duration of any activity has been reduced to its lower bound L(i,j) (crash time), the capacity is set to infinite.

Continue the algorithm by discarding the old labels and going back to the Labeling Procedure (Section 4.2.2).

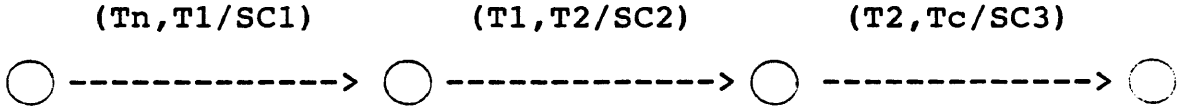
4.3 Formulation for a Nonlinear but Convex Activity Time-cost Trade-off Function

A nonlinear nonincreasing function for an activity

(i,j) is shown in Figure 4.4. To partition activity (i,j) into subactivities defines a new normal and crash time completion for each subactivity as shown in Table 4.4.

There are two methods to include this partition into the least-cost algorithm:

1. New activities (3) and events (2) are added into the network as shown below



The precedence constraint will prevail since the least-cost algorithm always selects the activity with the least slope cost (notice that SC1 < SC2 < SC3). Therefore, subactivity 1 is performed first, subactivity 2 second, and then subactivity 3.

2. Start with only subactivity 1 (least expensive slope cost) in the network. At the time the least-cost algorithm reduces both the time activity completion to its crash time completion and its slope cost to zero, then instead of adding a second arc with infinite capacity, subactivity 2 (next least expensive slope cost) is added to the network. This method does not violate any restriction either.

The latter method is implemented in the computer program, resulting in fewer activities and events to handle.

Table 4.4 Partition of Activity (i,j)

Sub-activity	Time		Slope Cost
	Crash	Normal	
1	T1	Tn	SC1
2	T2	T1	SC2
3	Tc	T2	SC3

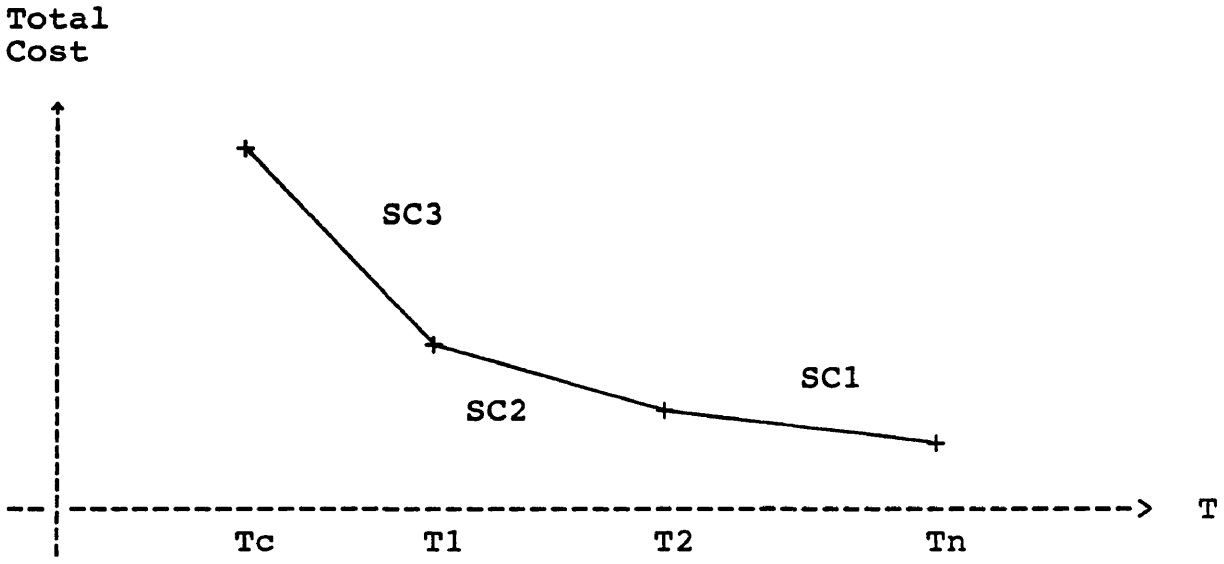


Figure 4.4 Partition of Activity (i,j)

CHAPTER 5

DYNAMIC NETWORK-BASED PLANNING TO DEFINE OPTIMAL PREPRODUCTION TIME COMPLETION

In calculating the cash flows required for a new mining project, it is often convenient to separate the project into preproduction and production periods. The beginning of the production period depends on the completion of all activities involved in the preproduction period.

The relationship between the present value of the negative cash flow (PVOUT) and the present value of the positive cash flow (PVIN) defines the return on venture capital on which the investment decision (go/no go) is based (Wells, 1978). These relationships may be expressed by the net present value (NPV) or the present value ratio (PVR). For this analysis the NPV will be used as the optimization criterion, where:

$$NPV = PVOUT - PVIN$$

There are two fundamental principles of investment optimization, namely, to decrease or defer expenditures and/or to increase or bring revenues closer in time. This thesis deals directly with the second principle of bringing revenues closer in time, and its influence on NPV because of the time-cost trade-off made in the preproduction period with increment investment.

The purpose is to define the optimal preproduction time completion by maximizing the NPV of the project, subject to the successful completion of all programs involved in the preproduction stage.

5.1 Network-based Planning

To bring revenues (production period) closer in time is necessary to schedule the completion of the preproduction period. Systematic scheduling methods such as the critical path method (CPM) facilitate the planning and control of the preproduction period. The following steps are needed to define the preproduction network:

5.1.1. Project Planning

The activities making up the preproduction period in a project are defined, and their technological dependencies upon one another are shown explicitly in the form of a network diagram. In other words, a network diagram showing plan of work for the preproduction period and all types of constraints is developed.

5.1.2 Time Estimation

Estimates of the time required to perform each of the network activities are made; these estimates are based upon assumed infinite resources.

5.1.3 Basic Scheduling

The basic scheduling computations give the earliest and latest allowable start and finish times for each activity, and as a by-product they identify the critical path through the network and indicate the amount of slack or float time associated with the noncritical paths. The critical path is the initial preproduction time completion on which the NPV is based.

5.2 Time-Cost Trade-Offs

To determine the cost of reducing the project completion time, time-cost trade-offs of activity performance times must be considered for all activities on the network.

The time-cost trade-off problem is directed to the task of determining a schedule of project activities which considers explicitly the indirect as well as the direct costs and attempts to minimize their sum. As a result, let C(T) be the minimal discounted value of costs of a preproduction plan with completion date T.

5.3 Production Revenues

Let R(T) be the present value of revenues as a result of the production period and a preproduction completion date T. R(T) depends on the rate of discount used and on the size and timing of the production revenues.

5.4 Optimal Project Schedule

The purpose is to optimize the preproduction time completion by maximizing its NPV. Therefore, the NPV for a preproduction completion date T is:

$$NPV(T) = R(T) - C(T)$$

The optimal preproduction schedule is presented graphically at Ts. It is easy to ascertain that when there is a solution, the following first-order condition will hold at Ts since we expect to maximize their difference:

$$\frac{dR}{dT}(Ts) = \frac{dC}{dT}(Ts)$$

5.5 Project Control

When the network plans have been developed to a satisfactory extent, they are prepared in final form for use in the field. The project is controlled by checking off progress against the schedule and by assigning and scheduling manpower and equipment and analyzing the effects of delays. Whenever major changes are required in the schedule, the network is revised accordingly and a new schedule is computed.

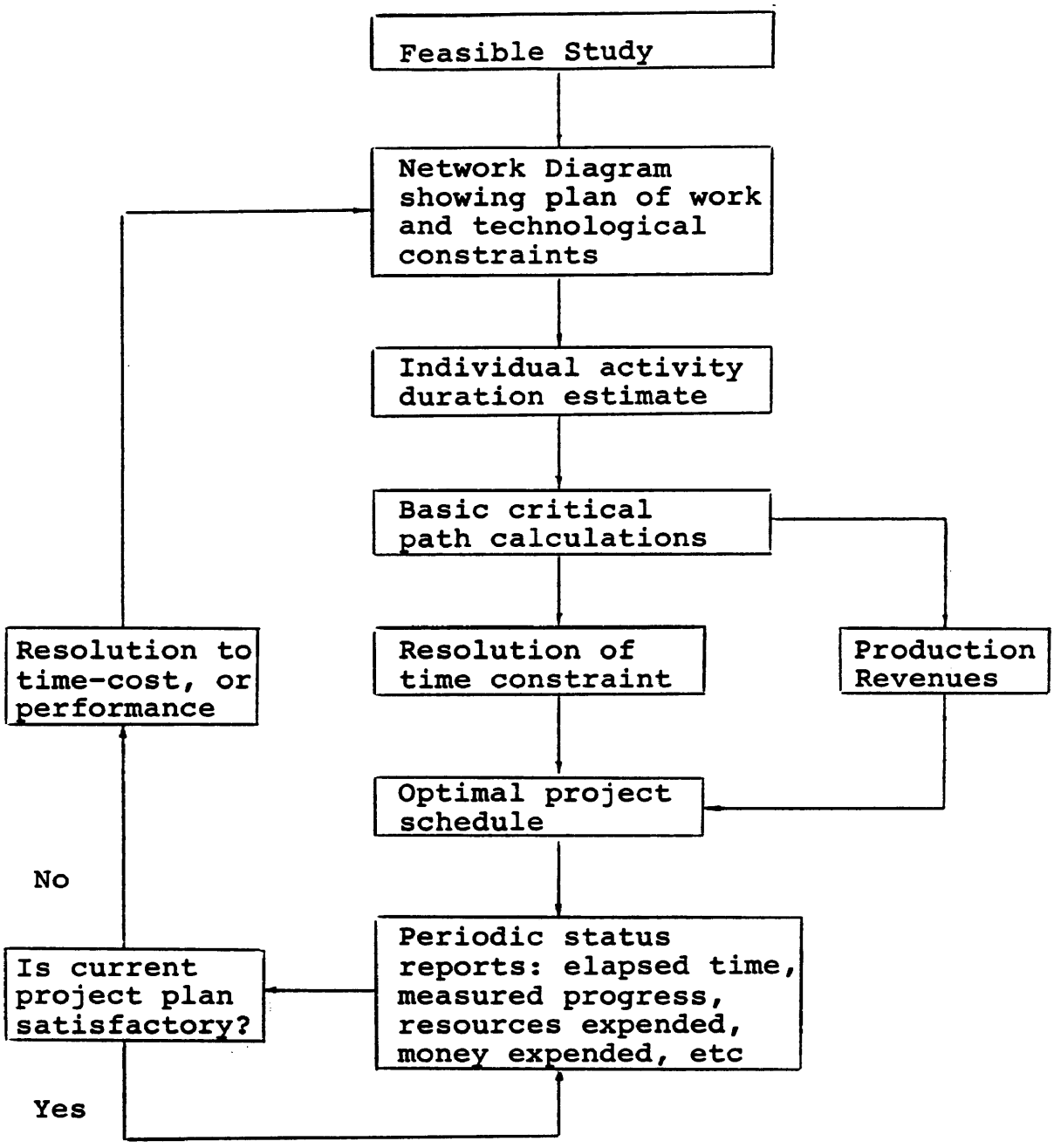


Figure 5.1 Algorithm Solution Flow Chart

CHAPTER 6

A HYPOTHETICAL MINE DEVELOPMENT PROJECT

This example shows the procedure for scheduling all activities involved in a mine development project in order to obtain its optimal preproduction schedule. This example also shows how a network-based planning algorithm can be used to plan, schedule, and control the development of this hypothetical mine from the initial surface surveys to completion of the underground development work.

The hypothetical nondetailed property considered as a case study consists of 18 activities (Table 6.1). The values assigned are purely hypothetical, and are intended only to illustrate the method of solution.

6.1 Critical Path Method

A critical path study involves the following general steps:

1. Establish a rough overall plan of attack for the entire project, noting special conditions such as climate, water and power availability, etc.

2. List all jobs necessary for the overall project (see Table 6.2). The degree of detail in the job breakdown should be left to the planner's discretion. For large projects of broad scope, it is logical to consider a series of critical

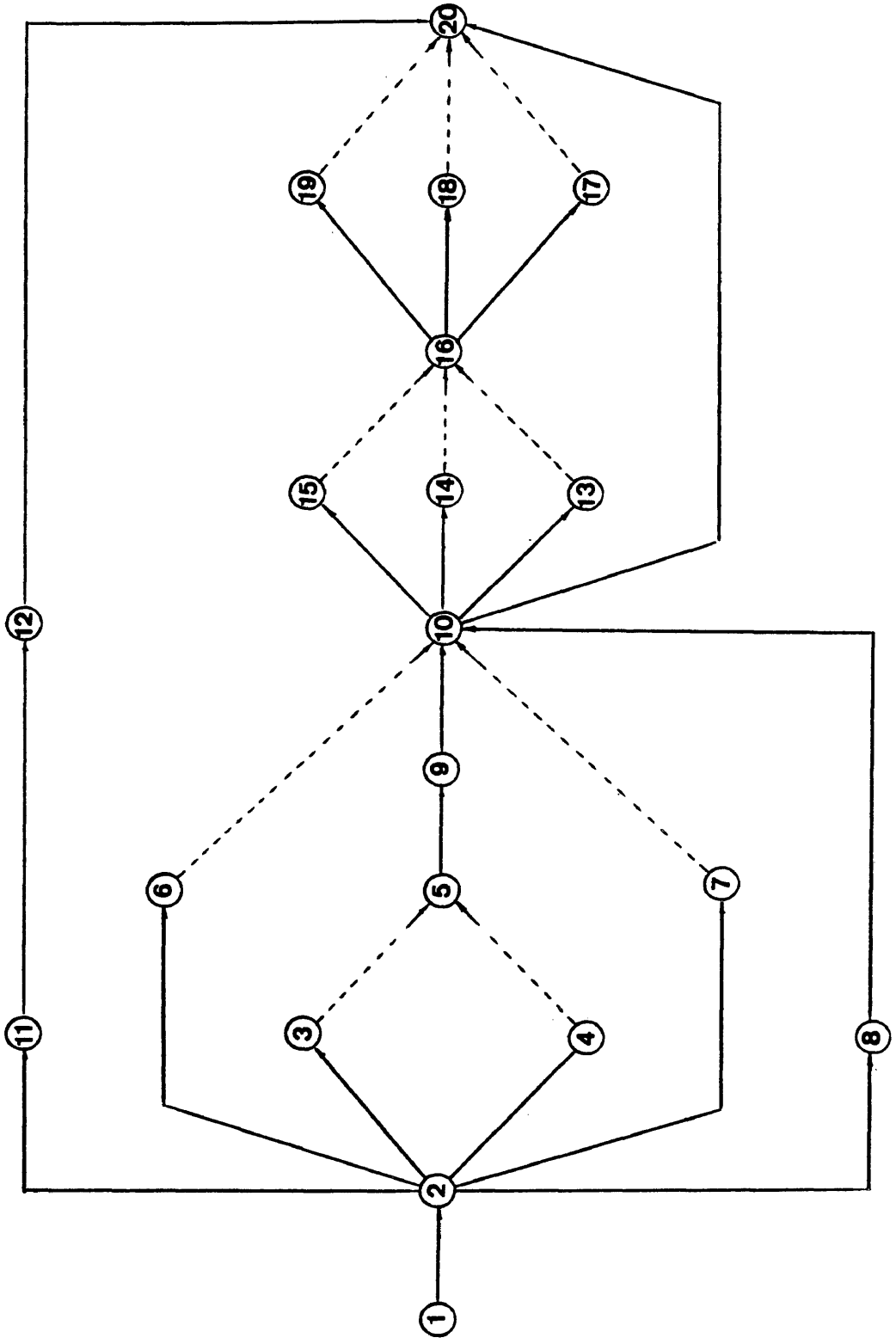


Figure 6.1 Precedence Diagram of the Hypothetical Mine Development Project plus some Dummy Activities

Table 6.1 Hypothetical Mine Development Project
Initial Activities List

No.	Activity description
1.	Prepare initial site.
2.	Construct powder magazine.
3.	Construct head frame.
4.	Construct electrical distribution system.
5.	Construct hoist house.
6.	Construct compressor house.
7.	Sink shaft.
8.	Equip shaft.
9.	Construct ore pass.
10.	Construct ore dump.
11.	Construct ore pocket.
12.	Drill ventilation shaft A.
13.	Drill ventilation shaft B.
14.	Drill Ventilation shaft C.
15.	Remove temporary facilities.
16.	Drift to ore body A.
17.	Drift to ore body B.
18.	Drift to ore body C.

