A SINGLE STEP CPM TIME-COST TRADEOFF ALGORITHM

Thesis submitted in partial fulfillment of the requirement for

The award of the degree of

Master of Science

In

Mathematics and Computing

Submitted by

Gurpreet Kaur

Roll No. – 300803007

Under

the guidance of

Dr. Mahesh Kumar Sharma

JULY 2010

School of Mathematics and Computer Applications

Thapar University

Patiala – 147001 (PUNJAB)

INDIA

CERTIFICATE

I hereby certify that the work which is being presented in the thesis entitled ***"A Single Step CPM Time-Cost Tradeoff Algorithm"*** *in partial fulfillment of the requirements for the award of degree of Master of Science, School of Mathematics and Computer Applications, Thapar University, Patiala is an authentic record of my own work carried out under the supervision of Dr. Mahesh Kumar Sharma.*

The matter presented in this thesis has not been submitted for the award of any other degree of this or any other university.

Gurpreet Kaur.

(Gurpreet Kaur)
Reg. No.300803007

This is to certify that the above statement made by the candidate is correct and true to the best of my knowledge.

(Dr. Mahesh Kumar Sharma)
Assistant Professor
SMCA, Thapar University
Patiala.

Countersigned by:

Dr. S.S. Bhatia
(Professor & Head)
School of Mathematics & Computer Applications
Thapar University, Patiala.

15/7

Dr. R.K. Sharma
Dean of Academic Affairs
Thapar University
Patiala.

23/7/1

ACKNOWLEDGEMENTS

The successful completion of task is often due less to any innate talent of an individual than to the fact that he has been blessed by Almighty and fortunate in the individuals with whom he has been associated. I owe the successful completion of my work to the blessings, guidance and encouragement of many personalities.

It transcends all the barriers of written words to owe a deep sense of gratitude to my most honorable, worthy and esteemed Supervisor ***Dr. Mahesh Kumar Sharma***, ***Assistant Professor***, *School of Mathematics & Computer Applications, Thapar University, Patiala for his invaluable guidance, keen interest, constant inspiration and encouragement. I have no words to express my heart felt thanks and sincere regards for my learned guide and revered teacher. I consider it my great privilege to have had the opportunity to work under his able guidance.*

I am also grateful to ***Dr. S.S. Bhatia***, ***Head***, *School of Mathematics and Computer Applications, Thapar University, Patiala for his encouragement and for providing the necessary research facilities in the school.*

I am thankful to ***Dr. A.K. Lal***, ***P.G. Coordinator*** *and other faculty members of School of Mathematics and Computer Applications, Thapar University, Patiala for their help rendered by them during the course of my stay in the University.*

I would like to place on record my special reverence to my parents because without their love, patience, support, encouragement, selflessness and abound blessings I would have never been what I am today. I owe them everything.

Above all, I am thankful to the ***God***, *Almighty, without whose grace none can ever succeed.*

Gurpreet Kaur

(GURPREET KAUR)

ABSTRACT

The development and use of the time-cost tradeoff for a project network is an essential part of a CPM (Critical Path Method) analysis, since small variations in project duration could heavily affect the project's cash flow. Managing projects involves the usual management functions of planning, scheduling and controlling. CPM, which is a planning tool, helps to do all three. One of the most salient features of CPM is the attempt to optimize the total project cost (when project duration must be reduced) by shortening appropriate project activities. CPM is a bottleneck route that reduces over all project time and provides a cost evaluation of "Crash" programs. CPM focused on optimizing the total costs (i.e., overhead plus activity costs) for various possible project completion dates; however, no systematic procedure was initially proposed to effectively achieve this goal of cost optimization.

Present thesis deals with a single step CPM time-cost tradeoff algorithm for efficiently shortening the duration of a project when the expected project duration is incompatible (exceeds) with pre-determined or externally imposed requirements.

Thesis contains three chapters; first chapter is introductory in nature and also contains a brief review of literature related to the topic. In second chapter, two distinct algorithms on CPM time-cost tradeoff have been reviewed. In third chapter, a single step CPM time-cost algorithm has been given consisting of a combination of previously discussed two algorithms in second chapter.

CONTENTS

CHAPTER – 1
INTRODUCTION

CHAPTER – 1

INTRODUCTION

The development and use of the time-cost tradeoff for a project network is an essential part of a CPM analysis, since small variations in project duration could heavily affect the project's cash flow. In a project network, each job has an associated crash completion time and normal completion time and the cost of doing the jobs varies linearly between these extreme times. Given that the entire project must be completed in a prescribed time interval, it is desired to find job times that minimize the total project cost. The essential ingredient of the technique is the network model that incorporates sequence information, durations and costs for each component of the project. Analysis of the network model enables operating personnel to answer the questions concerning labour needs, budget requirements, the effects of delays etc.

Network analysis has caught the attention and stimulated the interest of mathematicians, academicians, operations research analysts, practitioners, and government officials. The most common name associated with network analysis is CPM. CPM is not just a data manipulation tool, but a logical approach to work. Managing projects involves the usual management functions of planning, scheduling and controlling. CPM, which is a planning tool, helps to do all three. One of the most salient features of CPM is the attempt to optimize the total project cost (when project duration must be reduced) by shortening appropriate project activities. CPM is a bottleneck route that reduces over all project time and provides a cost evaluation of "Crash" programs. CPM focused on optimizing the total costs (i.e. overhead plus activity costs) for various possible project completion dates, however no systematic procedure was initially proposed to effectively achieve this goal of cost optimization.

1.1. INTRODUCTION TO PROJECT AND NETWORKS

A project is an endeavour to create a unique product or service. It is specific, timely and always conflict ridden. Projects are part of an overall programme and are broken down into well-defined set of tasks (jobs), subtask and further if desired, all of which must be completed in the specified time along with minimum cost. Examples of projects include

construction of a bridge, highway, power plant, repair and maintenance of an oil refinery or an air plane, design, development and marketing of a new product, research and development work, etc. Such projects involve large number of interrelated activities (or tasks) which must be completed in a specified time, in a specified sequence (or order) and require resources such as personnel, money, materials, facilities and/or space. The main objective before starting any project is to schedule the required activities in an efficient manner so as to complete it on or before a specified time limit at a minimum cost of its completion.

In managing a project, there are important relationships among various activities that are often better presented visually. For example, certain activities may not begin until others are completed. Other activities may be able to proceed simultaneously. One excellent way to visually represent these project activities is through a network. A network is a graphical plan consisting of certain configuration of arrows and nodes for showing the logical sequence of various activities to be performed to achieve project objectives. The project network may be of the form of Activity-on-Node (AON) network and Activity-on-Arc (AOA). In AON each node (or circle) represents a specific task while the arcs represent the ordering between tasks. AON network diagrams as shown in figure 1.1 (a) place the activities within the nodes, and the arrows are used to indicate sequencing requirements. The lack of dummy activities in these diagrams always make them easier to draw and to interpret whereas AOA network diagrams as shown in figure 1.1 (b) has each end of the activity arc as a node (or circle). These nodes represent points in time or instants, when an activity is starting or ending. The arc itself represents the passage of time required for that activity to be performed.

In some cases, we may need to indicate precedence, but any relevant activity is already being used to indicate another precedence relation. In those cases, a dummy activity can be used and is represented by a dashed arrow. Dummy activities have duration of zero and are used only to show precedence. A second situation in which a dummy activity may be useful is when each activity must have a unique pair of beginning and ending events.

