International Journal of Science, Environment and Technology, Vol. 3, No 2, 2014, 374 – 394
ISSN 2278-3687 (O)

# CRASH: AN AUTOMATED TOOL FOR SCHEDULE CRASHING

**Najib Georges[1], Nabil Semaan[2], and Joe Rizk[3]**
[1]Associate Professor, [2]Assitant Professor, [3]Senior Lecturer,
Civil Engineering Department, Faculty of Engineering, University of Balamand,
Koura District, Lebanon

**Abstract:** According to Goal Group fast facts, nearly USD10 trillion was funneled to projects, in 2010. However in PricewaterhouseCoopers survey about Mega-Projects in the Arabian Gulf area, 80% of respondents said that their projects are delayed, wrote Ed Attwood from "ArabianBusiness.com". Crashing a schedule in project management is an important tool, especially in time and cost management. Traditional methods are time consuming, and prone to human error. This paper's main objective is to present a developed automated tool model entitled "CRASH" that performs generic crashing of a schedule in a short period of time and without risk of errors. "CRASH" uses process decomposing and addresses the programmability of each operation. It is applied to schedule crashing problems and results show 90% to 95% reduction in time. This automated tool is relevant to the industry experts and academics, since it provides a new model that solves the problem of crashing in project schedules.
**Keywords:** Schedule Crashing, Cost Reduction; Critical Path Method.

## 1. INTRODUCTION

Goal Group stated in their fast facts that nearly USD10 trillion was funneled to projects, in 2010. Projects, run by one or many project managers, need to have a definite scope, budget, and time schedule. On the words of Sayles, "Project managers function as bandleaders who pull together their players each a specialist with individual score and internal rhythm. Under the leader's direction, they all respond to the same beat" (Sayles, 2011). Bringing to action those specialists necessitates preparing a "plan of work". In the planning phase, the project manager defines elements of work, commonly referred to as tasks or activities, having defined duration and relationships. Subsequently, he assembles these activities in an implementation structure, called schedule. This schedule serves as the baseline to estimate the total project duration, assign resources, and track and control the work progress.

A common method for scheduling is the critical path method (CPM). This method combines different activities according to their precedence relationships and dependencies. Each activity has predecessors and successors, along with different dependencies like Finish to Start, Start to Start and Finish to Finish with lead or lag time for each. The total project

*Received Jan 20, 2014 * Published April 2, 2014 * www.ijset.net*

duration is defined by the critical paths, which are the paths where accumulation of tasks durations is the longest (Levine, 2002).

Nonetheless, most project schedules may be affected by uncertainties, in other words, identified and unidentified risks. Risks may alter the schedule in different ways, among which are imposing new deadlines or work delays. In their fast facts, Goal Group also publishes that in 2009, 44% of IT projects were challenged. They explain that by challenged they mean that the projects were late, over budget, or did not meet their scope. Accelerating a late project is therefore an important aspect in project management. Two of the possible solutions are fast-tracking the works and crashing the schedule, or even a combination of the two. Given the uniqueness of projects, the solutions are thus diverse and must be creative.

When fast-tracking, the project manager alters task dependencies in a risky manner, reduces lag times, and splits lengthy tasks into smaller ones with the intention of congesting more work into a shorter period.

The other technique is called activity crashing. This technique assumes that a task time can be downsized by increasing the resources of this task, and subsequently increasing its cost. Hence, to crash a schedule, means to crash particular activities, resulting in an overall reduced project duration but increased project cost.

The problem in these methods is that they are time consuming and prone to human error, since it is a manual process, thus rendering it impractical. This paper's main objective is to develop an automated tool that:

(1) Takes a very short period of time to solve a crashing problem,

(2) Reduces the human error in the found solution.

## 2. LITTERATURE REVIEW

Since the invention of the CPM in 1959 by the DuPont Corporation, and simultaneously the development of the Program Evaluation and Review Technique PERT, a technique similar to the CPM, in 1958 by the U.S. Navy, hundreds of research papers have been published on the subject of time and cost control of the projects. Particularly, a limited number of those papers were focused on schedule crashing.

Lima et al., published in 2006, in the Third International Conference on Production Research – Americas' Region 2006 (ICPR-AM06), a research where they compared the use of three schedule crashing methods. The first is the brute force method, developed by Casarotto Filho et al. (1992), the second is the Linear Programming model, and the third is the traditional method used in this paper. By comparison of examples crashed by those methods, the authors

found out that the traditional method resulted in the least increase in the total project direct cost.

