

## Complex Project Crashing Algorithm

Nafish Sarwar Islam

(Department of Business Administration, East West University, Bangladesh)

**Abstract:** Crashing is the procedure by which project duration can be shorten up by expediting selective activities with in the project. But it requires allocating more resources than usual to compress an activity's duration, which in turns increases the budget of that activity. So, crashing is basically a time-cost trade-off by which specific deadline can be achieved. The traditional method of crashing only considers average activity times for the calculation of the critical path, ignoring the stochastic nature of activity time. This report is written to develop an algorithm for optimum crashing method to minimize the required cost while attaining a specified completion time.

**Keywords** – Critical Path, Project Crashing Algorithm, Time-Cost Trade-Off

Submitted date 12 June 2013

Accepted Date: 17 June 2013

### I. INTRODUCTION

Two of the most popular approaches to project management are the Critical Path Method (CPM) and the Project Evaluation and Review Technique (PERT), which were developed short after the World War II. The Critical Path Method (CPM) and the Project Evaluation and Review Technique (PERT) methods have been used since the 1950s to estimate the completion time of a project. CPM is a deterministic approach to calculate the duration of a project, and PERT is a probabilistic approach that enhances CPM by considering uncertainty in activity durations by calculating the probability to complete the project by a given time [1]. J.E. Kelly of Remington-Rand and M.B. Walker of DuPont developed CPM in the 1957 to assist in scheduling maintenance shutdowns of chemical processing plants. While PERT was developed shortly after by the U.S. Navy to manage the development of the Polaris missile. The original PERT Navy report (1958) does not identify the names of developers [2]. Both CPM and PERT are network based techniques, where PERT can be considered as an extension of CPM. After constructing the network diagram from precedence relationships and duration times PERT planning involves the following steps:

- (i) Estimating the time required for each activity
- (ii) Calculating slack time of each activity
- (iii) Determining the critical path(s)
- (iv) Project crashing

#### (i) Estimating the Time Required for Each Activity

A distinguishing feature of PERT is to deal with the uncertainty in activity completion times. The model includes three time estimates which are considered to follow the generalized beta distribution:

Optimistic time (a): the shortest time in which the activity can be completed.

Most likely time (m): the completion time having the highest probability.

Pessimistic time (b): the longest time that an activity might require.

The expected time for each activity can be estimated using the following weighted average:

Expected time =  $[(a + 4m + b) / 6]$ ; and the variance is: Variance =  $[(b - a) / 6]^2$

#### (ii) Calculating Slack time of Each Activity

The amount of time that a non-critical path activity can be delayed without delaying the project is referred to as slack time. As a matter of fact activities within the critical path(s) have a zero slack value. So if activities outside the critical path speed up or slow down up to their available slack time, the total project time does not change. To calculate slack time it is required to find out the following four values of each activity:

**ES** – Earliest Start time, **EF** - Earliest Finish time, **LS** - Latest Start time, **LF** - Latest Finish time.

These times are calculated using the expected time for the relevant activities. The earliest start and finish times of each activity are determined by working forward through the network and determining the earliest time at which an activity can start and finish considering its predecessor activities. The latest start and finish times are found by working backward through the network and determining the latest time that an activity

can start and finish without delaying the project considering the particular activity is predecessor of how many activities. The difference in the latest and earliest finish of each activity is that activity's slack;

$$\text{Slack Time} = (L_s - E_s).$$

### **(iii) Determining the Critical Path(s)**

The critical path is determined by adding the times for the activities in each sequence and determining the longest path in the project. That is why critical path determines the total completion time required for the project. For instance, if a project network diagram has 14 available paths, and by adding the times for the activities in path number 1, 4 and 7 gives the highest value compared to others, then that value would be the project completion time, while path number 1, 4 and 7 will be the critical paths. Now say "C" is an activity which is not available in critical paths, but available in path number 3, 6, 8 and 11 and among these four paths path number 11 gives maximum value if we add the times for the activities in path number 3, 6, 8 and 11. So, slack time of activity "C" can be calculated by subtracting the total duration of path number 11 from the total duration of any critical path 1, 4 or, 7; which is the project duration. Thus project's critical path(s) can be identified by considering the path(s) through the network in which none of the activities have slack.