Table 6.2 Preproduction Activities List

Task #s	Nodes #s Begin	End	Normal Time	Cost	Activity description
1	1	2	6	60	Prepare initial site
2	2	3	2	4	Construct powder magazine
3	2	4	4	16	Construct head frame
4	3	5	0		Dummy
5	4	5	0		Dummy
6	2	6	6	48	Construct electrical distribution
7	2	7	5	25	Construct hoist house
8	2	8	4	28	Construct compressor house
9	6	10	0		Dummy
10	7	10	0		Dummy
11	8	10	0		Dummy
12	5	9	6	60	Sink shaft
13	9	10	3	15	Equip shaft
14	10	13	5	25	Construct ore pass
15	10	14	2	8	Construct ore dump
16	10	15	3	18	Construct ore pocket
17	3	16	0		Dummy
18	14	16	0		Dummy
19	15	16	0		Dummy
20	2	11	6	54	Drill ventilation shaft A
21	11	12	6	54	Drill ventilation shaft B
22	12	20	6	72	Drill Ventilation shaft C
23	10	20	3	6	Remove temporary facilities
24	16	17	8	60	Drift to ore body A
25	16	18	8	60	Drift to ore body B
26	16	19	8	60	Drift to ore body C
27	17	20	0		Dummy
28	18	20	0		Dummy
29	19	20	0		Dummy
Total:			$6,730,000		

Note: The Normal Cost column is in thousands of dollars.

path studies for various segments of the project, each in greater detail than the overall. Dummy activities are also placed on the list.

3. List all known job interrelationships (see Table 6.3). For each job, this includes noting those jobs that must be completed before it can be started and those jobs that follow immediately.

4. Construct the precedence diagramming network (Figure 6.1), which is a graphic presentation of the planned sequence of individual jobs in the project.

5. Refine and revise the network diagram until a final plan is obtained that is suitable for estimating, scheduling, and operational control.

6.2 Time-cost Trade-off Function Generation

The least expensive time-cost trade-off function for the hypothetical mine development project is generated by executing a program called LCS (Least Cost Schedule). This program is written in FORTRAN 77 and is available for the IBM PC and 100% compatibles such as COMPAQ. Its entire code is shown in Appendix B.

6.2.1 General Data

The data input of the entire hypothetical mine development project is shown in Table 6.4. The first two data needed in a data file are the number of nodes and the

Table 6.3 Preproduction Activities Precedence

Task #s	Precendent Tasks				Activity description
1	0	-	-	-	Prepare initial site
2	1	-	-	-	Construct powder magazine
3	1	-	-	-	Construct head frame
4	2	-	-	-	Dummy
5	3	-	-	-	Dummy
6	1	-	-	-	Construct electrical distribution
7	1	-	-	-	Construct hoist house
8	1	-	-	-	Construct compressor house
9	6	-	-	-	Dummy
10	7	-	-	-	Dummy
11	8	-	-	-	Dummy
12	4	5	-	-	Sink shaft
13	12	-	-	-	Equip shaft
14	9	10	11	13	Construct ore pass
15	9	10	11	13	Construct ore dump
16	9	10	11	13	Construct ore pocket
17	14	-	-	-	Dummy
18	15	-	-	-	Dummy
19	16	-	-	-	Dummy
20	1	-	-	-	Drill ventilation shaft A
21	20	-	-	-	Drill ventilation shaft B
22	21	-	-	-	Drill Ventilation shaft C
23	9	10	11	13	Remove temporary facilities
24	17	18	19	-	Drift to ore body A
25	17	18	19	-	Drift to ore body B
26	17	18	19	-	Drift to ore body C
27	24	-	-	-	Dummy
28	25	-	-	-	Dummy
29	26	-	-	-	Dummy

number of activities, 20 and 29 respectively. The first activity to perform must begin at the starting node 1, and the last activity must end on the sink or ending node, such as node 20 for this case study.

Each activity in the project is stated in one data line. Therefore, the next 29 data lines will be read by the LCS program. Fewer activities in the data file will cause an execution error at the time the program is reading data. On the other hand, more activities will cause missing data input to the program, and consequently miscalculations.

In order to follow the network-flow algorithm used in the LCS program, it is important to distinguish the difference between numbers assigned to activities and the numbers assigned to nodes, such as activity #2 has to be performed between nodes 2 and 3.

6.2.2 Activity Data

Since the least-cost time-cost trade-off function is an extension of the Critical Path Method, then more planning is required in the construction of the data. The case study data (Table 6.4) was based on tables such as Tables 6.5 and 6.6 for activity # 26 (Drift to ore body C).

Cost Slope is a trade-off of some extra cost needed to shorten the time completion of the activity by one unit of time. Figures 6.2 and 6.3 demonstrate the stepwise

Table 6.4 Case Study Data

# Nodes , # Activities									Task #s
20,29									
1, 2,	600 ,	2 ,	5,6,100,	4,5,160,	0,0,	0			1
2, 3,	40 ,	1 ,	1,2, 20						2
2, 4,	160 ,	1 ,	3,4, 40						3
3, 5,	0 ,	1 ,	0,0, 0						4
4, 5,	0 ,	1 ,	0,0, 0						5
2, 6,	480 ,	2 ,	4,6, 80,	3,4,130,	0,0,	0			6
2, 7,	250 ,	1 ,	3,5, 50						7
2, 8,	280 ,	1 ,	3,4, 70						8
6,10,	0 ,	1 ,	0,0, 0						9
7,10,	0 ,	1 ,	0,0, 0						10
8,10,	0 ,	1 ,	0,0, 0						11
5, 9,	600 ,	2 ,	4,6,100,	3,4,180,	0,0,	0			12
9,10,	150 ,	1 ,	1,3, 50						13
10,13,	250 ,	1 ,	2,5, 50						14
10,14,	80 ,	1 ,	1,2, 40						15
10,15,	180 ,	1 ,	1,3, 60						16
3,16,	0 ,	1 ,	0,0, 0						17
14,16,	0 ,	1 ,	0,0, 0						18
15,16,	0 ,	1 ,	0,0, 0						19
2,11,	540 ,	2 ,	4,6, 90,	3,4,120,	0,0,	0			20
11,12,	540 ,	1 ,	4,6, 90						21
12,20,	720 ,	1 ,	4,6,120						22
10,20,	60 ,	2 ,	2,3, 20,	1,2, 30,	0,0,	0			23
16,17,	600 ,	1 ,	4,8, 75						24
16,18,	600 ,	2 ,	6,8, 75,	4,6,120,	0,0,	0			25
16,19,	600 ,	3 ,	6,8, 75,	4,6,120,	3,4,200		--->		26
17,20,	0 ,	1 ,	0,0, 0						27
18,20,	0 ,	1 ,	0,0, 0						28
19,20,	0 ,	1 ,	0,0, 0						29

Table 6.5 Task # 26 Data

Beginning Node = 16 Ending Node = 19

Subrange	Time Crash	Time Normal	Cost Crash	Cost Normal	Slope Cost
1	6	8	750,000	600,000	75,000
2	4	6	990,000	750,000	120,000
3	3	4	1,190,000	990,000	200,000

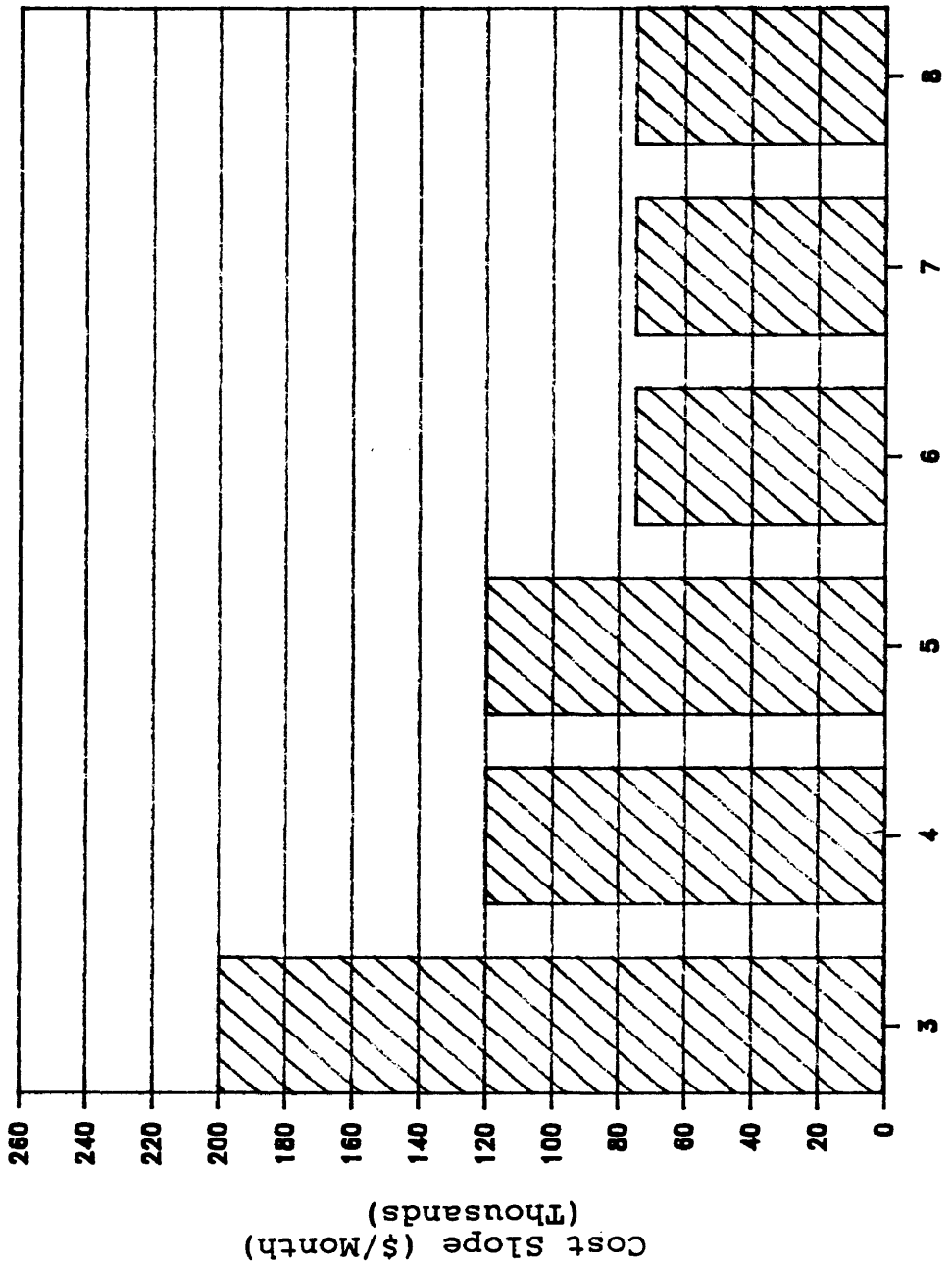


Figure 6.2 Task Cost Slope vs. Time Completion

Table 6.6 Task # 26 Time Completion and its Associated Total Cost

Time Completion (Months)	Total Cost Associated
8	$ 600,000
7	$ 675,000
6	$ 750,000
5	$ 870,000
4	$ 990,000
3	$ 1,190,000

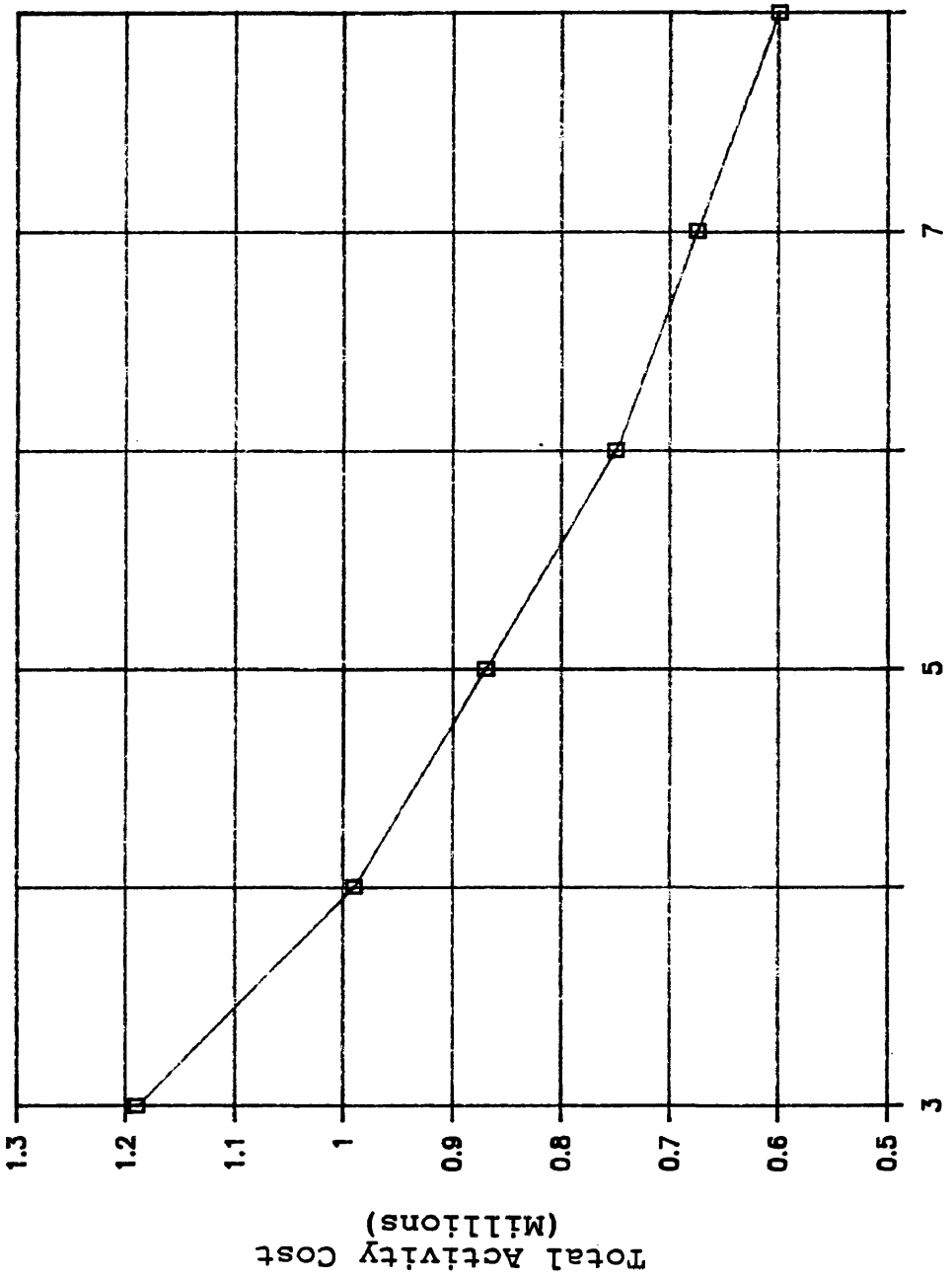


Figure 6.3 Total Task Cost vs. Time Completion

(nonlinear), nonincreasing (decreasing) and convex functions that each activity must follow as a pattern. Figure 6.2 is the derivative function of the total activity cost in Figure 6.3.