In 2010, Brunnhoeffer et al., from Roger Williams University, developed an algorithm that can be used in Microsoft Excel. The algorithm has nine steps. First, determine all the possible paths through the CPM diagram, then determine the duration of those paths. Afterwards, compute the cost to crash each task, and select the lowest cost that will affect the project duration and modify the project cost. This basic method can be re-run each time. In this manner, he developed an excel spreadsheet, where all those calculations are done. The authors concluded that this method is better than manual calculations and will eliminate the human error, once the number of critical paths increases. Moreover, they stated that it will be helpful if the possible paths were determined a priori.

Later on, in 2011, in the 47th Associated Schools of Construction Annual International Conference Proceedings, Celik et al. published a research paper entitled "Toward a Teaching Software Application for Crashing the Schedule: SPE Beta v.1". This paper, introduced a new software, SPE Beta v.1, developed based on Brunnhoeffer paper. The software requires the user to define a text file where the name of the activity and its predecessors are entered. Then, it will generate all the possible paths for the network diagram that was entered. This software developed with C++ language is still in Beta version.

Finally in 2012, Li et al. published "Paper Crashing Using Excel Solver: A Simple AON Network Approach", in the International Journal of Management & Information Systems. The authors formulated a model in Microsoft Excel, where project activities are listed with their information. Possible paths, critical paths and project durations are determined manually a priori, and using a set of constraints, the Excel Solver will determine the added cost when crashing to achieve a project deadline.

Previous research does not address the main problem of the crashing process practicality when applying it manually throughout the whole project scheduling. In both Excel spreadsheets developed previously, the file shall be altered and customized to each new project, in order to crash it. Moreover, in Li et al.'s model, the paths, critical paths and project durations shall be calculated manually and inputted rendering the process is semi-automated. Thus, the need of an automated tool, that solves the crashing problem with a short period of time and without errors, arises.

## 3. THE TRADITIONAL CRASHING METHOD

The basic and precise crashing method is the manual method, where activities are crashed on a day by day basis. This method, although resulting in correct results, is time consuming and requires repetitive work and trials. The key data values required to perform this method are four, along with the project network diagram. The project manager needs to prepare a list of the activities, with the following attributes for each activity: normal cost, normal duration, crash cost, and crash duration.

Determining the normal and crashed durations, requires the assessment of the normal rate of productivity and the maximum rate of productivity. By estimating the quantities of work, he evaluates the normal and crashed duration using the following Equations 1 and 2:

$$\text{Normal Duration} = \frac{\text{Estimated Quantity}}{\text{Normal Rate of Productivity}} \quad [1]$$

$$\text{Crashed Duration} = \frac{\text{Estimated Quantity}}{\text{Maximum Rate of Productivity}} \quad [2]$$

Likewise, the project manager needs to calculate the normal cost and crashed cost after determining the material cost, equipment cost and manpower cost. Afterwards, the slope is calculated, which is the daily increase in an activity cost using Equation 3:

$$\text{Slope} = \frac{\text{Crashed Cost - Normal Cost}}{\text{Normal Duration - Crashed Duration}} \quad [3]$$

At this stage, the project manager determines the critical paths, and selects from each path the critical activity that has the lowest slope, resulting in the lowest cost increase. Those slopes of the selected activities are then summed up, and compared to the slopes of common activities, or a combination of common activities and lowest slope activities. Thus, the activities selected to be crashed need to have the lowest possible project cost increase.

After selecting the activities to be crashed, the project manager will reduce the duration of each of those activities by one unit of time, and increase the project cost by the sum of their slopes. In that way, the project total duration will be reduced by one unit of time (i.e. one day). If the project manager wishes to reduce the project duration more, the same procedure will have to be applied again, until the desired decrease in the total project duration is achieved.

Furthermore, when crashing the project, a new set of critical paths may emerge and the crashing procedure needs to take those new critical paths into consideration.

On the other hand, when crashing an ongoing project schedule, the project manager will follow the same instructions discussed above, but with one major difference. An ongoing schedule has completion percentages for activities, and therefore the originally schedule durations should be replaced by remaining durations, and the crash duration shall be multiplied by the remaining percentage till completion, and rounded up. The completion percentage shall be determined using the actual work performed.

## 4. CRASH: AN AUTOMATED TOOL MODEL FOR SCHEDULE CRASHING

Converting the manual method into a programmable algorithm requires analyzing the steps needed for crashing the schedule, and determining its programmability. If a certain step cannot be programmed, an indirect route must be found. Figure 1 illustrates the flow chart of the automated tool.