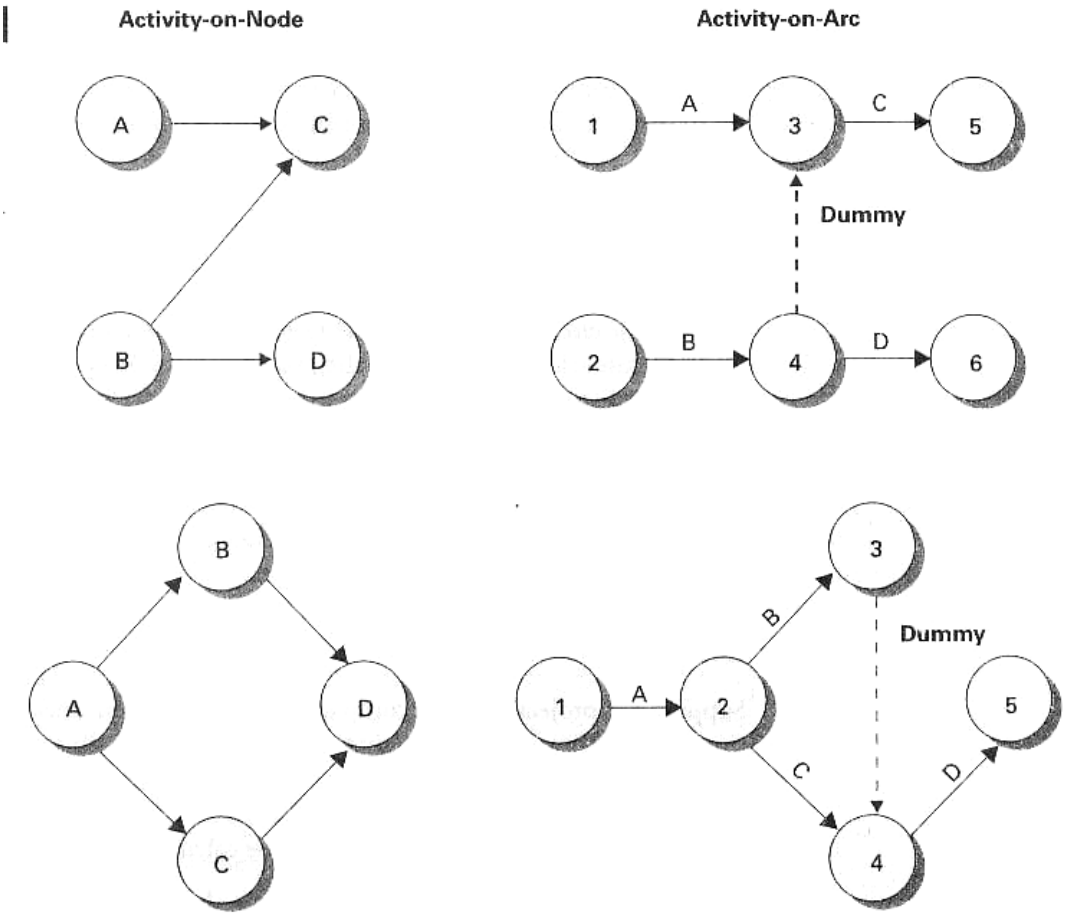

Figure 1.1 (a) **Figure 1.1 (b)**

1.2. THE CRITICAL PATH METHOD

CPM was developed by E.I. DuPont company along with Remington Rand Corporation. The aim behind its development was to provide a technique for control of the maintenance of company's chemical plants. CPM was designed to help plan how to coordinate a project's various activities, develop a realistic schedule for the project and then monitor the progress of the project. In course of time, use of CPM got extended to the field of cost and resource allocation.

CPM provides a way to determine the earliest time each activity could start, based on expected durations, precedence relationships and the expected entire duration. By working backward from a desired completion date for the project, it will also be possible to determine the latest time each activity could start and finish without delaying the entire project. Finally, critical path can be calculated, which will consist of those activities

that determine how long it will take to complete the entire project.

Calculating Start and Finish Times for AON networks

In AON networks, earliest start and earliest finish times can be calculated for each activity as these gives different values. Similarly, latest start and latest finish times also gives different values for each activity. So these can be calculated as below:

Earliest Times

If we let time zero represent the starting time for this project, then first activity can being immediately at time zero. The expected finish time can be found by adding the expected duration to the starting time of zero. We will designate these starting and finish times as the earliest start (ES) and earliest finish (EF) because they are the earliest time the activity can start and finish, based on the starting time of zero and expected duration. For any activity, its earliest start time will be the time at which all of its predecessor activities are completed. The general rule is

$$EF = ES + \text{activity duration}$$

The earliest start and earliest finish times for all activities in the project can be calculated the same way by adding the earliest start and earliest finish information. If an activity has more than one predecessor, its earliest start time will be the latest of the predecessor's earliest finish times. It means is that no activity can start until all of its predecessors are finished.

Latest Times

Suppose the project must be completed within nineteen weeks. That represents the latest time of the last activity of the project can finish without making the project late. At each node, the latest start time can be calculated using the general rule

$$LS = LF - \text{duration}$$

Keep working backward from the desired completion date to determine latest finish and latest start times by realizing that all of an activity's predecessors must be completed by its latest start time. The latest finish time of an activity will be the earliest of the latest start times of the

activity it precedes. All of an activity's predecessors must be finished before it can start.

Avoiding Late Completion

The information about earliest start, earliest finish, latest start, and latest finish can be used to keep a project on schedule. One more bit of information is useful. By comparing the earliest and latest times, we can determine the amount of slack for each activity. The slack is calculated as

$$\text{Slack} = LF - EF \text{ or } \text{Slack} = LS - ES$$

The slack for each activity in the project described above. The slack time, or slack, represents how much leeway each activity has in its starting time and duration.

Critical Path Computations for AOA networks

An activity is said to be critical if there is no "leeway" in determining its start and finish times. A non-critical activity allows some scheduling slack, so that the start time of the activity can be advanced or delayed within limits without affecting the completion time of the entire project.

To carry out the necessary computations in AOA networks, define an event as a point in time at which activities are terminated and others are started. In terms of the network, an event corresponds to a node. Define

$\square_j$ = Earliest occurrence time of event j
Δ_j = Latest occurrence time of event j
D_{ij} = Duration of activity (i, j)

The critical path calculations involve two passes: The forward pass determines the earliest occurrence times of the events, and the backward pass calculates their latest occurrence times.

Forward Pass (Earliest Occurrence Times, $\square$): The computations start at node 1 and advance recursively to end node n.

Initial Step: Set $\square_1 = 0$ to indicate that the project starts at time 0.

General Step j: Given that nodes p, q,....., and v are linked directly to node j by incoming activities (p, j), (q, j),....., and (v, j) and that the earliest occurrence times of events (nodes) p, q,....., and v have already been computed, then the earliest occurrence time of event j is computed as

$$\square_j = \max \{\square_p + D_{pj}, \square_q + D_{qj}, \ldots\ldots, \square_v + D_{vj}\}$$

The forward pass is complete when $\square_n$ at node n has been computed. By definition $\square_j$ represents the longest path (duration) to node j.

Backward Pass (Latest Occurrence Times, Δ): Following the completion of the forward pass, the backward pass computations start at node n and end at node 1.

Initial Step: Set $\Delta_n = \square_n$ to indicate that the earliest and latest occurrences of the last node of the project are the same.

General Step j: Given that node p, q,....., and v are linked directly to node j by outgoing activities (j, p), (j, q),....., and (j, v) and that the latest occurrence times of nodes p, q,....., and v have already been computed, the latest occurrence time of node j is computed as

$$\Delta_j = \min \{\Delta_p - D_{jp}, \Delta_q - D_{jq}, \ldots\ldots, \Delta_v - D_{jv}\}$$

The backward pass is complete when Δ_1 at node 1 is computed.

Based on the preceding calculations, an activity (i, j) will be critical if it satisfies three conditions.

1. $\Delta_i = \square_i$
2. $\Delta_j = \square_j$
3. $\Delta_j - \Delta_i = \square_j - \square_i = D_{ij}$

The three conditions state that the earliest and latest occurrence times of end nodes i and j are equal and the duration D_{ij} fits "tightly" in the specified time span. An activity that does not satisfy all three conditions is thus non-critical.

By definition, the critical activities of a network must constitute an uninterrupted path that spans the entire network from start to finish.

Determination of the Floats: - Floats are the slack times available within the allotted span of the non-critical activity. The most common floats are the total float and the free float.

Figure 1.2 gives a convenient summary for computing the total float (TF_{ij}) and the free float (FF_{ij}) for an activity (i, j). The total float is the excess of the time span defined from the earliest occurrence of event i to the latest occurrence of event j over the duration of (i, j) – that is

$$TF_{ij} = \Delta_j - \square_i - D_{ij}$$

The free float is the excess of the time span defined from the earliest occurrence of event i to the earliest occurrence of event j over the duration of (i, j) – that is.

$$FF_{ij} = \square_j - \square_i - D_{ij}$$

By definition, $FF_{ij} \leq TF_{ij}$

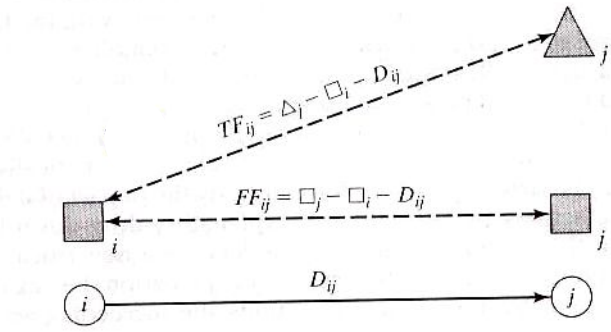

Figure 1.2

Red-Flagging Rule: For a non-critical activity (i, j)

(a) If $FF_{ij} = TF_{ij}$, then the activity can be scheduled anywhere within its ($\square_i$, Δ_j) span without causing schedule conflict.

(b) If $FF_{ij} < TF_{ij}$, then the start of activity (i, j) can be delayed by at most FF_{ij} relative to its earliest start time ($\square_i$) without causing schedule conflict. Any delay larger than FF_{ij} (but not more than TF_{ij}) must be accompanied by an equal delay relative to $\square_j$ in the start time of all the activities leaving node j.

The implementation of the rule is that a non-critical activity (i, j) will be red-flagged if its $FF_{ij} < TF_{ij}$. This red flag is important only if we decide to delay the start of the activity past its earliest start time, $\square_i$, in which case we must pay attention to the start times of the activities leaving node j to avoid schedule conflicts.

The Critical Path

Those activities with the least slack will form a path through the CPM diagram from beginning to end. This path is called the critical path, and its associated activities are said to be critical-path activities. A sequence of project network activities which add up to the longest overall duration is known as critical path. Any delay of an activity on the critical path directly impacts the planned project completion date.