### **(iv) Project Crashing**

The variance in the project completion time can be calculated by summing the variances in the completion times of the activities in the critical path. Given this variance, one can calculate the probability that the project will be completed by a certain date assuming a normal probability distribution for the critical path. The normal distribution assumption holds if the number of activities in the path is large enough for the central limit theorem to be applied. So, there would be 50% probability of completing a project if desired time and expected completion times are equal to each other. Since the critical path determines the completion date of the project, the project can be accelerated by adding the resources required to decrease the time for the activities in the critical path. Such a shortening of the project is referred as project crashing. Therefore, crashing is spending more money on the project in order to speed up accomplishment of scheduled activities [3]. Since crashing a project requires additional costs, crashing decisions need to be made in a cost-effective way [4]. Several simulation based crashing methods are developed in past two decades. These methods are heuristics that are developed to return satisfactory solutions but not necessarily an optimal solution. This research endeavored to generate an algorithm for optimum crashing method to minimize the required cost while meeting a specified completion time.

## **II. LITERATURE REVIEW**

Van Slyke (1963) was the first of many researchers to apply Monte Carlo simulations to study PERT [5]. Pritsker and Happ (1966) developed a modification of PERT called the Graphical Evaluation and Review Technique (GERT) [6]. Kennedy and Thrall (1976) developed a modification of GERT called Project Length Analysis and Evaluation Technique (PLANET) [7]. Ameen (1987) developed Computer Assisted PERT Simulation (CAPERTSIM) [8]. Badiru (1991) reported development of another simulation program for project management, called STARC [9]. Bissiri and Dunbar (1999) suggested the use of simulation to obtain the average time of each activity, the critical path, and the near critical paths to crash a project network. A near critical path in their model is a path which length is smaller than the original completion date but it is larger than the target completion date after crashing. After the path information is collected a linear program is applied to determine the crashing strategy [10]. Haga (1998) [11] along with Haga and O'keffe (2001) [2], also Haga and Marold (2004) [12], and Haga and Marold (2005) [13] worked with heuristic crashing methods for project management utilizing simulation. Haga and Marold (2004), propose a simulation-based method that deals with the time-cost trade-off involved with crashing a project. The method that they proposed is a two steps approach. The first step is to apply the traditional PERT method to crash the project, and the second step consists in testing each activity that had not been crashed to the upper crashing limit to determine if crashing that activity further reduces the average total cost of the project. Haga and Marold (2005) developed a method to monitor and control a project. The output of this method is a list of crashing points at which the project should be reviewed to decide if activities need to be crashed. Crashing points are determined by a backward run through the project network. The crashing points are established at the beginning of the project and they remain fixed during the entire project life.

## **III. PROJECT CRASHING ALGORITHM**

The project crashing algorithm described in this section is designed to give the mathematically correct time-cost trade-off curve for small project networks which are to be solved by hand. The algorithm proceeds from a network description of the project and the normal and crash times and durations for the project activities to a set of tables which enable the time-cost trade-off graph to be plotted. The three tables are the project activity marginal cost list with available compression, a path list for the project network, and a breakpoint table for the

steps in the project crashing process. These will be developed for an illustrative numerical example after a general statement of the algorithm is given.

**GENERAL ALGORITHM SPECIFICATION**

**Step1.** Each activity is assumed to have a known Normal cost if completed in a Normal time, and a (larger) Crash Cost if completed in a (shorter) Crash time. Compute the marginal crashing costs (i.e. cost per unit time) for each activity according to the following formula:

$$\text{Marginal Cost} = (\text{change in cost})/(\text{change in time}) = (\text{Crash cost} - \text{Normal cost})/(\text{Normal time} - \text{Crash time})$$

Place these marginal costs in the first column of the project activity list. Place the number of time periods of crashing availability in the second column of the table.