First, all the needed data must be acquired, like the name of the activities, their successors, their budgeted and actual (if applicable) quantities, their normal and maximum rate of productivity, their different material, equipment and manpower costs. This step is programmable by creating a list arrays (or equivalent), and listing the attributes of each activity in each array. The list will then be the list of activities.

Afterwards, the CPM will be applied and the possible paths will be listed. This method solved graphically in the manual method, can create a programming difficulty. Methods to overpass this difficulty shall be discussed in the next part.

At this time, the duration of each path should be calculated, by summing up the duration of the activities present in this path. Critical paths are determined, which are the paths having the longest duration.

Now, a manual trick appears and it cannot be programmed as is. Manually, the project manager will determine the activities with the lowest slopes in each critical path, and will compare those to a combination of those activities with the common activities, making sure not to crash more than one activity per path, thus determining directly which activities to crash. It is almost impossible to write an algorithm that will get this combination directly.

Therefore, one solution might be looking at all the possible combinations and choosing the combination with the lowest sum of slopes. So in brief, the main task will be finding all the possible combinations from the sets. This process is known as the Cartesian product or Direct product.

According to Weisstein, from Wolfram MathWorld, the Cartesian product of two sets, denoted A and B, is a set of points (a, b) where a is part of A and b is part of B. Thus, the

combinations needed, are the sets (a, b) and the critical paths (if two) are the sets A and B. Considering N sets, $N_1$, $N_2$, $N_3$… $N_N$, having each a dimension of $n_1$, $n_2$, $n_3$…$n_N$, the number of generated sets from the Cartesian products of those sets will be $n_1*n_2*n_3*…*n_N$, with each set containing N elements.

Therefore, in a quick straightforward calculation, if ten critical paths were present, and each path contains ten activities, the Cartesian product will generate $10^{10}$ sets, containing each 10 activities. The storage of such a list will create a difficulty, since it needs $10^{11}$ bytes, which is equivalent to 93 Gigabytes. This method will require huge computational power and large storage space.

Reducing the output of the Cartesian product shall be addressed by reducing the input. To reduce the input, three constraints must be applied. Initially, each activity in each path must be checked for sufficient crashing time. The activity must have a normal duration larger than the crashed duration, or else it cannot be crashed, and must be removed from the path inputted to the Cartesian product.

Furthermore, it can be noticed that certain activities will never be crashed, even if they are critical activities contained within in a critical path. Those activities are the ones that are not common to more than one critical path, and at the same time do not have the lowest slope in this particular critical path. Omitting those activities from the paths inputted to the Cartesian product, will furthermore reduce the amount of data generated.

Taking for example the same 10 critical paths example, it is clearly observed that the minimum number of activities left in a path must be the activity with the lowest slope. Additionally, some common activities may reside in the path. So if each path is left with 5 activities, the number of sets generated will be $5^{10}$, containing 10 activities each, which needs storage space of 45 Megabytes. Thus, it is noticeable that reducing the number of activities inside each set in half, will result in an output reduction of more than 2000 times.

This method of imposing constraints on the input of the Cartesian product will results in more feasible ways of programming this model.

Finally, once the possible combinations are determined, the sum of slopes of the activities in each combination will be calculated, and the combination with the lowest sum of slopes will be adopted. The activities within the adopted combination will then be modified in a way to reduce their normal duration by one unit of time, and increase the total project cost by the sum of their slopes.

### 4.1 Algorithm and programming of CRASH

The initial step to start programming any problem is to provide a system with clear decision making processes. For this purpose, it was decided to build the algorithm around a “Driver” function, which takes in data from the user and exports data to the user. The rest of the processes will be built-in functions programmed to interact with the main “Driver” function.

#### 4.1.1 Driver

To start, the “Driver” function shall retrieve from the user all the necessary data like the activity name, the activity successors, their budgeted and actual (if applicable) quantities, their normal and maximum rate of productivity, their different material, equipment and manpower costs. After processing those figures, calculating the normal and crashed costs, evaluating the normal and crashed durations, and interacting with the different functions to determine which activities to crash, the function shall display to the user the activities to be crashed as well as a crashed table of data. Note that the driver shall decide which durations to send to the functions to crash. If the activity has a non-zero percent complete, the durations to be sent are the remaining duration and the adjusted crashed duration, whereas if the activity has a zero percent complete, the driver will send to the functions the normal duration with the original crash duration.