1.3. PROJECT CRASHING: TIME COST TRADE-OFF

Project crashing problem is primarily concerned with the trade-off between compressed activity duration and the consequent increase in the direct cost due to crashing. In many situations, it is possible to reduce the length of a project by injecting additional resources. The impetus to shorten projects may reflect efforts to avoid late penalties or to take advantage of monetary incentives for timely completion of a project, or to free resources for use on other projects. In many cases, however, the desire to shorten the length of a project merely reflects an attempt to reduce the indirect costs associated with running the project, such as facilities and equipment costs, supervision and labour and personnel costs. Managers often have certain options at their disposal which will allow them to shorten, or crash, certain activities. Among the most obvious options are the use of additional funds to support additional personnel or more efficient equipment, and the relaxing of some work specifications. Hence, a project manager may be able to shorten a project, thereby realizing a saving in indirect project costs, by increasing direct expenses to speed up the project. The goal of evaluating time-cost trade-offs is to identify a plan which will minimize the sum of the indirect and direct project costs.

There are some costs which can be directly attributed to the production of a unit of a given product. Such costs are direct costs and can easily be

separated, ascertained and imputed to a unit of output. This is because these costs vary with the output units. However, there are other costs which cannot be separated and clearly attributed to individual units of production. These costs are, therefore, classified as indirect costs in the accounting process. For example, electricity charges may not be separate department-wise in a single product firm or even product-wise in a multiple product firm.

For technical reasons, the duration of an activity cannot be reduced indefinitely. The crash time represents the fully expedited or the minimum activity duration time that is possible, and any attempts to further 'crash' would only raise the activity direct costs without reducing the time. The activity cost corresponding to the crash time is called the crash cost which equals the minimum direct cost required to achieve the crash performance time. These are in contrast to the normal time and the normal cost of the activity. The normal cost is equal to the absolute minimum of the direct cost required to perform an activity. The corresponding activity duration is known as the normal time.

The slope of crashing an activity means the increment cost of expediting the activity per unit period calculated as:

$$\text{Cost slope} = \frac{CC - NC}{NT - CT} = \text{cost to crash per period}$$

Where NT – CT = Maximum time reduction for an activity

CC = Crash Cost, NC = Normal Cost,

NT = Normal Time, CT = Crash Time

In figure 1.3, beyond point A cost increases more quickly when time is reduced. Similarly beyond point B the time increases while the cost decreases. For simplicity the relationship between normal time and cost as well as crash time and cost for an activity is assumed to be linear.

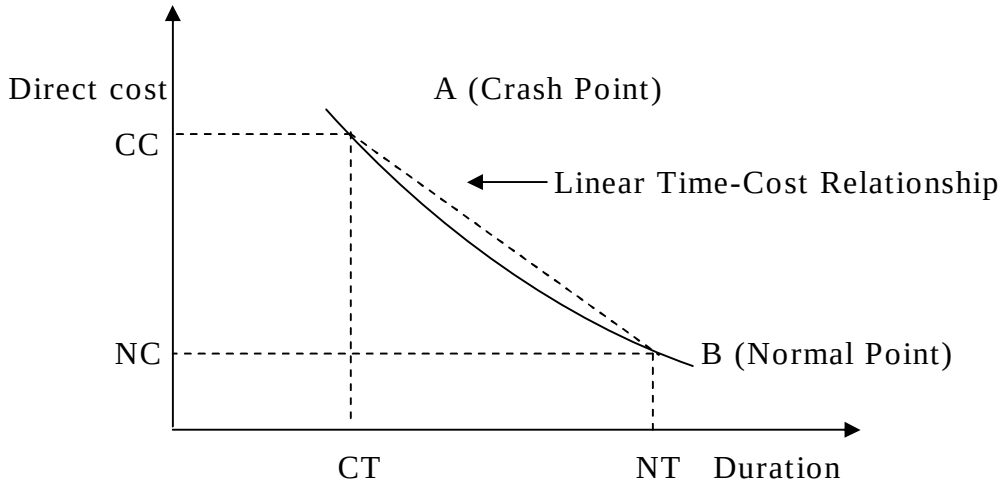

Figure 1.3

In order to make a rational decision on which activities (if any) to crash, and on the extent of crashing desirable, a manager needs certain information on regular time and crash time estimates for each activity, regular cost and crash cost estimates for each activity, and a list of activities which are on the critical path. Activities on the critical path are potential candidates for crashing, since shortening non-critical activities would not have an impact on total project duration. From an economic standpoint, activities should be crashed according to crashing costs: Crash those with the lower costs first. Moreover, crashing should continue as long as the cost to crash is less than the benefits received from crashing. These benefits might take the form of incentive payments for early project completion, or they might reflect savings in the indirect project costs, or both. Figure 1.4 illustrates the basic relationships between indirect, direct, and total project costs due to crashing. Crashing activities reduces indirect project costs while increasing direct costs. The optimum amount of crashing results in minimizing the sum of these two types of costs.

Indirect cost and crashing have positive relation so indirect cost has positive linear curve more prone to x-axis as a decrease in project length lead to a little decrease in indirect cost whereas direct cost has negative relation with crashing the duration shown by the negative slope of the curve means there is a more increase in cost with respect to a little decrease in project length. When project length seems to be infinite direct cost seems almost negligible.

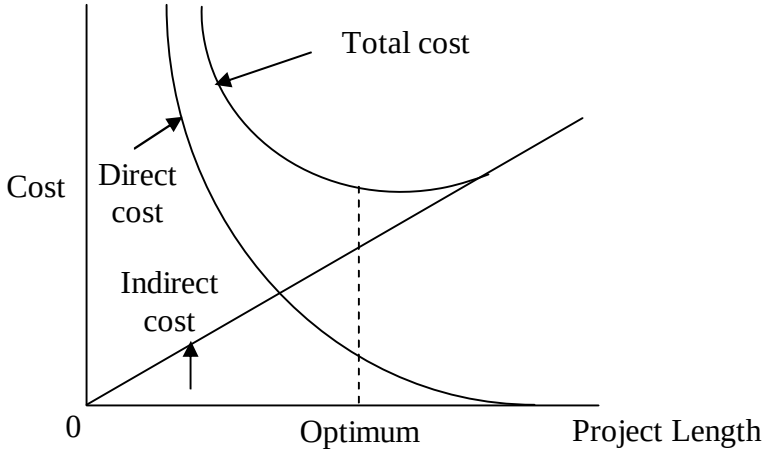

Figure 1.4 Time-cost relationship in projects

1.4. LITERATURE REVIEW

CPM was developed by E.I. DuPont company along with Remington Rand Corporation. The aim behind its development was to provide a technique for control of the maintenance of company's chemical plants. CPM was designed to help plan how to coordinate a project's various activities, develop a realistic schedule for the project, and then monitor the progress of the project. In course of time, use of CPM got extended to the field of cost and resource allocation.

The effort to develop effective techniques for solving the cost optimization problems includes the work of many researchers. Fulkerson (1961) has developed a network flow algorithm to minimize cost, where it is assumed that certain jobs must be finished before others can be started. It is desired to find job times that minimize the total project cost. Each job has an associated crash completion time and normal completion time, and the cost of doing the job varies linearly between these extreme times. Given that the entire project must be completed in a prescribed time interval, it is desired to find job times that minimize the total project cost. The method solves this problem for all feasible time intervals.

Kelley and Walker (1960) have formulated the cost optimization

problem as a linear program solution by means of computer. The essential ingredient of the technique is a mathematical model that incorporates sequence information, durations, and costs for each component of the project. It is a special parametric linear program that, via the primal-dual algorithm, may be solved efficiently by network flow methods. Analysis of the solutions of the model enables operating personnel to answer questions concerning labour needs, budget requirements, procurement and design limitations, the effects of delays, and communication difficulties.

Prager (1963) has developed a structural interpretation of cost optimization problem. His work shows that Fulkerson's (1961) algorithm can be given a structural interpretation using concepts that are familiar to civil engineers.

The work on CPM has been discussed in detail by many research workers. The basics of CPM include the work of Levy *et al* (1963). Their work details the CPM for decision making in project management and includes the critical path algorithm, concept of total slack, characteristics of CPM, scheduling program for allocating resources (SPAR) and examples from the construction industry.

Parikh and Jewell (1965) has given a method to decompose a project network into several sub-networks and then to put the project work back together for the networks with large activities as it becomes difficult to prepare a network for the project as one unit. Also, in the cases where there are two or more independent projects, which are weakly inter-related by common activities, the problem of efficient scheduling of all the projects become quite difficult.

Richard (1968) discussed the development of CPM technique for the monitoring of product development activities of various projects in organization. Features of the CPM techniques, representations of product development activities through CPM, source of project development activities for CPM, Application of CPM system for its scheduling have been discussed in his work.

Saindon (1969) introduced CPM as a management tool for defining analyzing and controlling all phases of a production cycle which has resulted in time and cost savings.

Siemen (1971) has presented his simple CPM time cost tradeoff algorithm describes efficient way of shortening the duration of a project when the expected project duration exceeds a pre-determined limit. The problem consists of determining which activities to expedite and by what amount. The objective is to minimize the cost of the project. With this method the CPM time-cost tradeoff problem is solved without access to a computer, thereby making this planning tool available to managers who otherwise would find implementation impractical. The algorithm is a logical systematic procedure beginning with the development of the project network and terminating when the desired project completion time has been achieved.

Falk and Horowiz (1972) has presented an algorithm for determining the minimum cost schedule of tasks in a CPM in which tasks costs arises may be concave.

Goyal (1975) has given an algorithm on CPM time cost tradeoff with a new concept of de-shortening a previously shortened activity and by redefining effective cost slope, selecting the activity with minimum effective cost slope, and simultaneously de-shortening appropriate activities on adequately shortened paths while shortening the selected activity. By employing the procedure described herein there is apparently less path over-shortening than with the Siemens approach.