**Step2.** Enumerate all the paths through the project network, and list them with their normal time durations in the path list. Identify the critical path(s) as those with longest duration, and mark the critical activities as such in the third column of the project activity list started in Step1.

**Step3.** Identify the normal project duration, the normal project cost and the normal critical path in the first row of the breakpoint table, labeled as iteration 0.

**Step4.** Select that subset of critical activities which, when compressed in parallel, enable all current critical paths to become shorter, and do so at the least group marginal cost, where the group marginal cost for a subset of critical activities is the sum of the marginal costs for activities in the group.

**Step5.** Compress the selected critical activities until one or both of the following two conditions occur: (i) one (or more) of the compressed activities becomes fully crashed (i.e. is reduced to crash time); or (ii) a new path becomes critical.

**Step6.** Record the selected activities, number of time periods compressed, the new project duration, the group marginal cost for the selected activities, the added cost resulting from the compression, the new total direct cost, and the new critical path (if any) as items in the breakpoint table for this iteration. Update the compression availabilities and the path list to reflect the reduction in path lengths resulting from the selected compression.

**Step7.** Repeat steps 4 through 6 until all activities on some (any) critical path become fully crashed. At this point the breakpoint table is complete, as no further time reduction is possible. Plot the time-cost trade-off graph by linear interpolation between the time/cost pairs which occur in each row of the breakpoint table.

**Algorithm Summary**

- I. Calculate cost slope of each activity of the project
- II. Identify critical path and find the expected duration of the project
- III. Select among the activities on critical path an activity which has least cost slope. For more than one critical path, select one activity on each critical path such that total cost of crashing all these activities is minimum among all such combinations of activities
- IV. Reduce the activity time of the selected activity progressively until either crashed time is reached or the previous non-critical parallel path becomes critical
- V. Proceed above steps until there is at least one critical path on which none of the activities can be further crashed

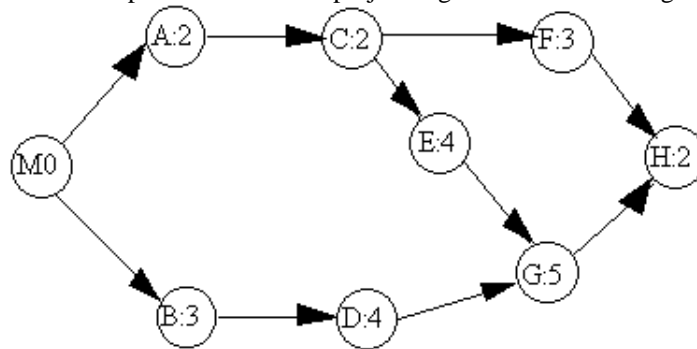
**NUMERICAL EXAMPLE**

The activities involved in the project, the precedence relations between them, and the activity durations in weeks are shown in the following table:

**TABLE 1: Activities and Immediate Predecessors**

Activity	Description	Predecessor	Duration (weeks)
A	Build internal components	None	2
B	Modify roof and floor	None	3
C	Construct collection stack	A	2
D	Pour concrete and install frame	B	4
E	Build high-temperature burner	C	4
F	Install control system	C	3
G	Install air pollution device	D,E	5
H	Inspection and testing	F,G	2

The network representation of the project is given in the following figure:



**FIGURE 1: Project's Network Diagram**

An analysis of early times shows that the duration of the project given the normal activity times is 15 weeks, which is 3 weeks longer than has been allotted for the project. Hence, the project time must be reduced, but the time reduction must be accomplished at least cost. Therefore, a tabulation of the normal and crash times and costs for each activity is compiled from time and cost estimates, with the following result:

**TABLE 2: Normal and Crash Data**

Activity	Times (weeks)		Cost (\$)	
	Normal	Crash	Normal	Crash
A	2	1	22,000	22,900
B	3	1	30,000	34,000
C	2	1	26,000	26,800
D	4	3	48,000	49,000
E	4	2	56,000	57,400
F	3	2	30,000	30,500
G	5	2	80,000	86,000
H	2	1	16,000	19,000

The first step in the procedure is to compute the marginal crash costs and the available compression for each project activity in the project activity table, as shown below.