6.2.3 Program Results Description

One computer run generates the least-cost preproduction completion time (see Table 6.7). A column by column description of Table 6.7 is given below:

1. A counter of the number of times a new possible preproduction time completion is obtained.

2. Project time duration is a possible preproduction time completion.

3. Total cost is the direct cost assocciated with the project time duration. Total cost is the total slope cost multiply by column 5 (number of time units to shorten) and then adding to the last total cost (see Table 6.8).

4. Total slope cost is a cost associated with every time unit that the project time duration is shortened. If more than one activity is shortened at the same time, then the total slope cost is the summation of all their slope cost (see Table 6.9).

5. e/o Cut is the number of time units to shorten time activity durations listed on right-hand side of Table 6.7.

6. Least expensive activities whose time durations will

Table 6.7 Time-Cost Trade-Off Program Output

(1)	(2)	(3)	(4)	(5)	(6)					
No.	Project Time Duration	Total Cost	Total Slope Cost	e/o Cut	Activities to Cut					
1	32	6,730	0	0	0	0	0	0	0	0
2	31	6,770	40	1	3	0	0	0	0	0
3	30	6,820	50	1	13	0	0	0	0	0
4	29	6,870	50	1	13	0	0	0	0	0
5	27	6,970	50	2	14	0	0	0	0	0
6	26	7,070	100	1	1	0	0	0	0	0
7	24	7,270	100	2	12	0	0	0	0	0
8	23	7,380	110	1	14	16	0	0	0	0
9	22	7,540	160	1	1	0	0	0	0	0
10	21	7,810	270	1	12	20	0	0	0	0
11	20	8,125	315	1	20	24	25	26	0	0
12	19	8,440	315	1	21	24	25	26	0	0
13	18	8,845	405	1	21	24	25	26	0	0
14	17	9,280	435	1	20	24	25	26	0	0

Table 6.8 Total Cost Evaluation

No.	Time Duration	Total Cost	Total Slope Cost	e/o Cut
7	24	7,270	-	-
8	23	7,380	110	1

Table 6.9 Total Slope Cost Evaluation

Project Time Duration	Task No.	Task Slope Cost	Time Units to Cut	Task Time Completion	Project Slope Cost
24	14	50	-	3 *	-
	16	60	-	3	-
23	14	50	1	2	110
	16	60	1	2	

* Activity # 14 time completion was already shortened in iteration No. 5 (Table 6.5) from its normal time duration of 5 to 3 months.

be shortened in order to match desired project time duration.

6.3 Analysis of Results

The activities (column 6, Table 6.7) whose time durations will be shortened in order to match desired project time duration (column 2) lead us to the analysis of the maximum time-shortening (column 5) and the increasing total slope cost (column 4).

6.3.1 Maximum Time Shortening

The reason why the time-shortening magnitudes (column 5, Table 6.7) are different is because the algorithm seeks the maximum time shortening possible by selecting the minimum value of either:

1. The minimum time shortening necessary to make the time duration of a critical activity be reduced to its crash time duration.

2. The minimum time shortening necessary to make a noncritical activity part of the Critical Path.

Therefore the time-shortening magnitudes vary from one step to another. The reason to shorten twice the time duration of activity #13 each by 1 time unit (steps 3 and 4) is because the first time shortening makes one noncritical activity part of the critical path.

6.3.2 Increasing Total Slope Cost

The more the project time duration is shortened, the bigger the total slope cost. Since any activity can be performed at a different cost slope, then its time duration can be shortened at different steps.

For instance, activity # 1 time duration is shortened from 6 to 5 months (step 6) for a time-cost trade-off of $100,000, and again shortened from 5 to 4 months (step 9) for a time-cost trade-off of $ 160,000. Steps 7 and 8 have less expensive activities to shorten than step 9 involving activities #12, #14, and #16.

6.4 Optimal Preproduction Schedule

The optimal preproduction schedule can be reached in three different ways: 1. Highest Net Present Value, 2. Fixed Project Budget, and 3. Fixed Preproduction Time Completion

6.4.1 Highest Net Present Value

The purpose here is to define the best hypothetical mine development schedule with the highest Net Present Value possible, so difference between Present Values of Revenues (PVr) and Costs are plotted in Figure 6.5 and tabulated in Table 6.10.

The Present Value of Revenues (PVr) is equal to $7,000,000. Shortening the preproduction period will result in a higher PVr (see Figure 6.5).

Table 6.10 Present Values of Costs and Revenues (Thousands of Dollars)

Preproduction Time Completion	PV Revenue	PV Cost	NPV	
32	7,000	6,730	270	
31	7,070	6,770	300	*
30	7,141	6,820	321	*
29	7,212	6,870	342	*
27	7,357	6,970	387	*
26	7,431	7,070	361	*
24	7,580	7,270	310	*
23	7,656	7,380	276	*
22	7,732	7,540	192	
21	7,810	7,810	0	
20	7,888	8,125	-237	
19	7,967	8,440	-473	
18	8,046	8,845	-799	
17	8,127	9,280	-1,153	

* Feasible solutions.

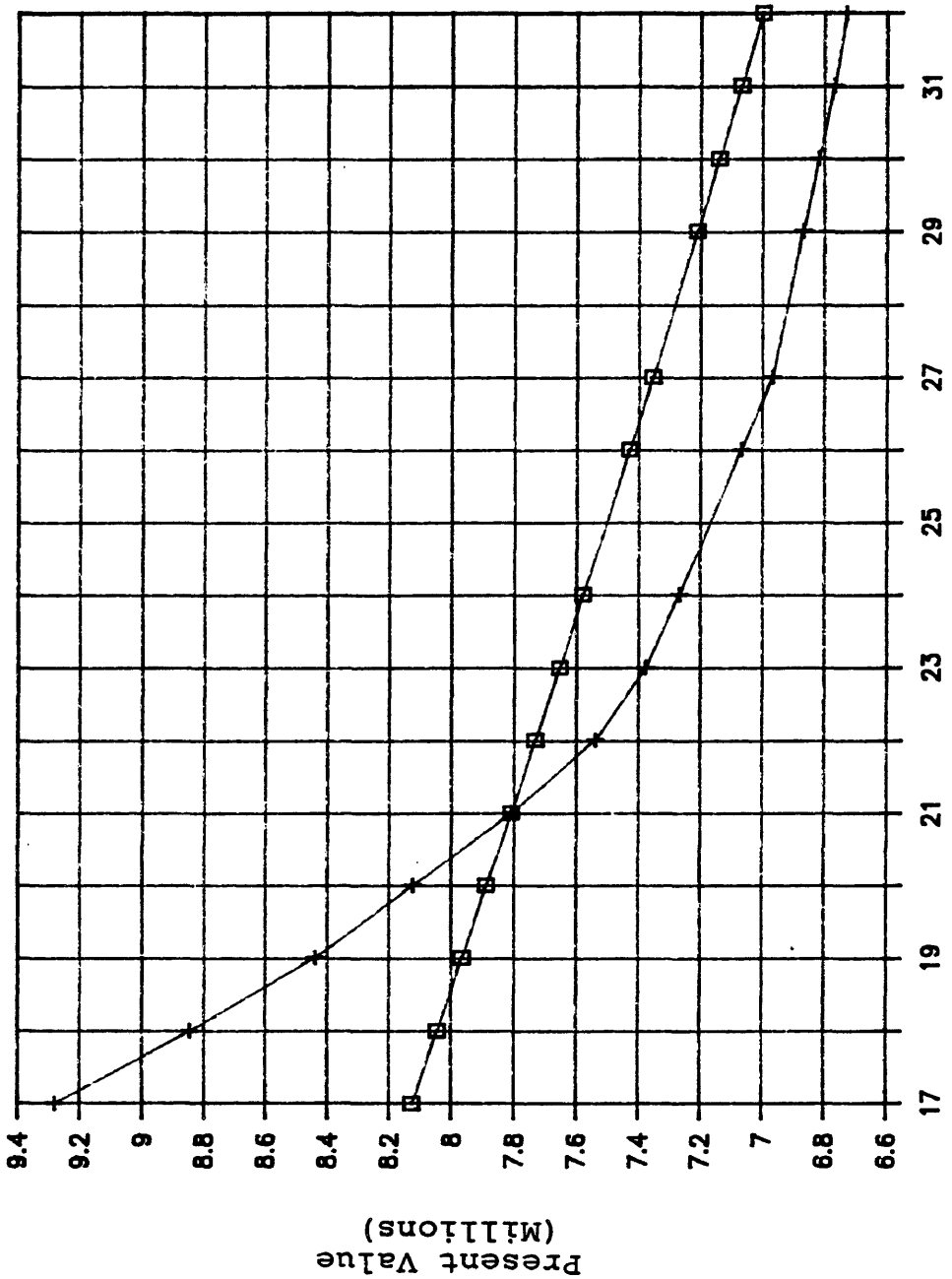


Figure 6.4 Present Values of Costs and Revenues vs. Mine Development Time Completion

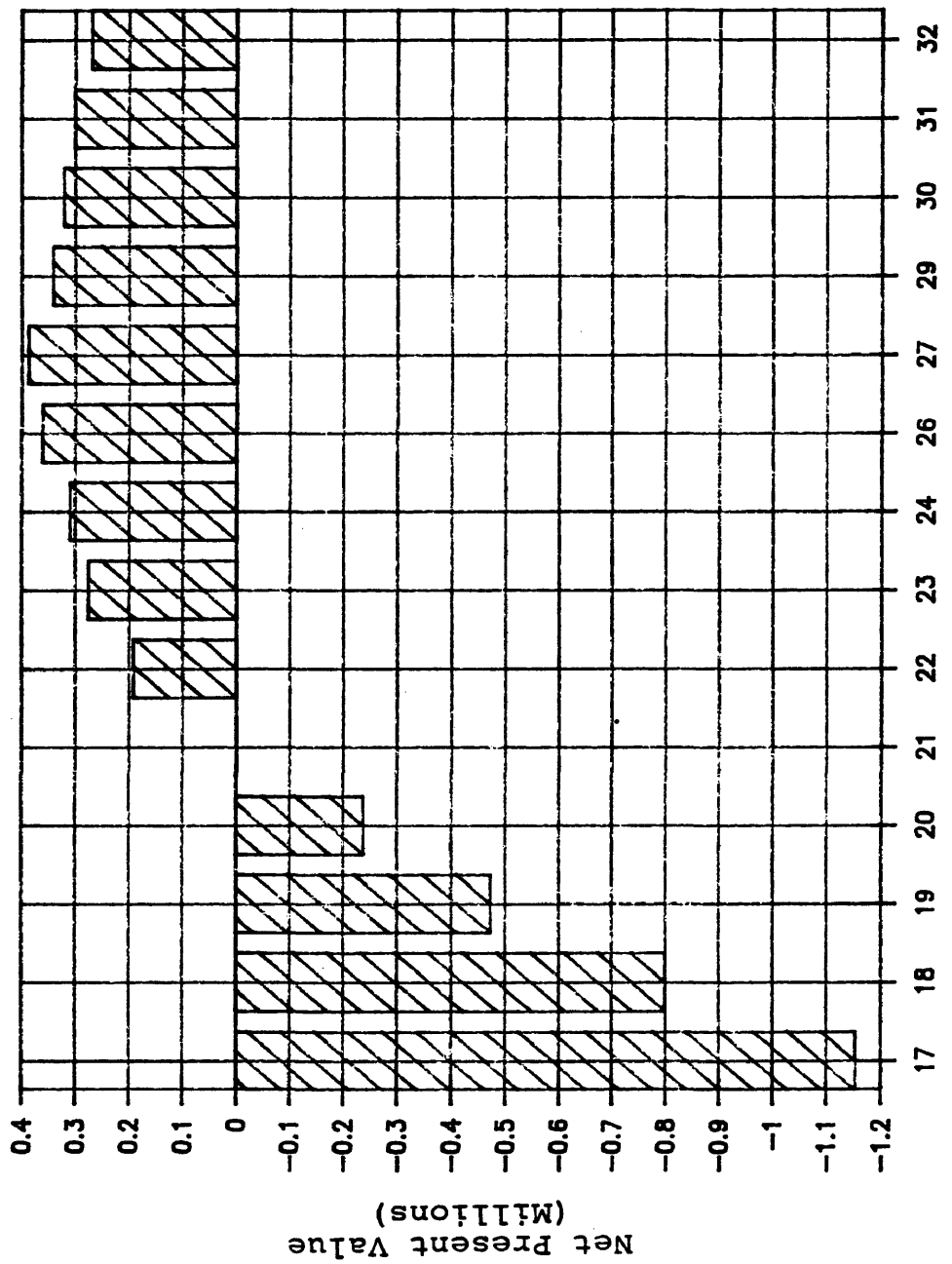


Figure 6.5 Net Present Value vs. Preproduction Time Completion

All preproduction time completions that achieve a higher NPV than the normal project time completion does are considered feasible solutions (see Table 6.10). Feasible solutions are those preproduction schedules for which an improvement on NPV can be achieved by shortening the preproduction time completion.

As a result, a feasible preproduction time completion schedule of 27 months will achieve the highest NPV ($387,000) for this hypothetical mine development project. Task time durations have to be scheduled as shown in Table 6.11.

6.4.2 Fixed Project Budget

A mine planner should be always prepared to answer appropriately questions such as: given a fixed amount of money to spend on the project, what is the shortest possible duration achievable?

The solution can be obtained by looking at Table 6.7 in the column Total Cost. For instance, given a fixed budget of $7,380,000 for the hypothetical mine development project, the shortest possible duration achievable is 23 months. Activity time durations should be scheduled as shown in Table 6.12.

When a fixed budget does not match a total cost on table 6.7, the shortest possible duration achievable is the

Table 6.11 Optimal Preproduction Schedule with the Highest Net Present Value

Task No.	Normal Task Duration (Mo.)	Time Units Shortened (Mo.)	Scheduled Task (Mo.)	Task Cost ($)
1	6	0	5	600,000
2	2	0	2	40,000
3	4	1	3	200,000
4	0	0	0	0
5	0	0	0	0
6	6	0	6	480,000
7	5	0	5	250,000
8	4	0	4	280,000
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	6	0	6	600,000
13	3	2	1	250,000
14	5	2	3	350,000
15	2	0	2	80,000
16	3	0	3	180,000
17	0	0	0	0
18	0	0	0	0
19	0	0	0	0
20	6	0	6	540,000
21	6	0	6	540,000
22	6	0	6	720,000
23	3	0	3	60,000
24	8	0	8	600,000
25	8	0	8	600,000
26	8	0	8	600,000
27	0	0	0	0
28	0	0	0	0
29	0	0	0	0
			Total:	$ 6,970,000

Table 6.12 Maximum Preproduction Schedule Compression for a Fixed Budget of $ 7,380,000

Task No.	Normal Task Duration (Mo.)	Time Units Shortened (Mo.)	Scheduled Task (Mo.)	Task Cost ($)
1	6	1	5	700,000
2	2	0	2	40,000
3	4	1	3	200,000
4	0	0	0	0
5	0	0	0	0
6	6	0	6	480,000
7	5	0	5	250,000
8	4	0	4	280,000
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	6	2	4	800,000
13	3	2	1	250,000
14	5	3	2	400,000
15	2	0	2	80,000
16	3	1	2	240,000
17	0	0	0	0
18	0	0	0	0
19	0	0	0	0
20	6	0	6	540,000
21	6	0	6	540,000
22	6	0	6	720,000
23	3	0	3	60,000
24	8	0	8	600,000
25	8	0	8	600,000
26	8	0	8	600,000
27	0	0	0	0
28	0	0	0	0
29	0	0	0	0
			Total:	$ 7,380,000

project time completion that corresponds to the lower total cost bound of the range on which the fixed budget falls.