Panagiotakopoulos (1977) has presented the development and use of time-cost trade off curve for a project as an essential part of a CPM analysis, since small variations in project duration could heavily affect the project's cash flow. Several procedures exist for developing this curve. However, either they assume unrealistically simplified forms for the cost-time relations of the activities, whereupon the project curve is inaccurate and of little use, or they employ complex and impractical analytical techniques. He suggested a simple and efficient computational algorithm for developing the project time-cost curve for arbitrary and thus realistic, cost-time relations for the activities.

Ahuja et al (1984) has given an algorithm for a cost network problem to determine the minimum cost flow in a network when cost of flow over each arc is given by a piecewise linear convex function.

Baweja (2006) has presented information related to CPM schedule that has

become de facto standard in the construction industry for planning, management and controls of project. The original intent of his work was to survey the two major communities in the construction environment. The goal was to present the findings on why each community uses CPM schedules on construction projects, how effectively they are used and to identify the disagreements, when they occur.

Shankar and Sireesha (2009) has proposed a graphical approach to CPM by constructing a rooted tree for the project network to find all paths in the network in such a way that the longest path in the tree as the critical path. Using the critical path we rank all the paths in the project network and evaluate the float time for each activity in the network. It is established that the new graphical approach is a better alternative for dealing with project networks to find critical path. The procedure includes seven steps to find critical path.

1.5. PRESENT WORK

In chapter two the work of Siemens (1971) and Goyal (1975) has been revised in detail. Siemens (1971) has formulated an algorithm on the time-cost tradeoff for project crashing using CPM, by making the use of the concept of effective cost calculated from given cost slopes for each activity under a Project. Algorithm consists of the problem of determining which activities to expedite and by what amount when expected project duration exceeds a pre-determined limit. The objective is to optimize the cost of the project. Also, Goyal (1975) has formulated a similar kind of algorithm by redefining effective cost slope, selecting the activity with minimum effective cost slope and simultaneously de-shortening appropriate activities on adequately shortened paths while shortening the selected activity.

A single step algorithm has been developed that deals with same kind of project crashing problem without using the concept of effective cost slope which makes it iteration free as we don't need to find the effective cost slope or any other such calculation that makes the whole computation iterative. This algorithm follows the rule of selecting the activity with least cost slope among the critical path and sub-critical paths require shortening rather than selecting an activity with least effective cost only among the activities forming critical path. The use of de-shortening process and comparison of costs has also been made.

CHAPTER – 2

ALGORITHMS ON CPM TIME COST TRADEOFF

CHAPTER – 2

ALGORITHMS ON CPM TIME COST TRADE OFF

2.1. INTRODUCTION

Earlier in chapter first CPM has been discussed. In short project networks, by summing the length of each path one can find out the critical path as this is the longest path in the network. The solution to the problem of reducing the project duration (by some specified amount) at the lowest possible cost is quite elementary whenever the desired project duration is equal to or greater than the second-largest path, i.e., the longest sub-critical path. All that is required in this case is to rank the activities on the critical path in order of ascending cost slope, shortening the lowest-cost slope activities as much as possible and continuing to shorten the progressively higher-cost-slope activities until the critical path has been shortened by the required amount. However, when the desired project duration is less than the second-longest network path (the longest sub-critical path), it is not only necessary to shorten the critical path, but one or more sub-critical paths as well. Now it is not just a simple matter of shortening those activities with the lowest cost slopes. Some activities may be constituents of the critical path as well as one or more sub-critical paths requiring reduction. Consequently, it may actually be less costly to reduce a moderately high cost slope activity that is common to several paths requiring shortening than to reduce the lowest cost slope activity that occurs only on one of the paths that need shortening.

Siemens (1971) developed SAM (Siemens Approximation Method) for efficiently shortening the duration of a project when the expected project duration exceeds a predetermine limit. The problem consists of determining which activities to expedite and by what amount. The objective is to minimize the cost of the project. The objective includes a method for efficiently shortening the duration of a project when the expected project duration is incompatible with (exceeds) pre-determined or externally imposed requirements. A typical example is when the scheduled project completion time cannot be met in the due course of normal activity. Consequently, to meet the externally imposed (scheduled) completion time it is necessary to expedite one or more activities to achieve this desired completion time. The problem then

involves determining which activities to expedite (shorten) and to what extent the various activities should be shortened.

Since the cost of shortening any activity by one day is typically not the same for each activity in the project, then the rational project administrator would not be in-different as to which activities to shorten in order to meet the scheduled (required) completion time. Furthermore, there is obviously a finite limit to which a given activity can be shortened. Consequently, a systematic approach is required to implement the reduction of the project duration in accordance with a rational criterion – the rational criterion typically being the objective of minimizing the additional costs incurred due to compressing the project duration by the amount specified.

2.2. THE STEPS OF A SIMPLE CPM TIME COST TRADEOFF ALGORITHM:

1. Develop the project network.
2. Identify all the alternate (parallel) paths through the network and determine the expected completion times for each of these paths. (The longest path is the critical path and determines the expected project duration).
3. Determine the desired project completion time. (This is typically an exogenous constraint imposed by considerations outside the jurisdiction of the network analyst, e.g., the government contract specifies when the project must be completed).
4. Determine the amount each alternate path in the network must be shortened to achieve the desired project completion time. (not all paths may require shortening–one or more paths may already require less time than is available to complete the project. Whenever the expected completion time of any path is equal to or less than the desired project completion time, then obviously this path need not be shortened). The amount each path must be shortened is the amount by which the path expected time exceeds the desired project completion time, i.e., amount to be shortened = (path expected time) – (desired project completion time).
5. Estimate the cost slope (cost per unit of time saved) for each activity in the network as well as the maximum amount that each activity can be shortened.

6. Develop the time-cost matrix.

    a) Each row will represent one activity.

    b) Each column will represent an alternate path through the network. (Only those paths which need shortening should be included in the matrix).

    c) The last two columns in the matrix are used to record (i) the cost slope of each activity and (ii) the maximum amount that each of the various activities can be shortened, respectively.

    d) The column totals (excluding the last two columns) constitute the minimum amounts that the various paths must be shortened to meet the desired project completion date.

    e) In each column, cross out those activities that are not included in (are not part of) the path represented by the column. (none of the paths will contain each activity).

7. Determine the effective cost slope of each activity by modifying the actual cost slope according to the following procedure:

    a) Determine which paths have not yet been adequately shortened. (Initially this will be every path listed in the matrix).

    b) Divide the actual cost slope of each activity by the number of inadequately shortened paths that include (contain) this activity. This result is the effective cost slope.

    c) Record the effective cost slope for each activity in each column of the matrix.

    d) Revise the effective costs of the various activities (as required) whenever any path has been adequately shortened, i.e. when the column requirements have been met. (not all activities will require revision of effective costs. Procedure for revision is described in 7(b) above).

8. Select the column (path) that still has the greatest need, i.e., greatest unfilled demand (difference between the amount required in the column and the amount already allocated to the column). Initially this selected column (path) will be the original critical path. If this greatest need (greatest unfilled demand) is common to more than one path,

discriminate in favour of the path containing the activity with the lowest effective cost slope.

Within this column, select the activity with the lowest effective cost slope, limiting the selection to those activities that still have a supply of time available for shortening. If this lowest effective cost slope is common to more than one activity in the selected column (path), then the following procedure can be used to discriminate between (or among) these tied activities within the selected column:

a) Discriminate in favour of the activity which is common to (i.e., is a component of) the greatest number of currently inadequately shortened paths.

b) If a choice of more than one activity still exists, then discriminate in favour of the activity that permits the greatest amount of shortening at the current effective cost slope.

   The amount that an activity may be shortened at any stage is limited by: (i) the amount of unallocated time remaining for the activity, as well as (ii) the path (of which this activity is a component) that currently has the least (demand) for shortening.

c) If the activity selection still has not been uniquely determined, discriminate in favour of the activity (within the selected column) that is common to the greatest number of paths in the matrix (adequately shortened as well as inadequately shortened paths).

9. Make the allocations in the matrix, i.e., the amount an activity will be shortened – initially, as well as at subsequent, additional allocations (to augment previous allocations, if required) – according to the following procedure:

   Allocate as much time as possible to the activity selected in Step 8 above, subject to:

   a) The current unfilled demand in any column (path) containing the selected activity (the activity being shortened). Ignore paths that have already been adequately shortened. These adequately shortened paths may coincidentally become shortened by more than the minimum required amount.

b) The amount of time still available for shortening the activity (as determined by the original supply available in the last column minus any previous allocations made to that activity).

c) The smallest quantity of the above two criteria ((a) or (b)) determines the actual quantity of time to be allocated to the selected activity.

The effect of the procedure in Step 9 is to allocate as much time as possible to the selected activity without changing the effective cost slope or exceeding the available supply of the activity.

10. Allocate this quantity (as determined in Step 9 above) to each column containing the activity selected above. Whenever the supply of any activity is, exhausted, this row is crossed out. Similarly, when a path requirement has been met the column is crossed out. Crossing out rows or columns eliminates them from further consideration.

11. Repeat Steps 7 through 10 until all paths have been adequately shortened (as indicated by the requirements at the bottom of each column). In some instances this will result in some paths being shortened by more than the minimum required amount (since some activities are common to more than one path and shortening the activity will simultaneously reduce the length of all paths on which it occurs).