#### 4.1.2 Longest Path

After collecting the necessary data, the algorithm should build the network diagram and determine all the possible paths. To do so, a step by step procedure should be followed. First, the initial assumptions would be that the network diagram should have a single start activity and end activity. To achieve that, a START milestone and FINISH milestone should be added to the list of activities.

Subsequently, to generate all the possible paths, the algorithm shall proceed as follows:

- Take in the first activity with its successors
- Write the activity as a set
- Increment each set that has this activity as the last element by the activity successors, by creating new sets
- Repeat this procedure until the Finish milestone is reached

At this point, after determining all the possible paths, the algorithm shall calculate the duration of each path by summing up the duration of activities in the path. Then, the paths with the longest duration, called critical paths will be selected and the rest will be discarded.

### 4.1.3 Pruning

To prune is to clip and trim the tree from excess branches and leaves. Similarly, the algorithm shall prune the list of critical paths from redundant activities. The redundant activities discussed earlier shall be omitted from the critical paths.

First, a check for the availability of enough crashing time should be done. The activities in the critical paths shall be checked if their crashed time is larger than the original normal duration in a new project. In an ongoing project, the activities crashed time should be multiplied by their completion percentage, and rounded up to the nearest integer, and compared to their remaining duration. If an activity does not have enough crashing time, it will be disregarded in the crashing process. To do so, a Boolean should be created, taking only 0 or 1, named B1. If the activity has enough crashing time, the Boolean B1 will be 1, if not it will be 0.

At this stage, the activities shall be checked for the lowest slope activity, by creating a Boolean B2, initialized to 1. Therefore, in each path, the algorithm will determine the activity having the lowest slope, and will keep its Boolean B2 value as 1, and change all the other B2 values to 0.

Afterwards, the activities with Boolean B2 value of 0, changed by the lowest slope check, shall be checked for common activities. The algorithm shall check if the activity is common to more than one critical path. If yes, the Boolean B2 value will change to 1, if not it will stay as 0. In that way, the common activities will be kept in the process and will get into the combination generator.

As a result of this pruning method, each activity having a Boolean value of 0 in B1 or B2 will be discarded, and the activities with both Booleans B1 and B2 set as 1 will be transferred to the Cartesian product generator.

### 4.1.4 Cartesian Product Generator

In this function, a list of sets will be sent to it. The sets contain the activities in each critical path that have passed the pruning function and hold the two Boolean numbers B1 and B2 as 1. This function shall generate all the possible combinations from those sets. To do so, the function shall be written as a recursive function. The sets will be sent to it one by one.

Initially, the function will increment an empty set by the new set. Then, the results will be incremented by the second set, and so on till the last set. Having determined all the possible combinations, those combinations will be sent to the next function to determine the activities to be crashed.

### 4.1.5 Crashing Activities

At this stage, the "Driver" function will take the combinations from the Cartesian Product Generator and send them to "Can_Crash" function. This function will sum the slope of each activity in each combination and will compare this sum, to get the combination with the lowest sum of slopes. But, when summing the slopes, if an activity is present more than once in a certain combination, it will only add its slope once.

The function will return the combination with the lowest sum of slopes to the "Driver". The "Driver" will then reduce the normal or remaining duration of the activity by one unit of time, and will add to the total project cost the sum of slopes of this combination.

## 5. THE CRASH TOOL APPLICATION AND TESTING

The crash automated tool is applied to two scheduling case studies.

### 5.1 Case Study 1

The first case study is a generic project schedule, easy and not complicated. Table 1 shows the network information of the project. Figure 2, shows the Activity On Network schedule.

The Critical Path Method is then applied and the critical paths are identified as follows:

- Critical path 1: START-A-C-G-I-FINISH , with a total project duration of 18 days,
- Critical path 2: START-A-C-H-J-FINISH , with a total project duration of 18 days,
- Critical path 3: START-B-F-FINISH, with a total project duration of 18 days.

For crashing this schedule, the critical paths are the only paths that are needed. Now, the slopes must be indicated on the network diagram along with the crashed duration of every activity. The slope is identified above the node, with the original duration under the node on the left and the maximum crashed duration under the node to the right. Figure 3 shows the schedule network with critical path and slopes identified.