6.4.3 Fixed Preproduction Time Completion

A mine planner can also face a question such as: how much time should be allowed each activity to complete the project at a certain target date and with minimum cost?

The solution can also be obtained just by looking at Table 6.7's column Project Time Duration. For instance, given a fixed project time completion of 20 months, the minimum cost to achieve the project on that target date is $8,125,000. Activity time durations should be scheduled as shown in Table 6.13.

The PVr will consider time 0 as the time when the production period starts (32 months from the beginning of the project).

Table 6.13 Optimal 20-Month Preproduction Schedule

Task No.	Normal Task Duration (Mo.)	Time Units Shortened (Mo.)	Scheduled Task (Mo.)	Task Cost ($)
1	6	2	4	860,000
2	2	0	2	40,000
3	4	1	3	200,000
4	0	0	0	0
5	0	0	0	0
6	6	0	6	480,000
7	5	0	5	250,000
8	4	0	4	280,000
9	0	0	0	0
10	0	0	0	0
11	0	0	0	0
12	6	3	3	980,000
13	3	2	1	250,000
14	5	3	2	400,000
15	2	0	2	80,000
16	3	1	2	240,000
17	0	0	0	0
18	0	0	0	0
19	0	0	0	0
20	6	2	4	720,000
21	6	0	6	540,000
22	6	0	6	720,000
23	3	0	3	60,000
24	8	1	7	675,000
25	8	1	7	675,000
26	8	1	7	675,000
27	0	0	0	0
28	0	0	0	0
29	0	0	0	0
			Total:	$ 8,125,000

Table 6.14 Least-Cost Duration vs. All-Time Crash Duration

Task No.	Normal Task Duration	Time Units: Least Cost	Time Units: All-Time Crash	Task Cost ($): Least Cost	Task Cost ($): All-Time Crash	Delta Cost
1	6	2	2	860,000	860,000	0
2	2	0	1	40,000	60,000	20,000
3	4	1	1	200,000	200,000	0
4	0	0	0	0	0	0
5	0	0	0	0	0	0
6	6	0	3	480,000	770,000	290,000
7	5	0	2	250,000	350,000	100,000
8	4	0	1	280,000	350,000	70,000
9	0	0	0	0	0	0
10	0	0	0	0	0	0
11	0	0	0	0	0	0
12	6	3	3	980,000	980,000	0
13	3	2	2	250,000	250,000	0
14	5	3	3	400,000	400,000	0
15	2	0	1	80,000	120,000	40,000
16	3	1	2	240,000	300,000	60,000
17	0	0	0	0	0	0
18	0	0	0	0	0	0
19	0	0	0	0	0	0
20	6	3	3	840,000	840,000	0
21	6	2	2	720,000	720,000	0
22	6	0	2	720,000	960,000	240,000
23	3	0	2	60,000	110,000	50,000
24	8	4	4	900,000	900,000	0
25	8	4	4	990,000	990,000	0
26	8	4	5	990,000	1,190,000	200,000
27	0	0	0	0	0	0
28	0	0	0	0	0	0
29	0	0	0	0	0	0
		Total :		9,280,000	10,350,000	1,070,000

CHAPTER 7

CONCLUSIONS AND RECOMMENDATIONS

7.1 Conclusions

1. The network-based planning approach (Critical Path Method) should be implemented to plan, schedule, and control the preproduction period of a new mining project. Gantt charts provide help only in the early phase of planning that period--the prefeasibility study, where not much detail is required. CPM, however, can go deeply into the details of how to perform every task in the project.

2. An efficient algorithm to obtain the least time-cost trade-off function is presented along with a computer program written in FORTRAN 77. There are several publications about algorithms to obtain an optimal time-cost trade-off function. The OKA algorithm was chosen because of personal preference of the author.

This computer program requires only one run to obtain the same output requiring 33 in Mathias and Redmon (1964). It has the additional advantage of cheaper computer costs.

3. The parameter optimal time for completion of the preproduction period of a new mining project have been solved. Three different optimal preproduction times can be

easily scheduled one with respect to each of the following objectives:

Highest net present value
Fixed preproduction period budget
Fixed preproduction time completion.

All three objectives involve the development of the least time-cost trade-off function. The first objective also introduces the development of positive cash flows or a revenue function.

The least time-cost trade-off function measures how sensitive the three objectives are to different preproduction schedules. A change in the preproduction period can also be easily measured against the objectives.

The amount of time alloted to the preproduction period can be considered in the project's financial feasibility analysis.

4. Personal computers (PC's) are well-proven tools to work with at any stage of a mining operation. PC's demonstrate again in the Least Cost Schedule (LCS) program their advantages over mainframes basically because the investment and running costs are much less.

5. The same algorithm solution presented in this study can be used for the expansion of an existing mine. The activities required to complete the expansion should be treated in the same way as the activities in the

preproduction period of a new mining project. The expected increase in revenues, from the expansion can also be calculated.

7.2 Recommendations for Further Research

1. The assumption of infinite resources was considered in this thesis. However, there are cases of mining operations with constraints on the availability of men and equipment that are not covered by the algorithm. There exist many heuristic solutions to this problem in the area of operations research. This can easily be factored into the LP form with a significant increment to set up all constraints.

2. A computerized graphical CPM would improve efficiency in planning, scheduling, and controlling any portion of the overall mining project. The use of plotters and large high-resolution screens is recommended.

3. A computerized interaction between cash flow analysis and CPM techniques is also needed. This would consist basically of sharing the same data file, so then any change in the schedule of an activity at any stage of the mining venture will be reflected inmediately in a cash flow analysis.

4. The LCS computer program could be improved and made user friendly by introducing menus.

5. The possibility of lenghtening the preproduction period could be considered.

6. How to measure the influence of scheduling the preproduction period on different decision criterias could be also taken into account.

REFERENCES

Agarwal, J.C., Brown, S.R. and Katrak, S.E. (1984), "Taking the Sting Out of Project Start-Up Problems", Engineering and Mining Journal, September.

Charles River Associates (1979), "Start-up of New Mine, Mill/Concentrator, and Processing Plants for Copper, Lead, Zinc, and Nickel: Survey and Analysis", in two volumes, Boston, Mass.

Cukierman, Alex and Shiffer, Zalman F. (1976), "Contracting for Optimal Delivery Time in Long-Term Projects", The Bell Journal of Economics, Vol. 7, No. 1, Spring 1976, pp. 132-149.

Daellenbach, H.G. and George, J.A. (1978) "Introduction to Operation Research Techniques", Ally and Bacon Inc., Boston.

Day, Jerome J. (1964), "A Generalization of Program Evaluation and Review Technique/Critical Path Method", A Thesis in Industrial Management, University of Pennsylvania.

Dessouky, Mohamed I. and Dunne, Edward J. "Cost-Duration Analysis with the Cut Network", University of Illinois at Urbana-Champaign.

Dowd, P. (1976) "Applications of Dynamic and Stochastic Programming to Optimize Cutoff Grades and Production Rates", Institution of Mining and Metallurgy Transactions, July.

Dowis, John E. (1982), "Detailed Cost Estimating for a Mining Venture Mineral Industry Costs", Northwest Mining Association, pp. 193-212

Elmaghraby, Salah E. (1977), "Activity Networks: Project Planning and Control by Network Models", John Wiley & Sons, N.Y., 443 p.

Ford, L.R. and Fulkerson, D.R. (1962), "Flows in Networks", Princeton University Press, Princenton, New Jersey.

Fulkerson, D.R. (1961) "A Network Flow Computation for Project Cost Curves", Management Science, Vol. 7, No. 2, January, pp. 167-179.

Gardner, John D. (1986) "Mine Evaluation-Design Optimization", 19th Application of Computers and Operation Research in the Mineral Industry, R.V. Ramani Edt., Society of Mining Engineers, Inc., Littleton, Colorado.

Gentry, D.W. and O'Neil, T.J. (1984) "Mine Investment Analysis", Society of Mining Engineers, New York, New York.

Gentry, Donald W. and Hrebar, Matthew J. (1976) "Procedure for Determining Economics of Small Underground Mines", Mineral Industries Bulletin, Volume 19, Number 1, January, Colorado School of Mines.

Hoffman, C.W. and Franklin, D.R. (1982) "Initial operations --Insuring the Return on Your Construction Investment", Mining Engineering, December.

Johnson, T.B. and Barnes, R.J. (1984) "Mine Systems Engineering", draft copy.

Kelley, J.E.,Jr. (1961) "Critical Path Planning and Scheduling: Mathematical Basis", Operations Research, Vol. 9, No. 3, pp. 296-320.

Mathias, Adrian J. (1966), "A Mine Production-Scheduling Model and Critical Path Analysis of Mine Development Work For Long-Range Mine Planning", Report of investigations 6937, Bureau of Mines.

Mathias, Adrian J. and Redmon, Donald E. (1965), "Critical Path Planning and Scheduling Applied to Mining Operations". Report of investigations 6739, Bureau of Mines.

Moder, Joseph J., Phillips, Cecil R., and Davis, Edward W. (1983), "Project Management with CPM, PERT and Precedence Diagraming", Third Edition, Van Nostrand Reinhold Company Inc., N.Y., 389 p.

Naftal, Ronald (1964), "Cerro's Fresh Approach to Critical Path Planning, Metal Mining and Processing", February.

O'Hara T.A. (1982), "Mine Evaluation, Mineral Industry Costs, Northwest Mining Association", pp. 89-99

Paulsen, Kenneth R. (1982), "Environmental and Regulatory Costs of Mining Projects", Mineral Industry Costs, Northwest Mining Association, pp. 27-32

Phillips, Don T. and Garcia-Diaz, Alberto (1981), "Fundamentals of Network Analysis", Prentice-Hall, Inc. Englewood Cliffs, N.J., 474 p.

Rudawsky, Oded (1977) "Economic feasibility Studies in Mineral and Energy Industries", Mineral Induntries Bulletin, , Volume 20, Numbers 3 and 4, Colorado School of Mines.

Siemens, N. (1971) "A Simple CPM Time-Cost Trade-Off Algorithm", Management Science, Vol. 17, No. 6, February, pp. B-354-363.

Tinsley, Richard C. (1982), "Mine Finance Mineral Industry Costs", Northwest Mining Association, pp. 213-220

Wells, Howard M. (1978), "Optimization of Mining Engineering Design in Mineral Valuation", Mining Engineering, December, pp. 1676-1684.

APPENDIX A

The Network-flow Algorithm Hand Computational Procedure

PROJECT COST CURVE ALGORITHM

1. Determine Critical Activities

   Solve time only problem using longest path algorithm to find TE(i) = TF(i) critical path and project duration.

2. Labeling for Minimum-Cost Activities

   Using only critical arcs, attempt to find a flow augmenting path from Start to End with labeling algorithm, arc capacity before shortening is equal to C(i,j).

   a).- if path is found, "breakthrough" go to 3.

   b).- if no path is found, "non-breakthrough" go to 4.

3. Flow Change Procedure

   Flow change by amount t(End) (Flow into Sink Node).

if t(End) = Infinite	STOP! Project cannot be shortened further.
if t(End) = Delta	GO! Increase flow along determined path and go to Step 2.

4. Node Time-change Procedure

   New Event Times as follows:

   TE(i) = TE(i) , i E X (labeled nodes)

   TE(i) = TE(i) - Delta , i E $\overline{X}$ (unlabeled nodes)

Where Delta is determined as follows:

Forward arcs: $i \in X$ and $j \in \overline{X}$.

$$\text{Delta}(1) = \text{Min} [TE(j) - TE(i) - K(i,j)] \quad i,j \text{ / Non-CP}$$

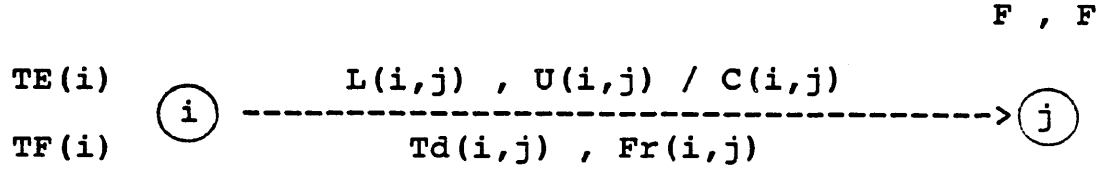
$$\text{Delta}(2) = \text{Min} [TE(j) - TE(i) - L(i,j)] \quad i,j \text{ / CP}$$

Reverse arcs: $i \in \overline{X}$ and $j \in X$.

$$\text{Delta}(3) = \text{Min} [K(i,j) + TE(i) - TE(j)] \quad i,j \text{ / CP}$$

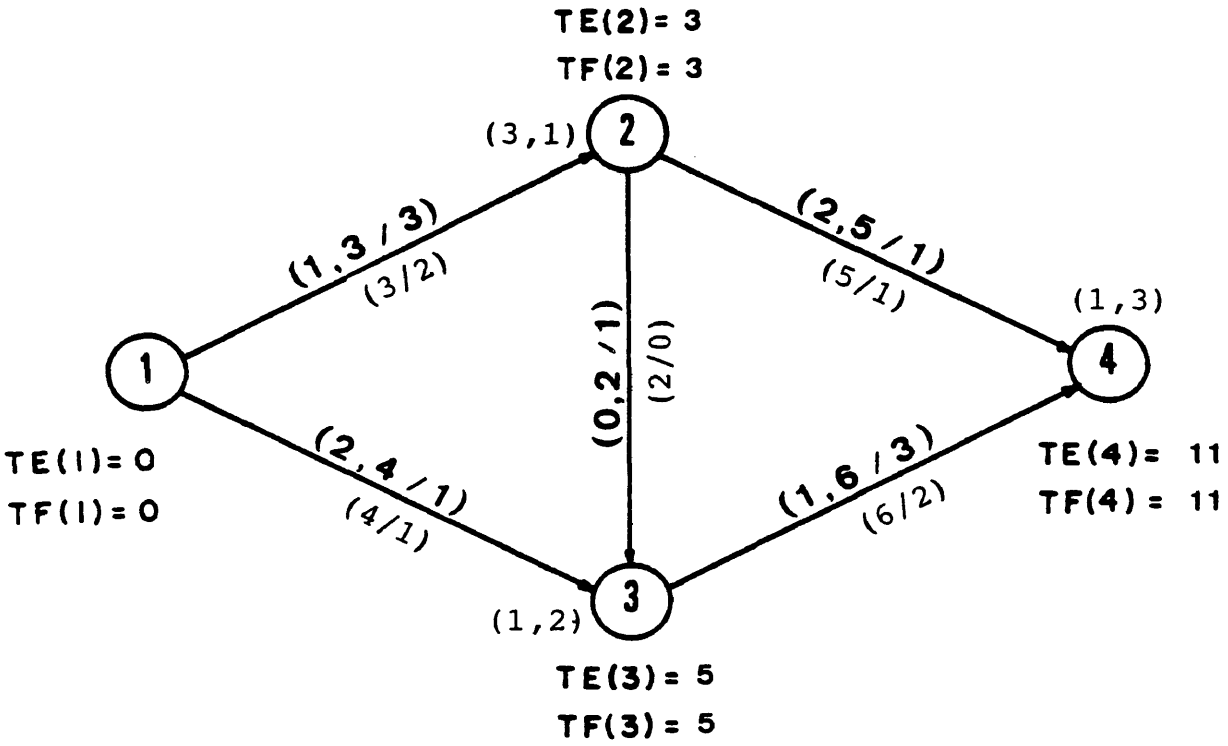
$$\text{Delta} = \text{Min} \{ \text{Delta}(1), \text{Delta}(2), \text{Delta}(3) \}$$

Calc New { TE(i) } and go to Step 2.