12. The application of the preceding eleven steps will result in an optimal solution.

Although this algorithm may initially appear rather formidable and obscured with detail, in practice it is not difficult to execute and can be assimilated quite readily. Of course, the procedure is not limited to hand computation, since the method is not only appropriate as a computer application, but also involves very little effort to develop the elementary computer program required to solve the problem.

2.3. EXAMPLE PROBLEM

Perhaps an elementary, illustrative problem is considered to demonstrate the algorithm and to avoid any ambiguity or abstruseness that may exist in the instructions.

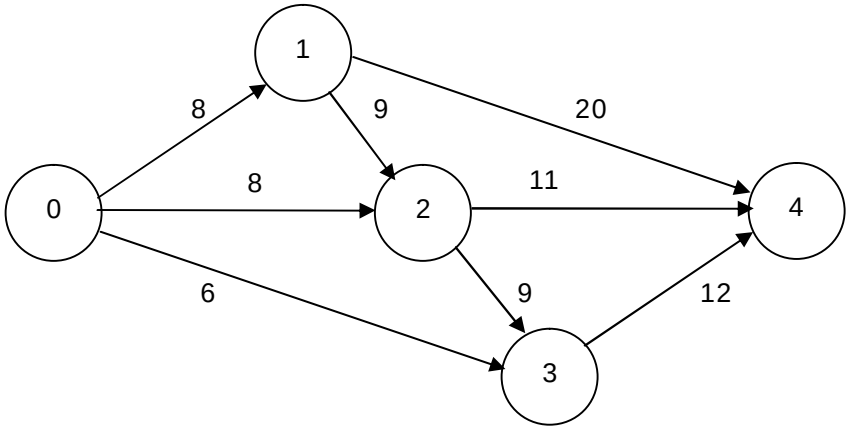

Figure 2.1 Project Network

The project network is depicted in figure 2.1. The numbers on arcs of the network diagram are the expected times of the various activities. Total expected completion time is 38 days given by the path 0-1-2-3-4. It is desired to shorten the project to 22 days. There are six alternate paths through the network: 0-1-4; 0-1-2-4; 0-1-2-3-4; 0-2-3-4; 0-2-4; 0-3-4. Only the first four paths need to be shortened–the other two are already less than 22 days. The first four paths have expected completion times of 28, 28, 38 and 29 days, respectively, and consequently must be shortened by 6, 6, 16 and 7 days, respectively, to meet the desired completion date. The cost slope for each of the eight activities and the maximum time reduction possible for each activity is estimated to be as follows:

Activity	Cost Slope	Maximum Time Reduction
0 – 1	9	5
0 – 2	2	4
0 – 3	4	3
1 – 2	6	7
1 – 4	3	7
2 – 3	7	5
2 – 4	1	5
3 – 4	8	7

This information can now be used to develop the time-cost matrix (Step 6) and determine the effective cost slope of each activity (Step 7). The effective cost slope is recorded in the upper portion of the box on each inadequately shortened path that contains the activity. The time-cost matrix is illustrated in figure 2.2.

Activity	Paths Requiring Reduction				Cost Slope	Time Reduction Possible (Supply)
	014	0124	01234	0234		
0–1	3	3	3		9	5
0–2				2	2	4
0–3					4	3
1–2		3	3		6	7
1–4	3				3	7
2–3			3.5	3.5	7	5
2–4		1			1	5
3–4			4	4	8	7
Time Reduction Required (Demand)	6	6	16	7		

Figure 2.2: Time-Cost Matrix

The column (path) with the greatest need is 0-1-2-3-4 (Step 8). The activity (within this column) with the lowest effective cost ($3 per day) is activity 0–1 (Step 8(a)). Shorten this activity by 5 days (Step 9(b)). This allocation is recorded in each of the first three columns, since all three of these paths contain activity 0–1. Activity 0–1 can now be crossed out to indicate that it is not available for further time reduction. This is shown in figure 2.3.

Column 0-1-2-3-4 still has the greatest unfilled demand and the activity (within this column) with the lowest effective cost is activity 1–2 ($3 per day). Allocate 1 day to this activity (Step 9(a)). Since path 0-1-2-4 is now adequately shortened, it can be crossed out to indicate that its requirements have been satisfied. The effective cost slope of activity 1–2 must now be revised (Step 7(d)) and recorded in the matrix. The effective cost of activity 1–2 becomes $6 per day since it only benefits one path now. These changes are shown in figure 2.3.

Activity	Paths Requiring Reduction				Cost Slope	Time Reduction Possible (Supply)
	014	0124	01234	0234		
0–1	3 5	3 5	3 5		9	5
0–2				2	2	4
0–3					4	3
1–2		 1	6 1		6	7
1–4	3				3	7
2–3			3.5	3.5	7	5
2–4		1			1	5
3–4			4	4	8	7
Time Reduction Required (Demand)	6	6	16	7		

Figure 2.3: Time-Cost Matrix

Column 0-1-2-3-4 still continues to manifest the greatest unfilled demand and the activity (within this column) with the lowest effective cost ($3.5 per day) is activity 2-3. Allocate 5 days to this activity (Step 9(b)). Activity 2-3 must now be crossed out to indicate that no further time reduction potential exists in figure 2.4.

The lowest-effective-cost activity in the path with the greatest need is now activity 3-4 ($4 per day) in path 0-1-2-3-4. Activity 3-4 can be shortened 2 days (Step 9(a)), fulfilling the requirements of path 0-2-3-4. Path 0-2-3-4 is therefore crossed out and the effective cost of activity 3-4 is revised to $8 per day. These changes are shown in figure 2.4.

Activity	Paths Requiring Reduction				Cost Slope	Time Reduction Possible (Supply)
	014	0124	01234	0234		
0–1	3 5	3 5	3 5		9	5
0–2				2	2	4
0–3					4	3
1–2			6		6	7
		1	1			
1–4	3				3	7
2–3			3.5	3.5	7	5
			5	5		
2–4		1			1	5
3–4			8		8	7
			2	2		
Time Reduction Required (Demand)	6	6	16	7		

Figure 2.4: Time-Cost Matrix

The final reduction of column 0-1-2-3-4 consists of allocating 3 additional days to activity 1-2, making a total allocation of 4 days. Since the requirements of path 0-1-2-3-4 have now been satisfied, it is crossed out. The final allocation is 1 day to activity 1-4, completing the requirements of path 0-1-4. These allocations are illustrated in figure 2.5. (it is not necessary to have more than one matrix to solve the problem, but in order to clearly illustrate the

step-by-step sequence it is helpful to show the additional changes that were made from one figure to another).

Activity	Paths Requiring Reduction				Cost Slope	Time Reduction Possible (Supply)
	014	0124	01234	0234		
0–1	3 5	3 5	3 5		9	5
0–2				2	2	4
0–3					4	3
1–2		 4	6 4		6	7
1–4	3 1				3	7
2–3			3.5 5	3.5 5	7	5
2–4		1			1	5
3–4			 2	 2	8	7
Time Reduction Required (Demand)	6	6	16	7		

Figure 2.5: Time-Cost Matrix

All paths have now been adequately shortened. Sometimes one or more paths may become excessively shortened. In some instances this is simply a coincidental by product and no further economic benefits can be achieved, while on other occasions the optional procedure of Step 12 will result in further minor improvement. In this example only path 0-1-2-4 was excessively shortened (9 days instead of the required 6 days). This was merely a coincidental by product of shortening path 0-1-2-3-4 and no further improvement in the solution is possible.

Solutions to the example problem are summarized below for the arbitrarily selected project duration of 22 days (i.e., the example discussed in illustrating

the procedure), 24 days, 26 days, and 28 days (i.e., shortening the normal project duration by 16, 14, 12, and 10 days, respectively).

Project Duration

Activity	22 days		24 days		26 days		28 days	
	Amount Shortened	Cost	Amount Shortened	Cost	Amount Shortened	Cost	Amount Shortened	Cost
0–1	5	45	4	36	2	18	0	0
0–2	0	0	0	0	0	0	0	0
0–3	0	0	0	0	0	0	0	0
1–2	4	24	5	30	7	42	7	42
1–4	1	3	0	0	0	0	0	0
2–3	5	35	5	35	3	21	3	21
2–4	0	0	0	0	0	0	0	0
3–4	2	16	0	0	0	0	0	0
Totals:	17 days	$123	14 days	$101	12 days	$81	10 days	$63

The above solutions are identical to the results obtained by means of the linear programming algorithm – not only with respect to the cost of the individual solutions, but also with regard to the amounts that the various activities were shortened.

The rules for SAM were developed as a result of intuitive logic and analysis and were verified by empirical results. Solutions obtained by means of SAM were compared with linear programming results derived by means of a computer. These comparisons confirmed the validity and effectiveness of SAM in solving problems of this nature. The SAM algorithm is a logical systematic procedure beginning with the development of the project network and terminating when the desired project completion time has been achieved. It is assumed that the objective is to minimize the total cost of reducing the project duration by some specified amount and multiple paths frequently share the benefits of shortening a specific activity.