Taking from each critical path the activity with the lowest slope, it is determined that the activities G, H and F are taken, with a total sum of 362+868+1841=3071$/day. But it is noticed that the activity C is common to two paths. Bearing in mind that C is considered for crashing along with activity F, their sum of slopes is determined to be 955+1841=2796$/day. Therefore, to crash this schedule by one day, the activities C and F should be crashed each by one day, and the total cost of the project will increase an amount of $2796. After this, for each additional day to be crashed, the network diagram should be revised for newly developed critical paths, and the same procedure should be done, taking into consideration all the possible combinations and calculating their sum of slopes. The lowest sum should be the

one to consider. For this case study, the combinations to be considered for maximum crashing of this schedule are listed in Table 2.

The case study schedule was loaded into the software, and crashed within seconds. Screenshots from the tools are shown in Figures 4 and 5.

### 5.2 Case Study 2

Case Study 2 project schedule has the same activities and network schedule as the first case study; however it shows an updated schedule. This updated schedule reflects how crashing is done manually and with the CRASH tool. Thus an additional column is added to the original project schedule information, the percent completion of the activities. Refer to Table 3 for the case study 2, or the updated schedule information.

After adjusting the network diagram and removing the completed activities and adjusting the original crashing time, critical paths and slope are identified on the schedule network. Refer to Figure 6 for network schedule of case study 2.

At this stage, the same procedure applied before in case study 1 is applied to this network diagram, and combinations of crashed activities with respective costs are shown in Table 4.

Case study 2 schedule is run using CRASH, and solution is found within seconds. Figures 7 and 8 show screenshots of the CRASH tool as applied to the case study 2.

### 5.3 CRASH Tool Testing

By comparing the results of the manual method and CRASH, it is observed that the crashed activities and the total added crash cost are identical. The major difference is the time required to perform the operation. Table 5 shows the time comparison, for Case study 1 & 2 presented in this paper, as well as three more case studies, between the manual and automated tool runs. Table 5 shows a reduction of 93% on average on run time for small projects, which is cost effective.

It is observed that the automated tool is faster and less prone to human error. Case study 1 schedule is however an illustrative small example. When faced with a standard project schedule, the time to perform the manual crashing will increase to several hours depending on project complexity, and may render the process unfeasible. By using CRASH, any project schedule, simple or complex, will be crashed in a matter of several minutes, which is the time needed to input the data.

## 6. CONCLUSIONS

This study develops a model for schedule crashing, and proved its validity after programming it and testing it. Using various techniques of combination mathematics, programming

optimization and in depth pruning techniques, "Crash" has an edge on the manual crashing method, by reducing the time needed to crash a schedule from hours to fractions of a second. Finally, it is recommended that this software gets integrated or used as a module in some of the major scheduling software i.e. Primavera or Microsoft Project. Thus, the full potential of the software will be utilized in complicated projects where manual crashing takes hours or days inducing thousands of dollars savings, made by crashing the correct items.

## 7. ACKNOWLEDGMENTS

The authors would like to express their gratitude and appreciation to Mr. Saeid Al Wazzan who provided support in writing the model in JAVA programming language.

## 8. REFERENCES

[1] Attwood, E. (2012, December 13). *Gulf mega projects facing overruns, delays - PwC survey*. Retrieved May 20, 2013, from http://www.arabianbusiness.com/gulf-mega-projects-facing-overruns-delays-pwc-survey-482670.html

[2] Brunnhoeffer, G.C. (2010). *Crashing the schedule – An algorithmic approach with caveats and comments.* Retrieved February 9, 2012, from http://ascpro0.ascweb.org/archives/cd/2010/paper/CEUE137002010.pdfGoal Group. (n.d.). *Fast Facts*. Retrieved May 20, 2013, from http://www.goalgroup.com.au/fastfacts/.aspx

[3] Gokhan Celik, B., Cokce Celik, S., & Brunnhoeffer, G.C. (2011, April). *Toward a teaching software application for crashing the schedule: SPE™ Beta v.1.* Symposium conducted at the 47th ASC Annual International Conference by the Associated Schools of Construction, Omaha Nebraska, USA.

[4] Heerkens, G. (2001). *Project management.* (1st ed.). New York: McGraw-Hill.

[5] Levine, H. (2002). *Practical project management – Tips, tactics, and tools.* (1st ed.). New York: John Wiley & Sons.

[6] Li, K., Shao, B., & Zelbst, P. (2012). Project Crashing Using Excel Solver: A Simple AON Network Approach. *International Journal of Management & Information Systems, Volume 16, Number 2(Second Quarter 2012)*, 177.

[7] Lima, M.B.F., Silva, L.B., & Vieira, R.J. (2006). *Project crashing and costs laws in the knowledge age.* Symposium conducted at the Third International Conference on Production Research – Americas' Region (ICPR-AM06), Curitiba, Parana, Brazil.