L(i,j)	...	Lower Cost Bound.
U(i,j)	...	Upper Cost Bound.
C(i,j)	...	Cost Slope.
Td(i,j)	...	Time Duration.
Fr(i,j)	...	Flow Remaining.
TE(i)	...	Earliest Time Completion.
TF(i)	...	Latest Time Completion.
F	...	Flow.
Ff(i)	...	Flow from Event i.

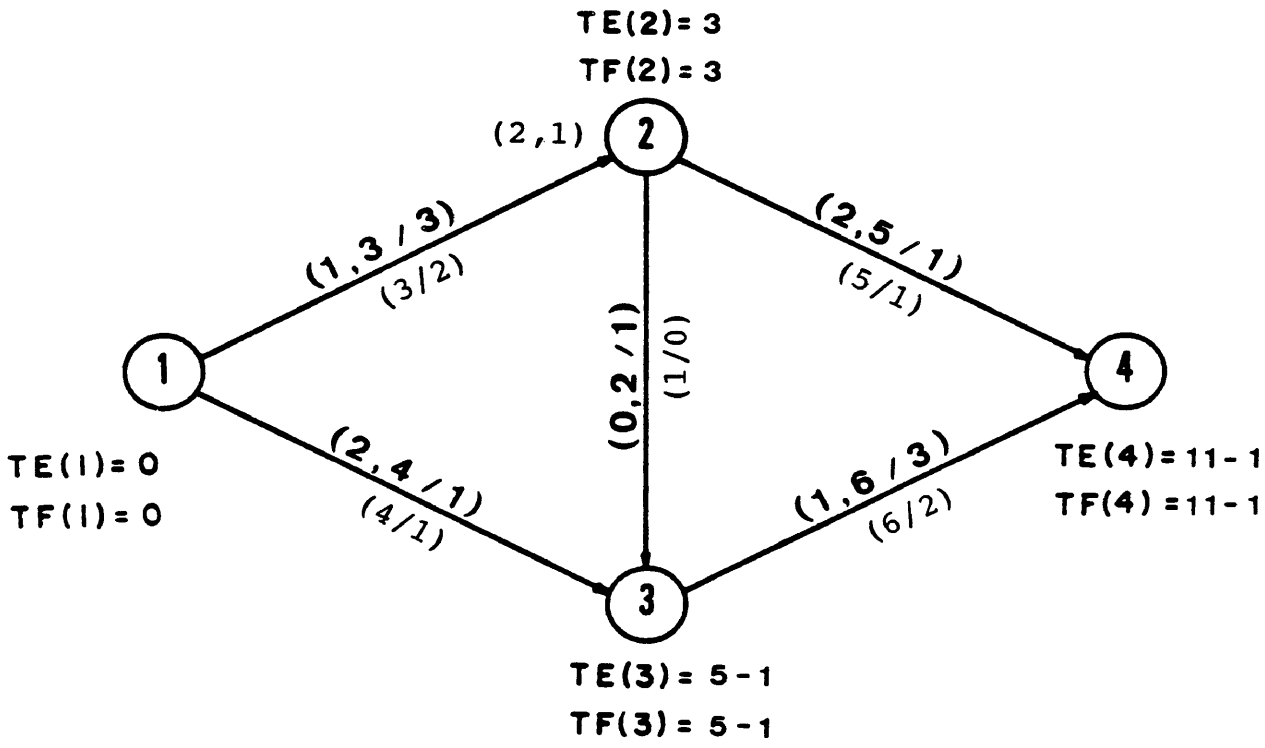
Figure A.1 Time-Cost Trade-Off Activity (i,j) Nomenclature



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	3	3	Y	3	1
3	5	5	Y	1	2
4	11	11	Y	1	3

Arc No.	CP	Time Duration	Flow Remain
1	Y	3	2
2	N	4	1
3	Y	2	0
4	N	5	1
5	Y	6	2

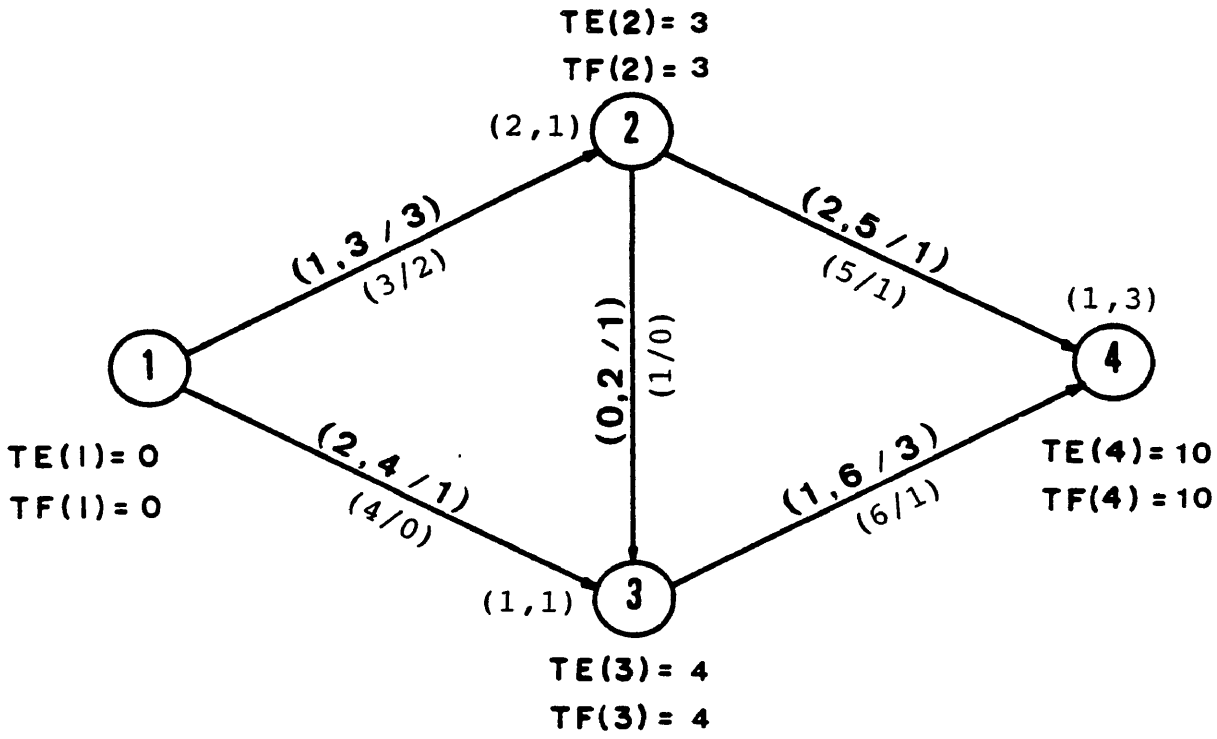
Figure A.2. Iteration No. 1



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	3	3	Y	2	1
3	(5-1)	(5-1)	N	1000	0
4	(11-1)	(11-1)	N	1000	0

Arc No.	CP	Time Duration	Flow Remain	Deltas D1	D2	D3
1	Y	3	2	1000	1000	1000
2	N	4	1	1	1000	1000
3	Y	1	0	1000	2	1000
4	N	5	1	3	1000	1000
5	Y	6	2	1000	1000	1000

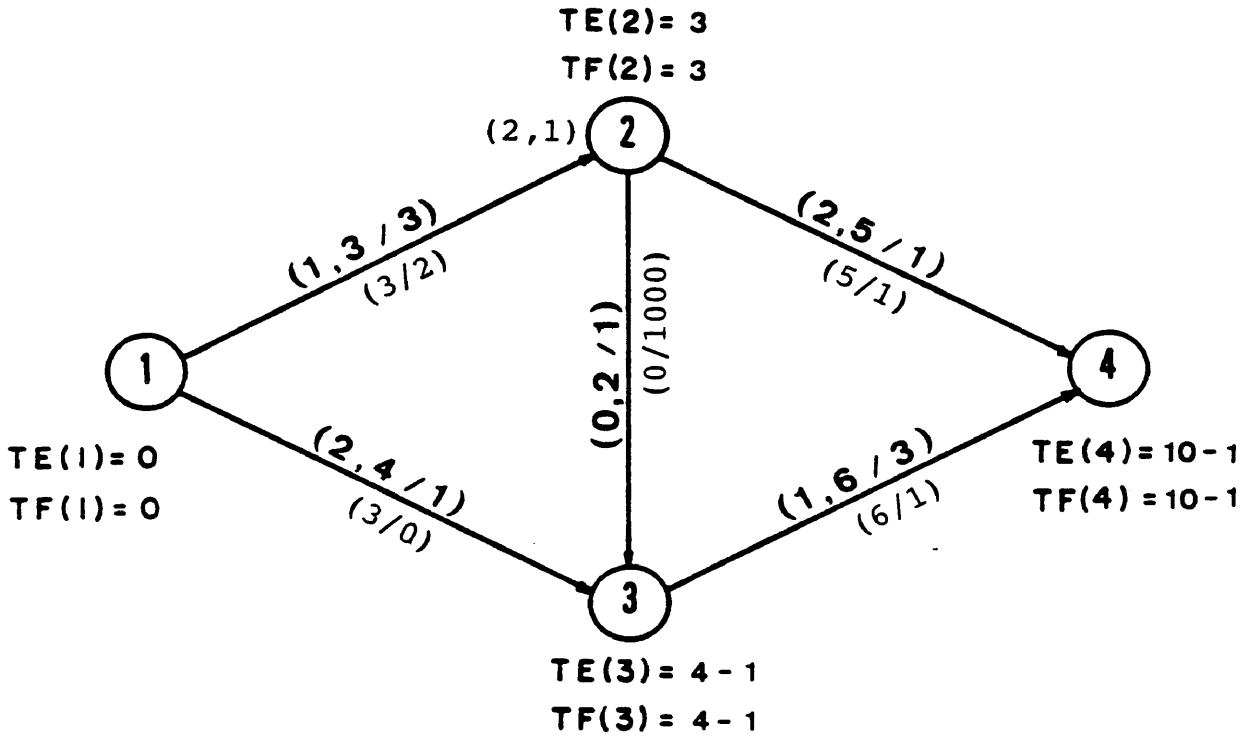
Figure A.3. Iteration No. 2



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	3	3	Y	2	1
3	4	4	Y	1	1
4	10	10	Y	1	3

Arc No.	CP	Time Duration	Flow Remain
1	Y	3	2
2	Y	4	0
3	Y	1	0
4	N	5	1
5	Y	6	1

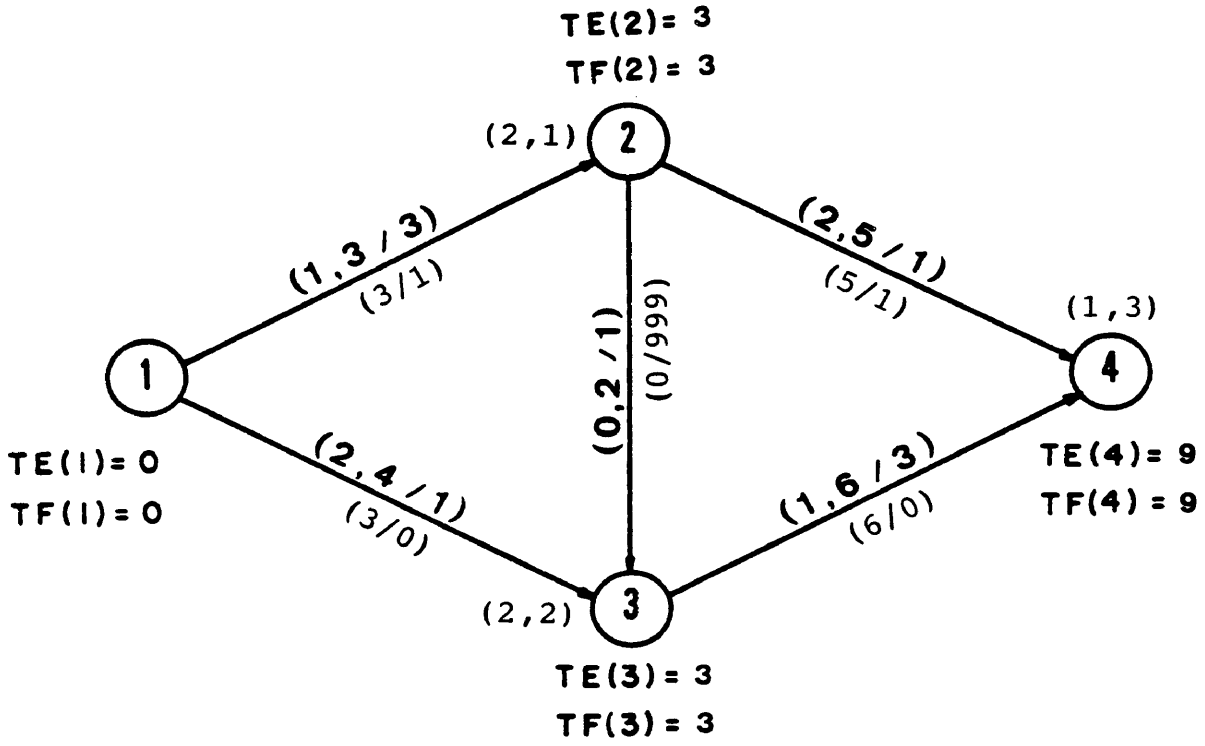
Figure A.4. Iteration No. 3



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	3	3	Y	2	1
3	(4-1)	(4-1)	N	1000	0
4	(10-1)	(10-1)	N	1000	0

Arc No.	CP	Time Duration	Flow Remain	Deltas D1	D2	D3
1	Y	3	2	1000	1000	1000
2	Y	3	0	1000	2	1000
3	Y	0	1000	1000	1	1000
4	N	5	1	2	1000	1000
5	Y	6	1	1000	1000	1000

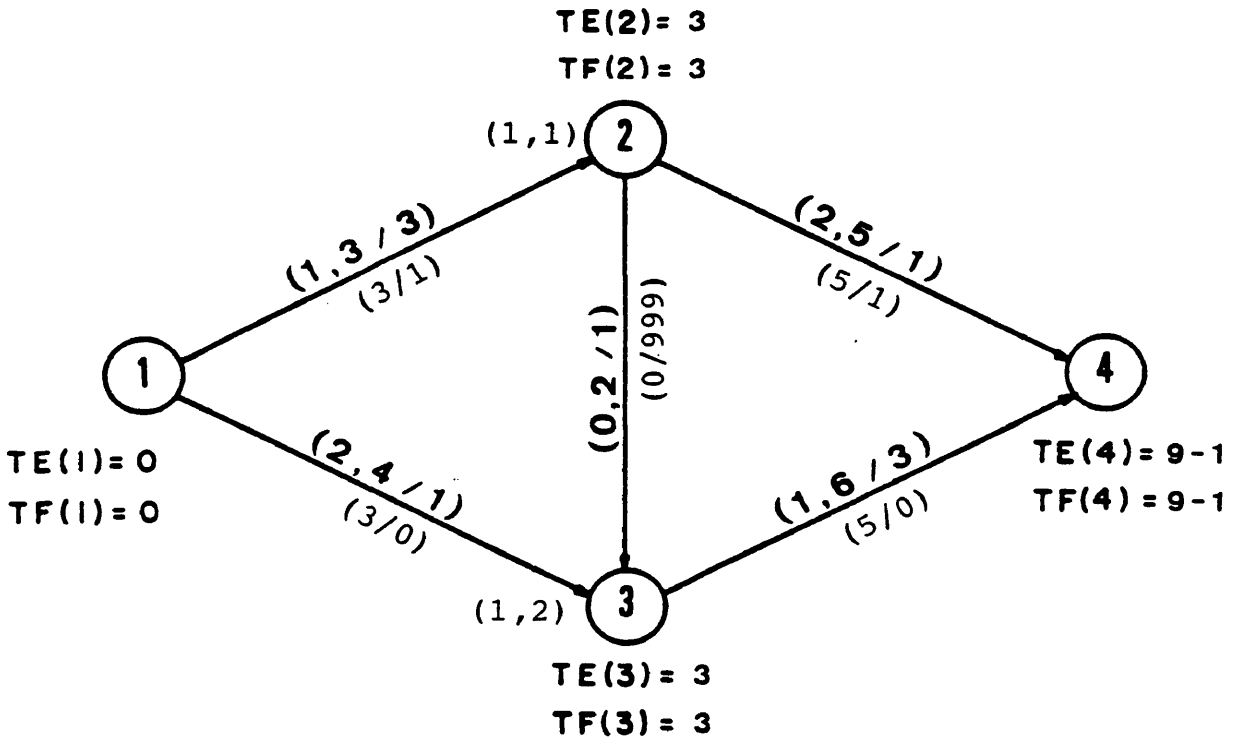
Figure A.5. Iteration No. 4



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	3	3	Y	2	1
3	3	3	Y	2	2
4	9	9	Y	1	3