2.4. A MODIFIED ALGORITHM FOR CPM TIME-COST TRADEOFF

Siemens describes an algorithm (SAM) for shortening the duration of a project when the expected duration of the project exceeds a predetermined limit. The objective is to minimize the total cost of expediting activities in order to complete the project in the desired time. The algorithm gives the rules for choosing paths and activities which must be expedited. This suggests an alternative technique given by Goyal (1975) which, in contrast to SAM, allows for de-shortening a previously shortened activity. This is accomplished by redefining "effective cost slope", selecting the activity with minimum effective cost slope, and simultaneously de-shortening appropriate activities on adequately shortened paths while shortening the selected activity. By employing the procedure described herein there is apparently less path over-shortening than with the Siemens approach. The re-definition of effective cost slope is as follows

(1) $$C(ij) = \frac{C_{ij} - \sum C^{ij}}{n_{ij}}$$

where

$C(ij)$ = Effective cost slope for activity i — j ,

C_{ij} = Cost slope of activity i — j ,

ΣC^{ij} = The sum of the cost slopes of all the activities which have been shortened and are contained *exclusively* by the adequately shortened paths containing activity i — j ,

n_{ij} = Number of paths requiring shortening which contain activity i — j

The steps are as follows.

1. Develop the project network.
2. Identify all the alternate paths through the network and determine the expected completion time for each of these paths.
3. Determine the project completion time.
4. Determine the amount each alternate path in the network must be shortened to achieve the desired project completion time.
5. Estimate the cost slope for each activity in the network as well as the maximum amount that each activity can be shortened.

6. Calculate the effective cost slope for each activity by adopting the following procedure.

   (a) Determine the number of paths which have not been shortened adequately and contain the activity $i — j$.

   (b) Among the adequately shortened paths containing activity $i — j$, determine the sum of the cost slopes of all the activities which have been shortened and are contained *exclusively* by the adequately shortened paths containing activity $i — j$. The result of this sum is ΣC^{ij} for activity $i — j$.

   (c) Subtract ΣC^{ij} from the cost slope, C_{ij},, of activity $i — j$, and divide it by n_{ij} the number of inadequately shortened paths. The result is the effective cost slope, *C(ij),* for activity $i — j$.

7. Select the activity with the lowest value of effective cost slope. In case of a tie do one of the following.

   (a) Give preference to the activity which is common to the greatest number of paths inadequately shortened.

   (b) If a choice of more than one activity exists then discriminate in favour of the activity which will result in the greatest amount of de-shortening of those activities which are exclusively contained in the paths adequately shortened and containing activity $i — j$.

   (c) If a choice exists after applying the above rules then give preference to the activity which gives maximum amount of shortening at the lowest value of the effective cost slope.

8. Allocate maximum possible time to the activity chosen in step (7). The amount by which an activity can be shortened at any stage is limited by:

   (a) the unallocated time remaining for the activity,

   (b) the path containing the activity which has the least demand for shortening, and

   (c) the minimum of the possible amounts of de-shortening on each adequately shortened path containing the activity. If an adequately shortened path contains the activity but there is zero de-shortening possible, then this step is inoperative.

9. Recalculate the amount of shortening required in each path, the effective cost slope for each activity and the time available for shortening the

activity (the available time will decrease for an activity which has been shortened in the current iteration but it will increase by an equal amount for all those activities which have been de-shortened in the current iteration). It is not necessary to recalculate the effective cost slope of those activities contained only by adequately shortened paths.

10. Repeat steps (7), (8) and (9) until all paths have been adequately shortened.

2.5. EXAMPLE PROBLEM

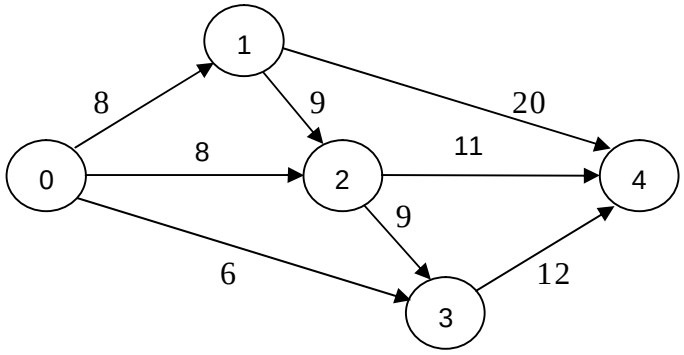

Figure 2.1: Project Network

The example given by Siemens [1] is solved in order to avoid any ambiguity. The calculations have been carried out for the case when the project completion time is required as 22.

Iteration – 1

Activity i – j	Paths requiring reduction				Cost slope C_{ij}	Time available for reduction	Effective cost slope C(ij)
	014	0124	01234	0234			
0-1	1	1	1	~~X~~	9	5	9/3 = 3
0-2	~~X~~	~~X~~	~~X~~	4	2	4	2/1 = 2
1-2	~~X~~			~~X~~	6	7	6/2 = 3
1-4		~~X~~	~~X~~	~~X~~	3	7	3/1 = 3
2-3	~~X~~	~~X~~			7	5	7/2 = 3.5
2-4	~~X~~	5	~~X~~	~~X~~	1	5	1/1 = 1
3-4	~~X~~	~~X~~			8	7	8/2 = 4
Reduction required	6	6	16	7			

Activities 2-4, 0-2, and 0-1 are shortened by 5, 4 and 1 respectively.

Iteration – 2

Activity i – j	Paths requiring reduction				Cost slope C_{ij}	Time available for reduction	Effective cost slope C(ij)
	014	0124	01234	0234			
0-1	1	1	1		9	4	(9–1)/2=4*
0-2				4	2	0	–**
1-2					6	7	(6 – 1)/1=5
1-4	5				3	7	3/1 = 3
2-3					7	5	7/2 = 3.5
2-4		5			1	0	–**
3-4					8	7	8/2 = 4
Reduction required	5	0	15	3			

* If activity 0-1 is shortened then 2-4 can be de-shortened by an equal amount up to a maximum of 4 time periods. Hence the effective cost slope will be valid for only 4 time periods only.

** No need to calculate the effective cost slope as those activities can not be further shortened.

Iteration-3

Activity i – j	Paths requiring reduction				Cost slope C_{ij}	Time available for reduction	Effective cost slope C(ij)
	014	0124	01234	0234			
0-1	1	1	1		9	4	(9–3–1)/1=5
0-2				4	2	0	–
1-2					6	7	(6 – 1)/1=5
1-4	5				3	2	–*
2-3			3	3	7	5	7/2 = 3.5
2-4		5			1	0	–
3-4					8	7	8/2 = 4
Reduction required	0	0	15	3			

* No need to calculate the effective cost slope as the path containing this activity is adequately shortened.

Iteration-4

Activity i – j	Paths requiring reduction				Cost slope C_{ij}	Time available for reduction	Effective cost slope C(ij)
	014	0124	01234	0234			
0-1	1 + 4	1 + 4	1 + 4		9	4	(9–3–1)/1=5**
0-2				4	2	0	–
1-2					6	7	(6–1)/1=5
1-4	5 – 4				3	2	–***
2-3			3	3	7	2	(7–2)/1=5
2-4		5 – 4			1	0	–***
3-4					8	7	(8–1)/6=4
Reduction required	0	0	12	0			

** An allocation in activity 0-1 is preferred over 2-3 and 1-2 because it results in greatest de-shortening of activities (a total of 8 in activities 1-4 and 2-4).

*** These activities have been de-shortened. Note the time available for reduction for these activities in iteration-5.

Iteration-5

Activity i – j	Paths requiring reduction				Cost slope C_{ij}	Time available for reduction	Effective cost slope C(ij)
	014	0124	01234	0234			
0-1	5	5	5		9	0	–
0-2				4 – 2	2	0	–
1-2					6	7	(6–1)/1=5
1-4	1				3	6	–
2-3			3 + 2	3 + 2	7	2	(7–2)/1=5
2-4		1			1	4	–
3-4					8	7	(8–2)/1=6
Reduction required	0	0	8	0			

Iteration – 6

Activity i – j	Paths requiring reduction				Cost slope C_{ij}	Time available for reduction	Effective cost slope C(ij)
	014	0124	01234	0234			
0-1	5	5	5		9	0	–
0-2				2	2	2	–
1-2		1	1		6	7	(6–1)/1=5
1-4	1				3	6	–
2-3			5	5	7	0	–
2-4		1 – 1			1	4	–
3-4					8	7	(8–2)/1=6
Reduction required	0	0	6	0			

Iteration – 7

Activity i – j	Paths requiring reduction				Cost slope C_{ij}	Time available for reduction	Effective cost slope C(ij)
	014	0124	01234	0234			
0-1	5	5	5		9	0	–
0-2				2 – 2	2	2	–
1-2		1 + 3	1 + 3		6	6	6/1=6
1-4	1				3	6	–
2-3			5	5	7	0	–
2-4					1	5	–
3-4			2	2	8	7	(8–2)/1=6
Reduction required	0	0	5	0			

All paths have been adequately shortened. The summary of the iterations are below:

Activity	Amount shortened	Cost
0 – 1	5	45
1 – 2	4	24
1 – 4	1	3
2 – 3	5	35
3 – 4	2	16
	Total	123

2.6. CONCLUSION

The same numerical problem was solved by both the algorithms with desired project completion time as 24, 26 and 28. The results obtained are identical. However, sometimes the Siemens' Algorithm failed to provide the optimal solution. Algorithm given by Goyal (1975) provides an equally good and sometimes better solution to the problem of minimizing the cost of expediting activities in order to complete a project within the desired time.