ACTIVITY	CRASH PER WEEK COST	AVAILABLE COMPRESSION	CRITICAL OR, NOT
A	\$900	1	Y
B	2000	2	N
C	800	1	Y
D	1000	1	N
E	700	2	Y
F	500	1	N
G	2000	3	Y
H	3000	1	Y

The second step is to enumerate all the possible paths through the project network and list them in the path list for the project. (Recall that a path must respect the direction indicated on each directed arc, so that B-D-E-F-H, for example, would not be an admissible path). The path lengths using normal times are listed in the first column.

A-C-F-H	9
A-C-E-G-H	15
B-D-G-H	14

Examination of the marginal costs for the activities on the critical path A-C-E-G-H reveals that the minimum marginal cost occurs for activity E at \$700 per week. We see that there are two weeks of compression available for activity E. However, we must also check the path list to see if any new paths become critical as we compress activity E. Since activity E only occurs on the second path, the first and third path stay at constant duration as E is compressed. This means that the third path, B-D-G-H, becomes critical after E is crashed only one week. Thus the first breakpoint recorded in the breakpoint table occurs after activity E is compressed just one week.

Iter.	Activities Crashed	Crashed by	Project Duration	Group M/C	Added Cost	Total Costs	New Critical Path
0	-	-	15	-	-	308,000	A-C-E-G-H
1	E	1	14	700	700	308,700	B-D-G-H

Update the compression available for E to 1. Updating the path list accordingly yields:

A-C-F-H	9	9
A-C-E-G-H	15	14
B-D-G-H	14	14

Returning now to step 4, we seek the least cost combination of activities which, when compressed in parallel, will enable both critical paths to become shorter. Now we have a choice, since the two critical paths overlap on the G-H part of the paths. We can either crash one activity, G or H, which is common to both CP-s, or we can crash two activities in parallel, one from the A-C-E branch and one from the B-D branch. Comparing the marginal costs for G and H, we find that of these two, G would have the lesser marginal cost at \$2,000 per week. Along the A-C-E branch, E has the least marginal cost at \$700 per week. Along the B-D branch, D has the least marginal cost at \$1,000 per week. Thus the D&E combination has a group marginal cost of only \$1,700 in comparison with \$2,000 per week for G. Thus our next step is to compress D&E together. Since there is only one week of compression available on D and on E, the next step is to crash D&E together by one week.

Iter.	Activities Crashed	Crashed by	Project Duration	Group M/C	Added Cost	Total Costs	New Critical Path
0	-	-	15	-	-	308,000	A-C-E-G-H
1	E	1	14	700	700	308,700	B-D-G-H
2	D&E	1	13	1700	1700	310,400	

Reduce the availability of D and E to zero. Updating the path list accordingly results in:

A-C-F-H	9	9	9
A-C-E-G-H	15	14	13
B-D-G-H	14	14	13

Since both D and E are fully crashed, the least cost activity from the A-C-E branch is C at \$800 per week, and the least cost activity from the B-D branch is B at \$2,000, so the C&B combination at \$2,800 is the best that can be done using two activities in parallel. Since this is more than the \$2,000 per week cost of activity G, our next step is to crash G. And since G is on both critical paths, we can crash it the full three weeks that are available for this activity. The breakpoint table then becomes as follows.