Arc No.	CP	Time Duration	Flow Remain
1	Y	3	1
2	Y	3	0
3	Y	0	999
4	N	5	1
5	Y	6	0

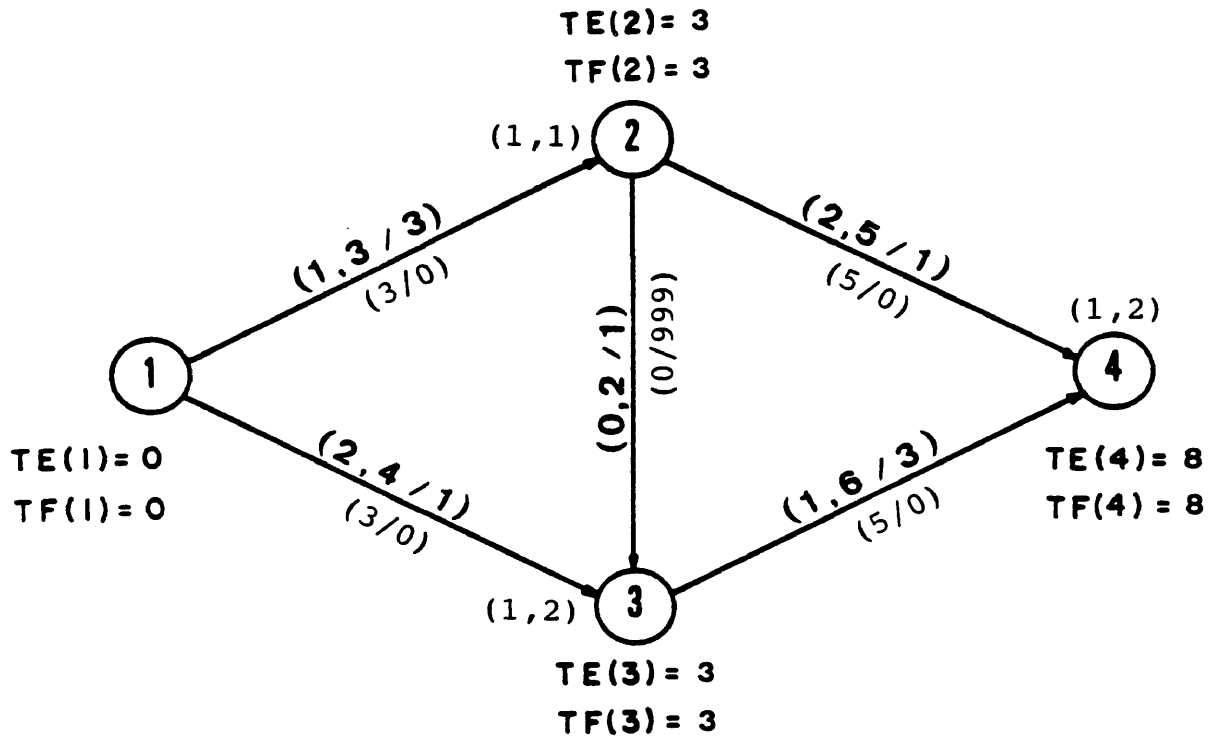
Figure A.6. Iteration No. 5



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	3	3	Y	1	1
3	3	3	Y	1	2
4	(9-1)	(9-1)	N	1000	0

Arc No.	CP	Time Duration	Flow Remain	Deltas D1	D2	D3
1	Y	3	1	1000	1000	1000
2	Y	3	0	1000	1000	1000
3	Y	0	999	1000	1000	1000
4	N	5	1	1	1000	1000
5	Y	5	0	1000	5	1000

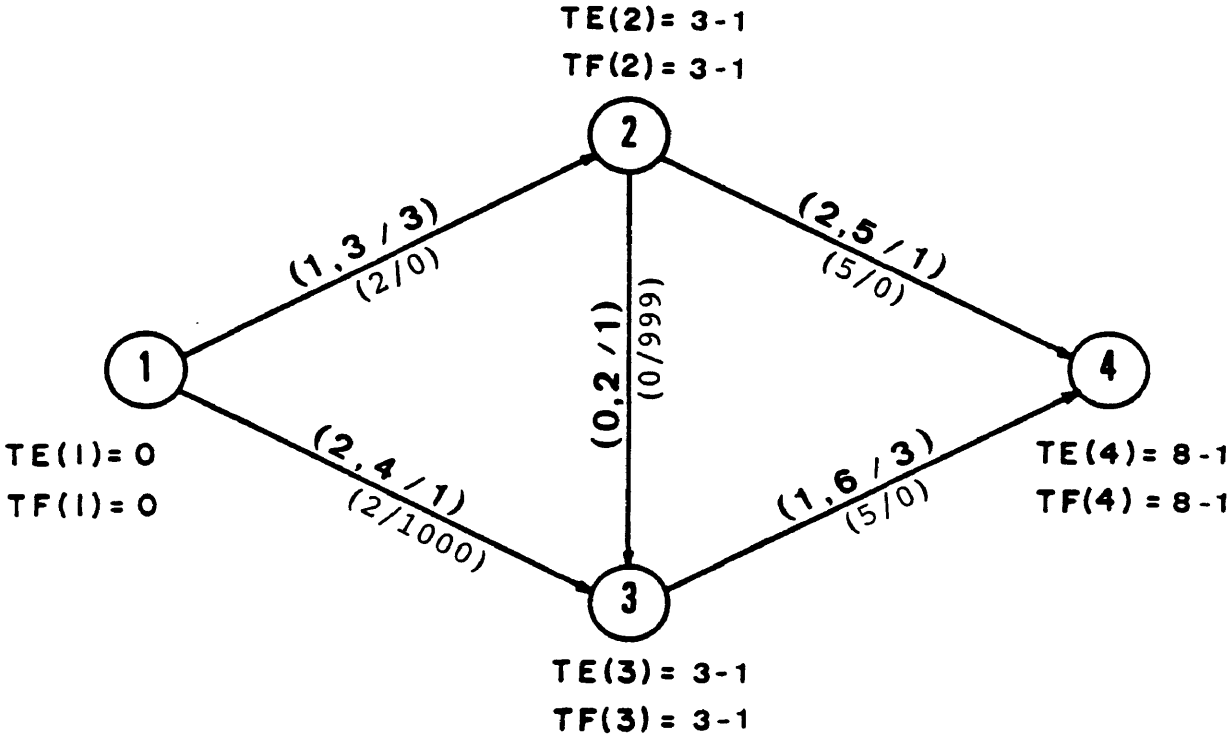
Figure A.7. Iteration No. 6



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	3	3	Y	1	1
3	3	3	Y	1	2
4	8	8	Y	1	2

Arc No.	CP	Time Duration	Flow Remain
1	Y	3	0
2	Y	3	0
3	Y	0	999
4	Y	5	0
5	Y	5	0

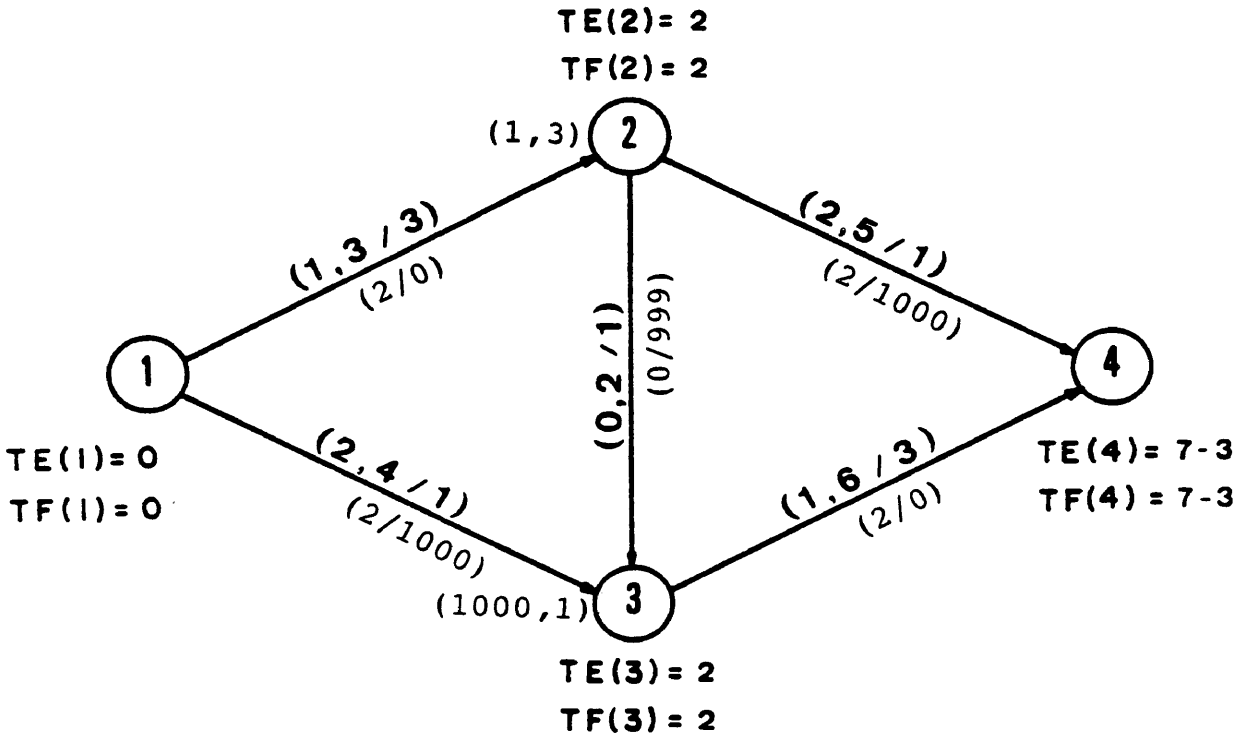
Figure A.8. Iteration No. 7



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	(3-1)	(3-1)	N	1000	0
3	(3-1)	(3-1)	N	1000	0
4	(8-1)	(8-1)	N	1000	0

Arc No.	CP	Time Duration	Flow Remain	Deltas D1	D2	D3
1	Y	2	0	1000	2	1000
2	Y	2	1000	1000	1	1000
3	Y	0	999	1000	1000	1000
4	Y	5	0	1000	1000	1000
5	Y	5	0	1000	1000	1000

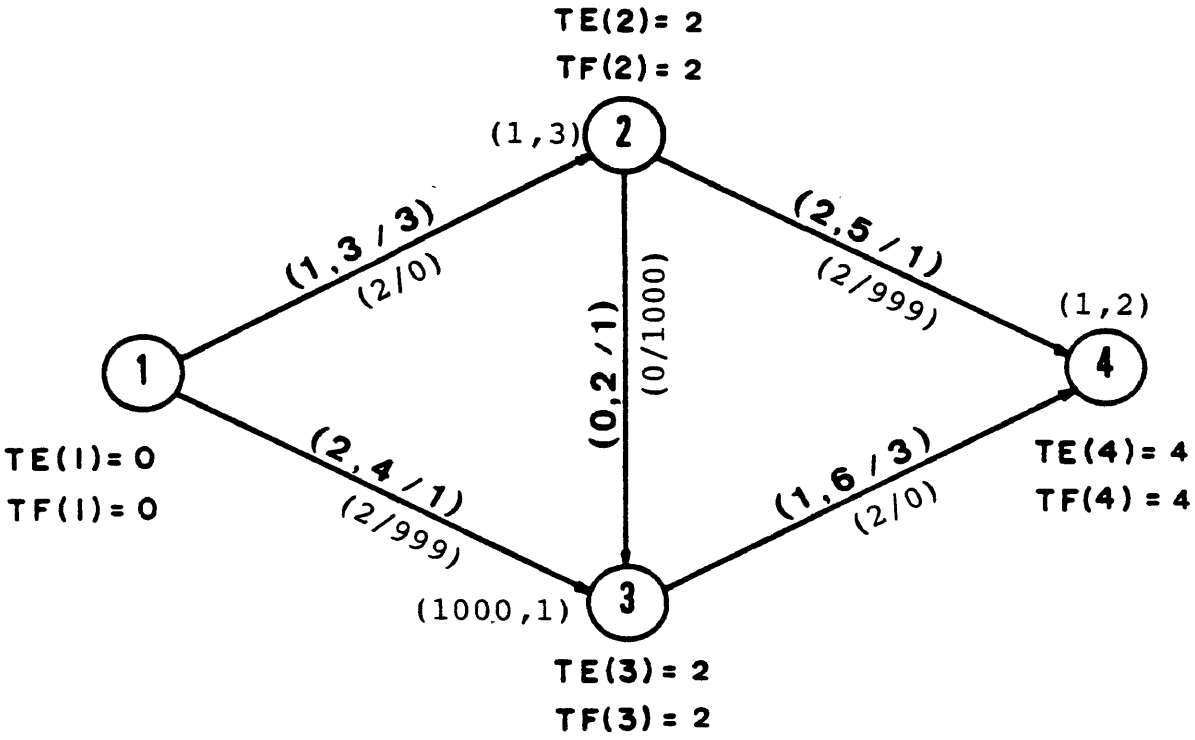
Figure A.9. Iteration No. 8



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	2	2	Y	1	3
3	2	2	Y	1000	1
4	(7-3)	(7-3)	N	1000	0

Arc No.	CP	Time Duration	Flow Remain	Deltas D1	D2	D3
1	Y	2	0	1000	1000	1000
2	Y	2	1000	1000	1000	1000
3	Y	0	999	1000	1000	1000
4	Y	2	1000	1000	3	1000
5	Y	2	0	1000	4	1000