CHAPTER – 3

A SINGLE STEP CPM TIME COST TRADEOFF ALGORITHM

CHAPTER – 3

A SINGLE STEP CPM TIME COST TRADEOFF ALGORITHM

3.1. INTRODUCTION

Siemens (1971) describes a SAM algorithm for shortening the duration of a project when the expected project duration exceeds the pre-determined time limit. The objective is to minimize the total cost of expediting activities in order to complete the project in desired time and this algorithm gives the rules for choosing paths and activities which must be expedite. With this note Goyal (1975) suggested an alternate approach which is contrast to SAM, allows for de-shortening of previously shortened activity and this is accomplished by the concept of redefining the effective cost slope.

In the present algorithm the concept of effective cost has not been used to solve a time cost tradeoff problem and this is observed that it is not just a simple matter of shortening those activities with the lowest cost slopes. Some activities may be constituents of the critical path as well as one or more sub-critical paths requiring reduction. Consequently, it may actually be less costly to reduce a moderately high cost slope activity that is common to several paths requiring shortening than to reduce the lowest cost slope activity that occurs only on one of the paths that need shortening.

De-shortening an activity is to de-short an activity which has already been shortened by some amount such that it becomes less shortened, as some value from it is deducted and again added to the remaining supply of days by which that activity can be shortened. The reason of de-shortening is that if the candidate activity has the least cost slope on the path require shortening and also this candidate activity sharing the path having exclusive shortened activity. Then de-shortening is performed on the exclusively shortened activity to avoid the over-shortening of the path as some amount has to be allocated to the candidate activity.

3.2. THE STEPS OF SINGLE STEP CPM TIME COST TRADEOFF ALGORITHM:

Keeping above in view, a combination of two algorithms discussed in chapter two an algorithm has been developed. The steps of algorithm are as follows:

1. Develop the project network.
2. Identify all the alternate (parallel) paths through the network and determine the expected completion times for each of these paths. (The longest path is the critical path and determine the expected project duration).
3. Determine the desired project completion time. (This is typically an exogenous constraint imposed by considerations outside the jurisdiction of the network analyst, e.g., the government contract specifies when the project must be completed).
4. Determine the amount each alternate path in the network must be shortened to achieve the desired project completion time. (Not all paths may require shortening–one or more paths may already require less time than is available to complete the project. Whenever the expected completion time of any path is equal to or less than the desired project completion time, then obviously this path need not be shortened). The amount each path must be shortened is the amount by which the path expected time exceeds the desired project completion time, i.e., amount to be shortened=(path expected time) – (desired project completion time).
5. Estimate the cost slope (cost per unit of time saved) for each activity in the network as well as the maximum amount that each activity can be shortened.
6. Develop the time-cost matrix.
    a) Each row will represent one activity.
    b) Each column will represent an alternate path through the network. (Only those paths which need shortening should be included in the matrix).
    c) The last two columns in the matrix are used to record (1) the cost slope of each activity and (2) the maximum amount that each of the various activities can be shortened, respectively.

d) The column totals (excluding the last two columns) constitute the minimum amounts that the various paths must be shortened to meet the desired project completion date.

e) In each column, cross out those activities that are not included in (are not part of) the path represented by the column.

f) Record the cost slope for each activity in each column of the matrix.

7. Among all the columns, select the activity with (next) lowest cost slope under inadequately shortened paths. If lowest cost slope is common to more than one activity –

   a) If these activities are on different paths required shortening then select all such activities and make allocation to them as per in step 8.

   b) Discriminate in favour of the activity which is common to (i.e., is a component of) the greatest number of currently inadequately shortened paths.

   c) If a choice of more than one activity still exists, then discriminate in favour of the activity that permits the greatest amount of shortening.

      The amount that an activity may be shortened at any stage is limited by: (1) the amount of unallocated time remaining for the activity, as well as (2) the path (of which this activity is a component) that currently has the least (demand) for shortening.

   d) If the activity selection still has not been uniquely determined, discriminate in favour of the activity (within the selected column) that is common to the greatest number of paths in the matrix (adequately shortened as well as inadequately shortened paths).

8. Make the allocations in the matrix, i.e., the amount an activity will be shortened – initially, as well as at subsequent, additional allocations (to augment previous allocations, if required) – according to the following procedure:

   Allocate as much time as possible to the activity selected in Step 7 above, subject to:

   a) The current unfilled demand in any column (path) containing the selected activity (the activity being shortened). Ignore paths that have already been adequately shortened. These adequately shortened

paths may coincidentally become shortened by more than the minimum required amount.

b) The amount of time still available for shortening the activity (as determined by the original supply available in the last column minus any previous allocations made to that activity).

c) The smallest quantity of the above two criteria ((a) or (b)) determines the actual quantity of time to be allocated to the selected activity.

The effect of the procedure in Step 8 is to allocate as much time as possible to the selected activity without exceeding the available supply of the activity.

9. **I)** If there is no common activity between any two or more paths then on repeating steps 7 and 8 we get all paths adequately shortened.

II) If two or more paths have some common activities and on using all steps under this algorithm we get all paths adequately shortened (or over-shortened but critical path should not be over-shortened) and left with some supply for common activities.

Then for such common activities (in ascending order) we will make a comparison between their costs with the sum of costs of the activities on the same paths (or more paths) which have been shortened.

Find such comparison for each common activity (in ascending order) (with remaining supply) with each pair of other activities. Find out the maximum sum among the sums of these costs and perform the following step in descending order of the sum of costs.

a) If cost of common activity is less than the sum then de-short these activities (conditions apply, that no path should become inadequately shortened otherwise go for next sum (corresponding activities)) which are giving sum by the single maximum amount given by the mutual decision of supply of common activity and by the minimum of the amount other activities can be de-shortened and allocate this de-shortened amount to the common activity.

b) When cost of common activity is equal to the sum of other activities and atleast one of these activities is exclusive activity then de-short these activities by the mutual decision of supply of common activity

and by the minimum of the amount exclusive and other activities can be de-shortened.

III) On repeating the steps 7 and 8 if we reach an activity which is common to two or more paths and atleast one of these paths is inadequately shorten then de-short the exclusive activity (if any) contained by other path (containing common activity) by the mutual decision of the remaining demand of path with supply for common activity and by the amount exclusive activity can be de-shortened, and allocate the same amount to the common activity.

If there is no exclusive activity or no allocation remains on exclusive activity then do allocation as suggested in step 8.

If all path get shortened then check for step 9 (II) otherwise

(IV) Allocate the remaining supply to same activity if demand is there (here some paths can become over-shortened).

(V) Then or else go to next candidate activity and find out the case satisfying by it and choose the appropriate step of this algorithm to deal with it.

Do step 7 to 9 as per the conditions satisfied by the candidate activity and we get all paths adequately shortened with total optimal cost of shortening.

3.3. EXAMPLE PROBLEM

To perform the above discussed algorithm same example discussed in chapter two for two different algorithms have been taken to compare the results.

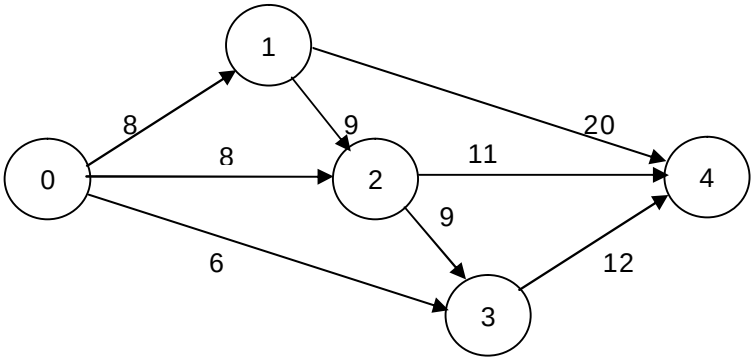

Figure 3.1: Project Network

The project network is depicted in figure 3.1. The numbers on the arcs of network diagram are the expected times of the various activities.

Total expected completion time is 38 days given by path 0-1-2-3-4. It is desired to shorten the project to 22 days (reduce by 16 days). There are six alternate paths through the network: 0-1-4; 0-1-2-4; 0-1-2-3-4; 0-2-3-4; 0-2-4; 0-3-4. Only the first four paths need to be shortened–the other two are already less than 22 days. The first four paths have expected completion times of 28, 28, 38 and 29 days, respectively, and consequently must be shortened by 6, 6, 16 and 7 days, respectively, to meet the desired completion date. The cost slope for each of the eight activities and the maximum time reduction possible for each activity is estimated to be as follows:

Activity	Cost Slope	Maximum Time Reduction
0 – 1	9	5
0 – 2	2	4
0 – 3	4	3
1 – 2	6	7
1 – 4	3	7
2 – 3	7	5
2 – 4	1	5
3 – 4	8	7

This information can now be used to develop the time-cost matrix (Step 6).

Activity	Paths Requiring Reduction				Cost Slope	Time Reduction Possible (Supply)
	014	0124	01234	0234		
0–1	9	9	9		9	5
0–2				2	2	4
1–2		6	6		6	7
1–4	3				3	7
2–3			7	7	7	5
2–4		1			1	5
3–4			8	8	8	7
Time Reduction Required (Demand)	6	6	16	7		

Figure 3.2

The time cost matrix is illustrated in figure 3.2. The cost slope is recorded in the upper portion of the box on each inadequately shortened the path that contains the activity.