Iter.	Activities Crashed	Crashed by	Project Duration	Group M/C	Added Cost	Total Costs	New Critical Path
0	-	-	15	-	-	308,000	A-C-E-G-H
1	E	1	14	700	700	308,700	B-D-G-H
2	D&E	1	13	1700	1700	310,400	
3	G	3	10	2000	6000	316,400	

Reduce the availability of G to zero. Updating the path list yields:

A-C-F-H	9	9	9	9
A-C-E-G-H	15	14	13	10
B-D-G-H	14	14	13	10

Since G is now fully crashed, activity H is the only one activity which will cause both critical paths to become shorter, and it costs \$3,000 per week. Thus the C&B combination is the least cost available at this point, so the next step is to crash B and C in parallel. In this case all three paths get shorter together, so no new critical paths occur. However, C has only one week of compression available, so the C&B combination can be compressed only one week. Thus the next breakpoint table is as follows.

Iter.	Activities Crashed	Crashed by	Project Duration	Group M/C	Added Cost	Total Costs	New Critical Path
0	-	-	15	-	-	308,000	A-C-E-G-H
1	E	1	14	700	700	308,700	B-D-G-H
2	D&E	1	13	1700	1700	310,400	
3	G	3	10	2000	6000	316,400	
4	B&C	1	9	2800	2800	319,200	

Reduce the availability of B to 1 and C to zero. The updated path list becomes:

A-C-F-H	9	9	9	9	8
A-C-E-G-H	15	14	13	10	9
B-D-G-H	14	14	13	10	9

At this point A and B are the only two activities on the upper and lower branches with remaining compression availability, and their combined cost is \$2,900 which is still less than the \$3,000 for activity H and since they each have only 1 week of compression remaining, the next step is to crash A&B together by a week.

Iter.	Activities Crashed	Crashed By	Project Duration	Group M/C	Added Cost	Total Cost	New Critical Path
0	-	-	15	-	-	308,000	A-C-E-G-H
1	E	1	14	700	700	308,700	B-D-G-H
2	D&E	1	13	1700	1700	310,400	
3	G	3	10	2000	6000	316,400	
4	B&C	1	9	2800	2800	319,200	
5	A&B	1	8	2900	2900	322,100	

The availability of A and B are reduced to zero. The updated path list becomes:

A-C-F-H	9	9	9	9	8	7
A-C-E-G-H	15	14	13	10	9	8
B-D-G-H	14	14	13	10	9	8

At this point H is the only activity left to shorten the last two paths, and only one week of compression is available, so the final step in the breakpoint table is to compress H by one week.

Iter.	Activities Crashed	Crashed by	Project Duration	Group M/C	Added Cost	Total Costs	New Critical Path
0	-	-	15	-	-	308,000	A-C-E-G-H
1	E	1	14	700	700	308,700	B-D-G-H
2	D&E	1	13	1700	1700	310,400	
3	G	3	10	2000	6000	316,400	
4	B&C	1	9	2800	2800	319,200	
5	A&B	1	8	2900	2900	322,100	
6	H	1	7	3000	3000	325,100	

The final path list is therefore:

A-C-F-H	9	9	9	9	8	7	6
A-C-E-G-H	15	14	13	10	9	8	7
B-D-G-H	14	14	13	10	9	8	7

Since both critical paths are now fully crashed, the project crashing process is complete. The optimal time-cost trade-off values are contained in the project duration and project cost columns of the final breakpoint table. The final step of the process is to graph these points and connect them with straight line segments, as shown on Figure 2 [given on next page].

The only remaining task is to interpolate for the project cost corresponding to the 12 week project duration. Since this is just one week less than the 13 week solution in the breakpoint table, activity G will be crashed only one week rather than three, so the project cost for the 12 week schedule will be \$310,400 plus

\$2,000 or \$312,400. This result can be computed directly from the final breakpoint table, but the shape of the complete trade-off curve is of interest as well, and is more easily seen from the chart.

#### IV. CONCLUSION

In this paper the algorithm proposed for unit crashing reduces the cost of project. When activities are crashed by one day then only the crashing cost corresponding to one day is increased thereby reducing the project duration as well as cost. The purpose of this study was to explore the suitability of a new method to crash PERT networks. This approach is well suitable for places where cost is of major consideration. Plans for future research include running the simulation on additional networks, preferably real-world projects.