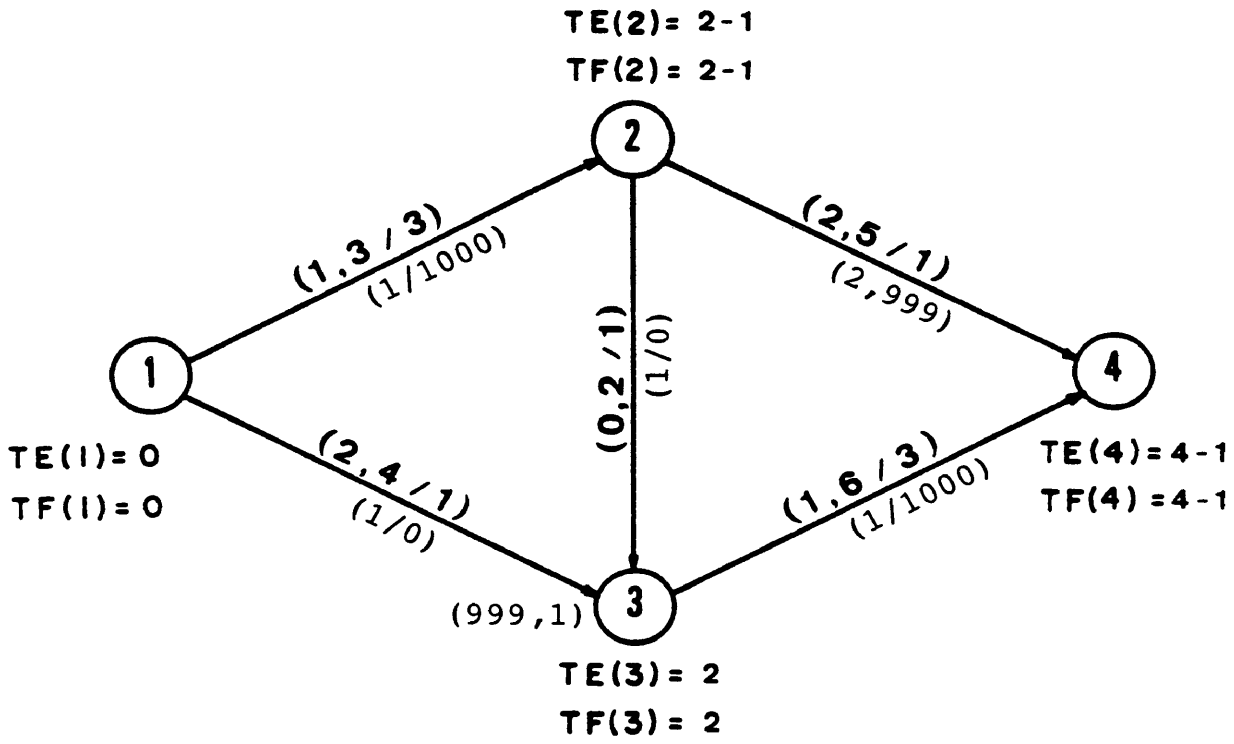
Figure A.10. Iteration No. 9



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	2	2	Y	1	3
3	2	2	Y	1000	1
4	4	4	Y	1	2

Arc No.	CP	Time Duration	Flow Remain
1	Y	2	0
2	Y	2	999
3	Y	0	1000
4	Y	2	999
5	Y	2	0

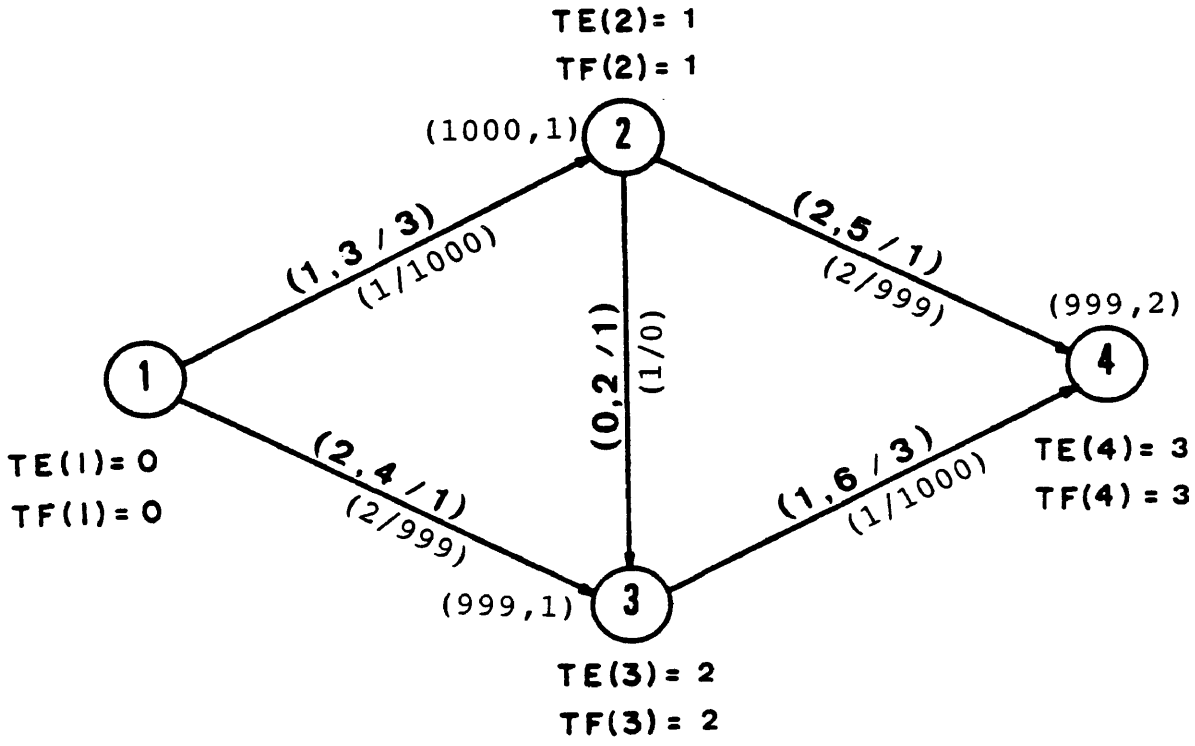
Figure A.11. Iteration No. 10



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	(2-1)	(2-1)	N	1000	0
3	2	2	Y	999	1
4	(4-1)	(4-1)	N	1000	0

Arc No.	CP	Time Duration	Flow Remain	Deltas D1	D2	D3
1	Y	1	1000	1000	1	1000
2	Y	2	999	1000	1000	1000
3	Y	1	0	1000	1000	2
4	Y	2	999	1000	1000	1000
5	Y	1	1000	1000	1	1000

Figure A.12. Iteration No. 11



Node i	TE(i)	TF(i)	Label	Flow	Flow from Node
1	0	0	Y	1000	0
2	1	1	Y	1000	1
3	2	2	Y	999	1
4	3	3	Y	999	2

Arc No.	CP	Time Duration	Flow Remain
1	Y	1	1000
2	Y	2	999
3	Y	1	0
4	Y	2	999
5	Y	1	1000

Figure A.12. Iteration No. 12

No.	Time Duration	Delta Cost	Slope Cost	Delta Time	Tasks shortened			Tasks lengthened	
1	11	0	0	0	-	-	-	-	-
2	10	1	1	1	3	-	-	-	-
3	9	3	2	1	2	3	-	-	-
4	8	6	3	1	5	-	-	-	-
5	7	10	4	1	1	2	-	-	-
6	4	22	4	3	4	5	-	-	-
7	3	27	5	1	1	5	-	3	-