Activity	Paths Requiring Reduction				Cost Slope	Time Reduction Possible (Supply)
	014	0124	01234	0234		
0–1	9	9	9		9	5
0–2				2	2	4
				4		
1–2		6	6		6	7
1–4	3				3	7
	6					
2–3			7	7	7	5
2–4		1			1	5
		5				
3–4			8	8	8	7
Time Reduction Required (Demand)	6	6	16	7		

Figure 3.3

In figure 3.3, in whole matrix least cost is 1 (step 7) so allocate 5 here (step 8). Next cost is 2 (step 7) allocate 4 (step 8). Similarly, next cost is 3 (step 7) so make an allocation 6 days here (step 8).

Activity	Paths Requiring Reduction				Cost Slope	Time Reduction Possible (Supply)
	014	0124	01234	0234		
0–1	9 2	9 2	9 2		9	5
0–2				2 4 – 4	2	4
1–2		6 5 + 2	6 5 + 2		6	7
1–4	3 6 – 2				3	7
2–3			7 4 + 1	7 4 + 1	7	5
2–4		1 5 – 5			1	5
3–4			8 2	8 2	8	7
Time Reduction Required (Demand	6	6	16	7		

Figure 3.4

From figure 3.4 it is clear that next cost is 6 so de-short activity (2, 4) by 5 (step 9 (III)) and allocate same amount to activity (1, 2) and also allocate 2 days more to it (step 9 (IV)). Next cost is 7, de-short activity (0, 2) by 4 (step 9 (III)) and allocate same amount of 4 days to activity (2, 3) plus 1 day more (step 9 (IV)).

Next cost is 8 (Step 9 (V)) and allocate 2 days to it (Step 8).

Now, all paths get shortened except path 0-1-2-3-4. Path 0-1-2-3-4 requires 2 days more so next candidate activity is (0, 1) so make allocation 2 days there by de-short activity (1, 4) by 2 days (step 9 (III)).

Activity	Paths Requiring Reduction				Cost Slope	Time Reduction Possible (Supply)
	014	0124	01234	0234		
0–1	9 2 + 3	9 2 + 3	9 2 + 3		9	5
0–2				2 4 – 4	2	4
1–2		6 5 + 2 – 3	6 5 + 2 – 3		6	7
1–4	3 6 – 2 – 3				3	7
2–3			7 4 + 1	7 4 + 1	7	5
2–4		1 5 – 5			1	5
3–4			8 2	8 2	8	7
Time Reduction Required (Demand	6	6	16	7		

Figure 3.5

From figure 3.5 it is observed that common activities are (1, 2), (2, 3), (3, 4) and (0, 1) (step 9 (II)) but supply is available is only for (3, 4) and (0, 1). For (3, 4) a comparison of its cost with sum of costs of (2, 3) and (0, 2) (both contained by same paths containing activity (3, 4)) cannot be made because activities (0, 2) is not shortened at all (step 9 (II)) and clearly cost slope (2, 3) < cost slope (3, 4).

So go to activity (0, 1) common to first 3 paths. Here,

Slope of (1, 2) + slope of (1, 4) = slope of (0, 1)

6 + 3 = 9 (step 9 (II-b))

Perform it later first check for step 9 (II-a)

Slope of (1, 4) + slope of (2, 3) > slope of (0, 1)

$$10 = 3 + 7 > 9$$

Also, slope of (1, 4) + slope of (3, 4) > slope of (0, 1)

$$11 = 3 + 8 > 9$$

Clearly, 11 > 10 so maximum sum = 11 (step 9 (II)) but here activity (3, 4) cannot be de-shortened because if this is done then path 0–2–3–4 will become inadequately shortened as activity (0, 1) is not a member of this path to which de-shortened amount is going to allocate. So step 9 (II-a) is not satisfied (it should be very clear that while applying step 9 (II) no path become inadequately shortened and also 9 (II-b) must perform (if satisfied) after 9 (II-a) has been performed).

Earlier it is found that step 9 (II-b) is satisfied. Clearly, activities (1, 4) and (1, 2) can be de-short by 3 days because an allocation of just 3 days more can be made to the activity (0, 1) as its total supply is of 5 days so by 9 (II-b) perform the steps in figure 3.5 and with this step we reached the optimal result. Hence, the final matrix is shown in figure 3.6:

Activity	**Paths Requiring Reduction**				**Cost Slope**	**Time Reduction Possible (Supply)**
	014	**0124**	**01234**	**0234**		
0–1	9 5	9 5	9 5		9	5
0–2				2	2	4
1–2		6 4	6 4		6	7
1–4	3 1				3	7
2–3			7 5	7 5	7	5
2–4		1			1	5
3–4			8 2	8 2	8	7
Time Reduction Required (Demand)	6	6	16	7		

Figure 3.6

Result to the example problem is summarized below:

Project Duration

Activity	22 days	
	Amount Shortened	**Cost**
0–1	5	45
0–2	0	0
0–3	0	0
1–2	4	24
1–4	1	3
2–3	5	35
2–4	0	0
3–4	2	16
Totals:	17 days	$123

During the solution of the problem a number of times matrix is shown just because the reader can get the clarity of steps of discussed algorithm. Otherwise the whole solution can be done in a single matrix. It is an iteration free method and calculations during the solution for the selection of activities herein not required hand computation, just a glance over all the values is enough to compare the costs of different activities.

3.4. CONCLUSION

The present algorithm is a combination of the algorithms discussed in chapter two. The rules for this algorithm were developed as a result of intuitive logic, analysis and same example was solved as considered by Siemens (1971) and Goyal (1975) with desired project completion time as 24, 26 and 28. The results obtained are identical to the results obtained by applying Siemens' and Goyal's algorithms. A large number of other examples were solved by all three algorithm and it is observed that the algorithm presented in this chapter provide the optimal solution to all these network problems. The algorithm is single step, simple and straight forward. This was also the case for other examples solved by this algorithm.

BIBLIOGRAPHY

BIBLIOGRAPHY

Ahuja, R.K., Batra, J.L. & Gupta, S.K. (1984), "A parametric algorithm for convex cost network flow and related problems," European Journal of Operational Research, Vol.16, No.2, p.222.

Atmanand (2002), "Cost Analysis," Managerial Economics, Edition 2nd, pp.161-190.

Battersby, Albert (1964), "Network Analysis for Planning and Scheduling," St. Martins Press, Inc., New York.

Baweja, Satinder Singh (2006), "CPM Schedules – Why and How," AACE International Transactions, pp.22.1-22.5.

Falk, James E. & Horowitz, Joel L. (1972), "Critical Path Problems With Concave Cost-Time Curves," Management Science, Vol.19, pp.446-455.

Freier, Keith (2000), "CPM Scheduling – Getting Back to the Basics," Cost Engineering, Vol.42, p.7.

Fulkerson, D.R. (1961), "A Network Flow Computation for Project Cost Curves," Management Science, Vol.7, No.2, pp.167-178.

Goyal S.K. (1975), "A Simple CPM Time-Cost Tradeoff Algorithm," Management Science, Vol.21, No.6.

Hillier, Frederick S., Lie Berman and Gerald, J. (2007), "Network Model For Optimization of Time-Cost Tradeoff," Operations Research, McGraw-Hill Companies New York, Edition 8th, pp.414-426

Kanti Swarup, Gupta P.K. & Man Mohan (2004), "Network Scheduling", Operations Research, Edition 12th, pp.459-465.

Kasana H.S. & Kumar K.D. (2003), "Project Management", Operations Research, pp.277-293.

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

Law, C.E. (1966), "Network Analysis for Planning and Scheduling," Operations Research, Vol.14, p.543.

Levy, Ferdinand L., Thompson, Gerald L. & Weist, Jerome D. (1963), "The ABCs of the CRITICAL PATH Method," Harvard Business Review, Vol.41, No.5, pp.98-108.

Mark, Lamendola (1998), "The Critical Path Method: What it's all about", Vol.97.

Monks, Joseph G. (1985), "Project Management", Operation Management, McGraw Hill, New York, Edition 2nd, pp.352-375.

Panagiotakopoulos, D. (1977), "A CPM Time-cost Computational Algorithm for Arbitrary Activity Cost Arbitrary Activity Cost Functions," INFOR, Vol.15, No.2, pp.183-195.

Parikh, Shailendra C. & Jewell, William S. (1965), "Decomposition of Project Networks," Management Science, Vol.11, pp.444-459.

Prager, W. (1963), "A Structural Method of Computing Project Cost Polygons," Management Science, Vol.9, No.3, pp.394-404.

Richard, J. Jean Paul (1968), "CPM as a Monitoring Tool," Production & Inventory Management, Vol.9, No.1, pp.61-66.

Saaty, Thomas L. (1971), "Network Analysis for Planning and Scheduling," Operations Research, Vol.19, pp.1544-1545.

Saindon, L.J. (1969), "CPM-An Introduction and Application," Advanced Management Journal, p.83.

Schwartz, Al (2009), "Scheduling basics: the CPM approach," Contractor Magazine, Vol.56, No.6, pp.22-56.

Sharma J.K. (2007), "Project Management", Operations Research, Edition 3rd, pp.525-596.

Siemens, N. (1971), "A Simple CPM Time-Cost Tradeoff Algorithm," Management Science, Vol. 17, No.6.

Sireesha, V., Ravi Shankar, N. (2009), "A Graphical Approach to Find the critical path and Project Network," Vol.2, No.12, pp.553-559.

Taha, Hamdy A. (2007), "Network Models", Operations Research & Introduction, Edition 8th, pp.295-305.

White, Gregory P. & Vanderembse, Mark A. (2005), "Project Management", Operations Management, John Wilay & Sons, pp.433-440.