Time-Cost Trade-Off Curve

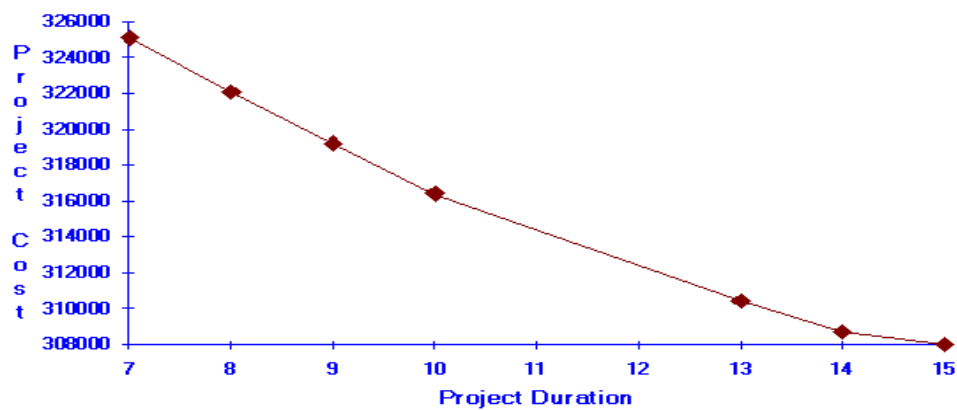


FIGURE 2: Time-Cost Trade-Off Diagram

#### REFERENCES

- [1] Lee, D., and Arditi, D. (2006) Automated statistical analysis in stochastic project scheduling simulation. *Journal of Construction Engineering and Management*, 132(3): 268-277.
- [2] Haga, W. A. & Tim O'keefe. (July 14, 2001) *Crashing Pert Networks: A Simulation Approach*. 4th International conference of the Academy of Business and Administrative Sciences Conference Quebec City, Canada.
- [3] Rosenau, M. D. and G. D. Githens. (2005) *Successful Project Management: a Step-by-step Approach with Practical Examples*. 4th Ed., Wiley, Hoboken, N.J.
- [4] Michael E. Kuhl and Radhamés A. Tolentino-Peña. (2008) *A Dynamic Crashing Method For Project Management Using Simulation-Based Optimization*. Winter Simulation Conference, S. J. Mason, R. R. Hill, L. Mönch, O. Rose, T. Jefferson, J. W. Fowler eds.
- [5] Van Slyke, R. M. (1963). Monte Carlo methods and the PERT problem; *Operations Research*, 13, 141-143.
- [6] Pritsker, A. B. & Happ, W. W. (1966). GERT: Graphical evaluation and review technique, Part-I, *Fundamental Journal of Industrial Engineering*, 17(5), 267-274.
- [7] Kennedy, K. W. & Thrall, R. M. (1976). PLANET: A Simulation approach to PERT; *Computers and Operations Research*, 3, 313-325.
- [8] Ameen, D. A. (April 1987). A computer assisted PERT simulation. *Journal of Systems Management*, 6-9.
- [9] Badiru, A. B. (1991). A simulation approach to network analysis. *Simulation*, 57, (4), 245-255.
- [10] Bissiri, Y., and S. Dunbar. (1999). Resource allocation model for a fast-tracked project. *International Conference on Intelligent Processing and Manufacturing of Materials*, 635-640.
- [11] Haga, W. A. (1998). *Crashing PERT networks*; Ph.D. Dissertation, University of Northern Colorado, Colorado.
- [12] Haga, W. A., and K. A. Marold. (2004). A simulation approach to the PERT-CPM time-cost trade-off problem. *Project Management Journal*, 35(2): 31-37.
- [13] Haga, W. A., and K. A. Marold. (2005). Monitoring and control of PERT networks. *The Business Review*, 3(2): 240-245.