Table A.1. The Optimal cost-duration curve

APPENDIX B

Least-cost Program Code in FORTRAN 77

```
C     ******************************************************
C                THE LEAST COST CURVE PROJECT
C     ******************************************************

C                          GENERAL DATA
C                          ============

C     x           -  arc(i,j)
C     I(x)        -  node i that defines begin of arc(i,j)
C     J(x)        -  node j that defines end of arc(i,j)
C     NCOST(x)    -  Normal cost of arc(i,j)
C     NROW(x)     -  How many sub-ranges in arc(i,j) (Max = 3)
C     LO(n,x)     -  lower bound sub-range n of arc(i,j)
C     HI(n,x)     -  upper bound sub-range n of arc(i,j)
C     COST(n,x)   -  Cost of flow in arc(i,j) sub-range n
C     NODES       -  # of nodes in network
C     ARCS        -  # of arcs in network


C                     CALCULATION VARIABLES
C                     =====================

C     TE(x)       -  Earliest completion of task (i,j)
C     TF(x)       -  Latest completion of task (i,j)
C     DMHI(x)     -  Current duration time  of task (i,j)

C     MKCP(x)     -  0   task (i,j) isn't in the critical path
C                    1   task (i,j) is in the critical path

C     LBL(x)      -  0   node j of task (i,j) can't be labeled
C                    1   node j of task (i,j) can be labeled

C     DMCOST(x)   -  Current total cost of task (i,j)
C     FROM(x)     -  Node i from which node j can be labeled
C     FLNODE(x)   -  Maximum flow through arc (i,j)
C     BARC(x)     -  Node j from which node i can be labeled

C     FORWD(x)    -  0   arc (i,j) isn't a forward arc
C                    1   arc (i,j) is a forward arc

C     REVER(x)    -  0   arc (i,j) isn't a reverse arc
C                    1   arc (i,j) is a reverse arc

C     DEL1(x)     -  Delta 1 of arc (i,j)
C     DEL2(x)     -  Delta 2 of arc (i,j)
C     DEL3(x)     -  Delta 3 of arc (i,j)
```

```
C                    OUTPUT at iteration k
C                    =====================

C      TOTIME(k)     -  Total duration time of project
C      TOCOST(k)     -  Total direct cost of project
C      SLCOST(k)     -  Total slope cost
C         CUT(k)     -  Units of times shortened
C      ACTCUT(k,10)  -  Activities to be shortened  (Max = 10)
C      ACTLEN(k,10)  -  Activities to be lenghtened (Max = 10)

C      ******************************************************
C                  DECLARATION OF VARIABLES
C      ******************************************************

       IMPLICIT INTEGER(A-Z)
       COMMON /DAT1/ NODES,ARCS,I(100),J(100),NCOST(100)
       COMMON /DAT2/ LO(3,100),HI(3,100),COST(3,100),NROW(100)
       COMMON /CAL1/ TE(100),TF(100),DMHI(100),MKCP(100)
       COMMON /CAL2/ DMCOST(100),FROM(100),FLNODE(100)
       COMMON /CAL3/ LBL(100),FORWD(100),REVER(100),BARC(100)
       COMMON /CAL4/ DEL1(100),DEL2(100),DEL3(100)
       COMMON /OUT1/ TOTIME(50),TOCOST(50),SLCOST(50),CUT(50)
       COMMON /OUT2/ ACTCUT(50,10),ACTLEN(50,10),KNUM

C      ******************************************************
C                     OPEN and PRINT DATA FILE
C      ******************************************************

       OPEN(3,FILE='DATAOKA')
       READ(3,*)     NODES, ARCS
       WRITE(*,105)  NODES, ARCS
       WRITE(*,110)

       CALL INITIAL

       DO 5 M=1,ARCS
         READ(3,*) I(M),J(M),N,
      +            (LO(K,M),HI(K,M),COST(K,M),K=1,N)
         WRITE(*,115) M,I(M),J(M),
      +            (LO(K,M),HI(K,M),COST(K,M),K=1,3)
     5 CONTINUE

   105 FORMAT(/,5X,'NODES:',I5,5X,'ARCS:',I5)
   110 FORMAT(//,4X,'M     I     J',6X,'LO1   HI1   CT1',5X,
      +'LO2   HI2   CT2',5X,'LO3   HI3   CT3')
   115 FORMAT(/,3(1X,I4),3(4X,3(1X,I4)))
```

```
C     ******************************************************
C                    INITIAL TOTAL PROJECT COST
C     ******************************************************

      DO 15 M=1,ARCS
        TOCOST(1)=TOCOST(1)+NCOST(M)
        DMCOST(M)=COST(1,M)
        DMHI(M)=HI(1,M)
   15 CONTINUE

C     ******************************************************
C          INITIAL DUMMY VARIABLES WITH INFINITE VALUE
C     ******************************************************

      DO 20 A=1,ARCS
        IF((LO(1,A).EQ.0).AND.(HI(1,A).EQ.0) THEN
          IF(DMCOST(A).EQ.0) THEN
            DMCOST(A)=1000
          ENDIF
        ENDIF
   20 CONTINUE

C     ******************************************************
C                      ALGORITHM - APPENDIX B
C     ******************************************************

      SOURCE=1
      SINK=NODES
      KNUM=1
      DO 25 COUNT=1,100
        CALL STEP1
        CALL STEP2
        IF(LBL(SINK).EQ.0)THEN
          CALL STEP4
          CALL PRINT
          GOTO 50
        ENDIF
        IF(FLNODE(SINK).GE.500)THEN
          CALL PRINT
          CALL ANSWER
          STOP
        ELSE
          CALL STEP3
          CALL PRINT
        ENDIF
   25 CONTINUE
      STOP
      END
```

```
C     ********************************************************
C                DETERMINE CRITICAL ACTIVITIES
C
C                             STEP 1
C                             ------
C     ********************************************************

      SUBROUTINE STEP1
      IMPLICIT INTEGER(A-Z)
      COMMON /DAT1/ NODES,ARCS,I(100),J(100),NCOST(100)
      COMMON /DAT2/ LO(3,100),HI(3,100),COST(3,100),NROW(100)
      COMMON /CAL1/ TE(100),TF(100),DMHI(100),MKCP(100)
      COMMON /OUT1/ TOTIME(50),TOCOST(50),SLCOST(50),CUT(50)
      COMMON /OUT2/ ACTCUT(50,10),ACTLEN(50,10),KNUM
      DATA  ZERO / 0 /

C     ********************************************************
C                       INITIALIZE VARIABLES
C     ********************************************************

      CALL BLANK1(MKCP,ARCS,ZERO)

C     ********************************************************
C           DEFINE TE(J) - EARLIEST TIME COMPLETION
C     ********************************************************

      DO 15 A=1,ARCS
        IA=I(A)
        JA=J(A)
        DURAT=TE(IA)+DMHI(A)
        IF(DURAT.GT.TE(JA)) THEN
          TE(JA)=DURAT
        ENDIF
15    CONTINUE

C     ********************************************************
C                  TE(J) FOR THE ENTIRE PROJECT
C     ********************************************************

      IF(KNUM.EQ.1)THEN
        TOTIME(1)=TE(NODES)
      ENDIF
```

```
C     ********************************************************
C              DEFINE TF(I) - LATEST TIME COMPLETION
C     ********************************************************

      TF(NODES)=TE(NODES)
      DO 25 A=ARCS,1,-1
        IA=I(A)
        JA=J(A)
        DURAT=TF(JA)-DMHI(A)
        IF(DURAT.LE.TF(IA)) THEN
          TF(IA)=DURAT
        ENDIF
25    CONTINUE

C     ********************************************************
C                      CRITICAL ACTIVITIES
C     ********************************************************

      DO 17 A=1,ARCS
        IA=I(A)
        JA=J(A)
        IF((TE(IA).EQ.TF(IA)).AND.(TE(JA).EQ.TF(JA)))THEN
          DURAT=TE(IA)+DMHI(A)
          IF(DURAT.EQ.TF(JA))THEN
            MKCP(A)=1
          ELSE
            MKCP(A)=0
          ENDIF
        ENDIF
17    CONTINUE

      RETURN
      END
```

```
C     ********************************************************
C             LABELING FOR MINIMUM-COST ACTIVITIES
C
C                           STEP 2
C                           ------
C     ********************************************************

      SUBROUTINE STEP2(BREAK)
      IMPLICIT INTEGER(A-Z)
      COMMON /DAT1/ NODES,ARCS,I(100),J(100),NCOST(100)
      COMMON /DAT2/ LO(3,100),HI(3,100),COST(3,100),NROW(100)
      COMMON /CAL1/ TE(100),TF(100),DMHI(100),MKCP(100)
      COMMON /CAL2/ DMCOST(100),FROM(100),FLNODE(100)
      COMMON /CAL3/ LBL(100),FORWD(100),REVER(100),BARC(100)
      DATA ZERO,INF / 0 , 10000 /

C     ********************************************************
C                       INITIALIZE VARIABLES
C     ********************************************************

      CALL BLANK1(LBL,NODES,ZERO)
      CALL BLANK1(FROM,NODES,ZERO)
      CALL BLANK1(BARC,ARCS,ZERO)
      CALL BLANK1(FLNODE,NODES,INF)

C     ********************************************************
C               NODE 1 IS ALWAYS LABELED FIRST
C     ********************************************************

      LBL(1)=1

C     ********************************************************
C           TEST OF LABELING THROUGH CRITICAL ARCS
C     ********************************************************

      DO 15 COUNT=1,NODES
        DO 10 A=1,ARCS
C         ****************************************************
C                   MKCP(A)=0  --->  NO CRITICAL ARC
C         ****************************************************
          IF(MKCP(A).EQ.0)THEN
            GOTO 10
          ENDIF
          IA=I(A)
          JA=J(A)
```

```
C          ************************************************
C                      BEGIN OF ARC A IS LABELED
C                           FORWARD LABELING
C          ************************************************
           IF((LBL(IA).EQ.1).AND.(DMCOST(A).GE.1))THEN
             LBL(JA)=1
             IF(BARC(A).EQ.0)THEN
               IF((DMCOST(A)  .LT.FLNODE(JA)).OR.
     +            (FLNODE(IA).LT.FLNODE(JA)))  THEN
C              ********************************************
C                      MINIMUM FLOW IN CRITICAL ARCS
C              ********************************************
                 FLNODE(JA)=MIN0(DMCOST(A),FLNODE(IA))
                 FROM(JA)=IA
               ENDIF
             ENDIF
           ENDIF
C          ************************************************
C                      END OF ARC A IS LABELED
C                         BACKWARD LABELING
C          ************************************************
           IF((LBL(JA).EQ.1).AND.(LBL(IA).EQ.0))THEN
             IF((DMCOST(A).LE.999).AND.(DMCOST(A).GE.500))THEN
               LBL(IA)=1
               IF(BARC(A).EQ.0)THEN
                 FLBACK=1000-DMCOST(A)
                 FLNODE(IA)=MIN0(FLBACK,FLNODE(JA))
                 FROM(IA)=JA
                 BARC(A)=1
               ENDIF
             ENDIF
           ENDIF
    10   CONTINUE
    15 CONTINUE
       RETURN
       END
```

```
C     ********************************************************
C                      FLOW CHANGE PROCEDURE
C
C                              STEP 3
C                              ------
C     ********************************************************

      SUBROUTINE STEP3
      IMPLICIT INTEGER(A-Z)
      COMMON /DAT1/ NODES,ARCS,I(100),J(100),NCOST(100)
      COMMON /DAT2/ LO(3,100),HI(3,100),COST(3,100),NROW(100)
      COMMON /CAL2/ DMCOST(100),FROM(100),FLNODE(100)
      COMMON /CAL3/ LBL(100),FORWD(100),REVER(100),BARC(100)

C     ********************************************************
C               FLOW CHANGE BY FLOW INTO SINK NODE
C     ********************************************************

      END=NODES
    5 START=FROM(END)

      DO 10 A=1,ARCS
        IA=I(A)
        JA=J(A)
        IF((START.EQ.IA).AND.(END.EQ.JA))THEN
          DMCOST(A)=DMCOST(A)-FLNODE(NODES)
          END=START
          IF(END.EQ.1)THEN
            RETURN
          ELSE
            GOTO 5
          ENDIF
        ENDIF
        IF((START.EQ.JA).AND.(END.EQ.IA))THEN
          DMCOST(A)=1000
          END=START
          IF(END.EQ.1)THEN
            RETURN
          ELSE
            GOTO 5
          ENDIF
        ENDIF
   10 CONTINUE
      RETURN
      END
```

```
C     ****************************************************
C                  NODE TIME-CHANGE PROCEDURE
C
C                            STEP 4
C                            ------
C     ****************************************************

      SUBROUTINE STEP4
      IMPLICIT INTEGER(A-Z)
      COMMON /DAT1/ NODES,ARCS,I(100),J(100),NCOST(100)
      COMMON /DAT2/ LO(3,100),HI(3,100),COST(3,100),NROW(100)
      COMMON /CAL1/ TE(100),TF(100),DMHI(100),MKCP(100)
      COMMON /CAL2/ DMCOST(100),FROM(100),FLNODE(100)
      COMMON /CAL3/ LBL(100),FORWD(100),REVER(100),BARC(100)
      COMMON /CAL4/ DEL1(100),DEL2(100),DEL3(100)
      COMMON /OUT1/ TOTIME(50),TOCOST(50),SLCOST(50),CUT(50)
      COMMON /OUT2/ ACTCUT(50,10),ACTLEN(50,10),KNUM
      DATA  MIN1,MIN2,MIN3 / 1000 , 1000 , 1000 /
      DATA  CT1,CT2 / 1 , 1 /

C     ****************************************************
C           PROJECT SHORTENING PROCEDURE BY DELTA
C     ****************************************************

      DO 10 A=1,ARCS
        IA=I(A)
        JA=J(A)
C       ****************************************************
C                          FORWARD ARC
C       ****************************************************
        IF((LBL(IA).EQ.1).AND.(LBL(JA).EQ.0))THEN
          FORWD(A)=1
        ENDIF
C       ****************************************************
C                          REVERSE ARC
C       ****************************************************
        IF((LBL(IA).EQ.0).AND.(LBL(JA).EQ.1))THEN
          REVER(A)=1
        ENDIF
   10 CONTINUE
```

```
C     ******************************************************
C                  CALCULATE DELTA OF EACH ARC
C     ******************************************************

      DO 15 A=1,ARCS
        IA=I(A)
        JA=J(A)
        NR=NROW(A)
        IF((MKCP(A).EQ.0).AND.(FORWD(A).EQ.1))THEN
          DEL1(A)=TF(JA)-TF(IA)-HI(NR,A)
        ELSE
          IF((MKCP(A).EQ.1).AND.(FORWD(A).EQ.1))THEN
            DEL2(A)=TF(JA)-TF(IA)-LO(NR,A)
          ELSE
            IF((MKCP(A).EQ.1).AND.(REVER(A).EQ.1))THEN
              DEL3(A)=HI(NR,A)+TF(IA)-TF(JA)
            ENDIF
          ENDIF
        ENDIF
   15 CONTINUE


C     ******************************************************
C                     DEFINE MINIMUM DELTA
C     ******************************************************

      DO 25 A=1,ARCS
        IF((DEL1(A).LT.MN1).AND.(DEL1(A).GE.1))THEN
          MN1=DEL1(A)
        ELSE
          IF((DEL2(A).LT.MN2).AND.(DEL2(A).GE.1))THEN
            MN2=DEL2(A)
          ELSE
            IF((DEL3(A).LT.MN3).AND.(DEL3(A).GE.1))THEN
              MN3=DEL3(A)
            ENDIF
          ENDIF
        ENDIF
   25 CONTINUE

      DELTA=MIN0(MN1,MN2,MN3)
```

```
C     ******************************************************
C                    TIME SHORTENING BY DELTA
C     ******************************************************

      DO 30 A=1,NODES
        IF(LBL(A).EQ.0)THEN
          IF(TE(A).EQ.TF(A))THEN
            TE(A)=TE(A)-DELTA
            TF(A)=TF(A)-DELTA
          ELSE
            TF(A)=TF(A)-DELTA
            IF(TF(A).LT.TE(A))THEN
              TE(A)=TE(A)-DELTA
            ENDIF
          ENDIF
        ENDIF
   30 CONTINUE

C     ******************************************************
C                           NEW ITERATION
C     ******************************************************

      KNUM=KNUM+1
      TOTIME(KNUM)=TE(NODES)

C     ******************************************************
C            LEAST COST and ACTIVITIES TO CONSIDER
C     ******************************************************

      DO 35 A=1,ARCS
        NR=NROW(A)
        IF((MKCP(A).EQ.1).AND.(FORWD(A).EQ.1))THEN
          SLCOST(KNUM)=SLCOST(KNUM)+(DELTA*COST(NR,A))
          DMHI(A)=DMHI(A)-DELTA
          ACTCUT(KNUM,CT1)=A
          CT1=CT1+1
        ELSEIF((MKCP(A).EQ.1).AND.(REVER(A).EQ.1))THEN
          SLCOST(KNUM)=SLCOST(KNUM)-(DELTA*COST(NR,A))
          DMHI(A)=DMHI(A)+DELTA
          IF(COST(NR,A).NE.0)THEN
            ACTLEN(KNUM,CT2)=A
            CT2=CT2+1
          ENDIF
        ENDIF
   35 CONTINUE
```

```
C     ******************************************************
C               SECOND ARC WITH INFINITE CAPACITY
C     ******************************************************

      DO 40 A=1,ARCS
        NR=NROW(A)
        IF((DMCOST(A).EQ.0).AND.(DMHI(A).EQ.LO(NR,A)))THEN
          N1=NR+1
          IF(HI(N1,A).EQ.0)THEN
            DMCOST(A)=1000
          ELSE
            NROW(A)=NROW(A)+1
            NR1=NROW(A)
            DMCOST(A)=COST(NR1,A)-COST(NR1-1,A)
            DMHI(A)=HI(NR1,A)
          ENDIF
        ENDIF
        IF((DMCOST(A).EQ.1000).AND.(DMHI(A).NE.LO(NR,A)))THEN
          DMCOST(A)=0
        ENDIF
   40 CONTINUE

C     ******************************************************
C                 TOTAL COST OF THE PROJECT
C     ******************************************************

      CUT(KNUM)=TOTIME(KNUM-1)-TOTIME(KNUM)
      SLCOST(KNUM)=SLCOST(KNUM)/CUT(KNUM)
      TOCOST(KNUM)=TOCOST(KNUM-1)+(SLCOST(KNUM)*CUT(KNUM))

      RETURN
      END
```

```
C     ******************************************************
C                        INITIALIZE VARIABLES
C     ******************************************************

      SUBROUTINE INITIAL
      IMPLICIT INTEGER(A-Z)
      COMMON /DAT1/ NODES,ARCS,I(100),J(100),NCOST(100)
      COMMON /DAT2/ LO(3,100),HI(3,100),COST(3,100),NROW(100)
      COMMON /CAL1/ TE(100),TF(100),DMHI(100),MKCP(100)
      COMMON /CAL2/ DMCOST(100),FROM(100),FLNODE(100)
      COMMON /CAL3/ LBL(100),FORWD(100),REVER(100),BARC(100)
      COMMON /CAL4/ DEL1(100),DEL2(100),DEL3(100)
      COMMON /OUT1/ TOTIME(50),TOCOST(50),SLCOST(50),CUT(50)
      COMMON /OUT2/ ACTCUT(50,10),ACTLEN(50,10),KNUM

      DATA  FIX1,FIX2,FIX3,FIX4  / 3 , 50 , 1 , 10 /
      DATA  ZERO,INF  / 0 , 1000 /

      CALL BLANK2(LO,ARCS,FIX1,ZERO)
      CALL BLANK2(HI,ARCS,FIX1,ZERO)
      CALL BLANK2(COST,ARCS,FIX1,ZERO)

      CALL BLANK1(TOTIME,FIX2,ZERO)
      CALL BLANK1(TOCOST,FIX2,ZERO)
      CALL BLANK1(SLCOST,FIX2,ZERO)
      CALL BLANK1(CUT,FIX2,ZERO)

      CALL BLANK1(DEL1,ARCS,INF)
      CALL BLANK1(DEL2,ARCS,INF)
      CALL BLANK1(DEL3,ARCS,INF)

      CALL BLANK1(TE,NODES,ZERO)
      CALL BLANK1(TF,ARCS,INF)

      CALL BLANK1(NROW,ARCS,FIX3)

      CALL BLANK2(ACTCUT,FIX2,FIX4,ZERO)
      CALL BLANK2(ACTLEN,FIX2,FIX4,ZERO)

      RETURN
      END
```

```
C     ******************************************************
C             INITIALIZE VARIABLES - ONE DIMENSION
C     ******************************************************

      SUBROUTINE BLANK1(ARRAY,N,VALUE)
      IMPLICIT INTEGER(A-Z)
      DIMENSION ARRAY(N)
      DO 5 A=1,N
        ARRAY(A)=VALUE
    5 CONTINUE
      RETURN
      END


C     ******************************************************
C             INITIALIZE VARIABLES - TWO DIMENSIONS
C     ******************************************************

      SUBROUTINE BLANK2(ARRAY,N1,N2,VALUE)
      IMPLICIT INTEGER(A-Z)
      DIMENSION ARRAY(N1,N2)
      DO 10 A1=1,N1
        DO 5 A2=1,N2
          ARRAY(A1,A2)=VALUE
    5   CONTINUE
   10 CONTINUE
      RETURN
      END
```

```
C     ******************************************************
C                        PRINT ALL ITERATIONS
C     ******************************************************

      SUBROUTINE PRINT
      IMPLICIT INTEGER(A-Z)
      COMMON /DAT1/ NODES,ARCS,I(100),J(100),NCOST(100)
      COMMON /DAT2/ LO(3,100),HI(3,100),COST(3,100),NROW(100)
      COMMON /CAL1/ TE(100),TF(100),DMHI(100),MKCP(100)
      COMMON /CAL2/ DMCOST(100),FROM(100),FLNODE(100)
      COMMON /CAL3/ LBL(100),FORWD(100),REVER(100),BARC(100)
      COMMON /CAL4/ DEL1(100),DEL2(100),DEL3(100)
      DATA ZERO,INF / 0 , 1000 /

      WRITE(*,1)  (M,TE(M),TF(M),LBL(M),FLNODE(M),FROM(M),
     +             M=1,NODES)

      WRITE(*,5)  (M,MKCP(M),DMHI(M),DMCOST(M),FORWD(M),
     +            REVER(M),DEL1(M),DEL2(M),DEL3(M),M=1,ARCS)

      CALL BLANK1(FORWD,ARCS,ZERO)
      CALL BLANK1(REVER,ARCS,ZERO)
      CALL BLANK1(DEL1,ARCS,INF)
      CALL BLANK1(DEL2,ARCS,INF)
      CALL BLANK1(DEL3,ARCS,INF)

    1 FORMAT(//,' NODE #  TE(N)  TF(N)  LBL(N)  FLNODE(N)',
     +       2X,'FROM(N)',/,(4(1X,I5),2(1X,I10)))

    5 FORMAT(//,5X,' A    CP    DMHI  DMCOST ',4X,
     +          'FOR    REV    D1    D2    D3',/,(9(1X,I6)))

      RETURN
      END
```

```
C     ************************************************************
C                            PRINT RESULTS
C     ************************************************************

      SUBROUTINE ANSWER
      IMPLICIT INTEGER(A-Z)
      COMMON /OUT1/ TOTIME(50),TOCOST(50),SLCOST(50),CUT(50)
      COMMON /OUT2/ ACTCUT(50,10),ACTLEN(50,10),KNUM

      WRITE(*,5)
      WRITE(*,10)
      WRITE(*,15)  (M,TOTIME(M),TOCOST(M),SLCOST(M),CUT(M),
     +             (ACTCUT(M,M1),M1=1,6),
     +             (ACTLEN(M,M2),M2=1,3),M=1,KNUM)

    5 FORMAT(//,7X,'TIME',4X,'TOTAL',2X,'SLOPE',3X,'e/o',/,
     +          1X,'No.',1X,'DURATION',2X,'COST',3X,'COST',
     +          4X,'CUT',7X,'TASKS TO SHORTEN',
     +          8X,'TASKS TO LENGTHEN')

   10 FORMAT(,/,1X,'---',1X,'--------',2X,'-----',2X,'-----',
     +          3X,'---',5X,'--------------------',
     +          4X,'-----------------')

   15 FORMAT(/,(1X,I2,I8,I9,I6,I7,I7,5(I4),I9,2(I4)))

      RETURN
      END
```