[8] Portny, S. (2010). *Project management for dummies.*(3rd ed.). Indiana: Wiley Publishing.

[9] Sayles, L.S. (2011). *Project management quotes*. Retrieved November 15, 2011, from http://www.projectauditors.com/Company/Project_Management_Quotes.html.

List of Tables:



List of Figures:

**Table 1**: Case Study 1 Network Schedule Information

| Activity Name | Activity Successor | Normal Duration | Normal Cost | Crashed Duration | Crashed Cost |
|---|---|---|---|---|---|
| START | A, B | 0 | 0 | 0 | 0 |
| A | C, D | 2 | 5000 | 1 | 10000 |
| B | E, F | 4 | 6080 | 2 | 12000 |
| C | G, H | 5 | 3135 | 2 | 6000 |
| D | J | 6 | 1704 | 4 | 4000 |
| E | J | 5 | 4000 | 3 | 8000 |
| F | FINISH | 14 | 6636 | 10 | 14000 |
| G | I | 6 | 876 | 4 | 1600 |
| H | J | 4 | 864 | 2 | 2600 |
| I | FINISH | 5 | 6615 | 3 | 14000 |
| J | FINISH | 7 | 11620 | 3 | 24000 |
| FINISH | - | 0 | 0 | 0 | 0 |

**Table 2:** Case Study 1 Manual Crashing Results

| Crashing Combination | Crashing Amount |
|---|---|
| C + F | 2796$/day |
| C + F | 2796$/day |
| C + B | 3915$/day |
| G + F + J | 5298 $/day |
| G + F + J | 5298 $/day |
| A +B | 7960 $/day |

**Table 3**: Case Study 2 Network Schedule Information

| Activity Name | Activity Successor | %Complete | Remaining Duration | Normal Cost | Original Crashed Duration | Adjusted Crashed Duration | Crashed Cost |
|---|---|---|---|---|---|---|---|
| START | A, B | 0 | 0 | 0 | 0 | 0 | 0 |
| A | C, D | 100 | 0 | 5000 | 1 | 0 | 10000 |
| B | E, F | 100 | 0 | 6080 | 2 | 0 | 12000 |
| C | G, H | 100 | 0 | 3135 | 2 | 0 | 6000 |
| D | J | 100 | 0 | 1704 | 4 | 0 | 4000 |
| E | J | 60 | 2 | 4000 | 3 | 1 | 8000 |
| F | FINISH | 50 | 7 | 6636 | 10 | 5 | 14000 |
| G | I | 50 | 3 | 876 | 4 | 2 | 1600 |
| H | J | 50 | 2 | 864 | 2 | 1 | 2600 |
| I | FINISH | 0 | 5 | 6615 | 3 | 3 | 14000 |
| J | FINISH | 0 | 7 | 11620 | 3 | 3 | 24000 |
| FINISH | | 0 | 0 | 0 | 0 | 0 | 0 |

**Table 4:** Case Study 2 Manual Crashing Results

| Crashing Combination | Crashing Amount |
|---|---|
| H + E | 2868$/day |
| J + G | 3457$/day |
| F + J + I | 8628$/day |

**Table 5**: Processing Time Comparison

| **Processing Time** *[minutes]* | **Case Study 1** | | **Case Study 2** | | **Case Study 3** | | **Case Study 4** | | **Case Study 5** | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Manual method | CRASH | Manual Method | CRASH | Manual Method | CRASH | Manual Method | CRASH | Manual Method | CRASH |
| Preparing/Inputting Data | 0 | 2 | 0 | 2 | 0 | 2 | 0 | 4 | 0 | 2 |
| Drawing the network diagram | 3 | 0 | 3 | 0 | 3 | 0 | 4 | 0 | 3 | 0 |
| Determining and listing the possible paths | 3 | 0 | 5 | 0 | 4 | 0 | 5 | 0 | 2 | 0 |
| Determining the paths durations | 3 | 0 | 3 | 0 | 4 | 0 | 4 | 0 | 2 | 0 |
| Determining the critical paths | 1 | 0 | 1 | 0 | 1 | 0 | 2 | 0 | 1 | 0 |
| Crashing days | 20 | 0 | 20 | 0 | 25 | 0 | 30 | 0 | 15 | 0 |
| **TOTAL TIME** | **30** | **2** | **32** | **2** | **37** | **2** | **45** | **4** | **23** | **2** |

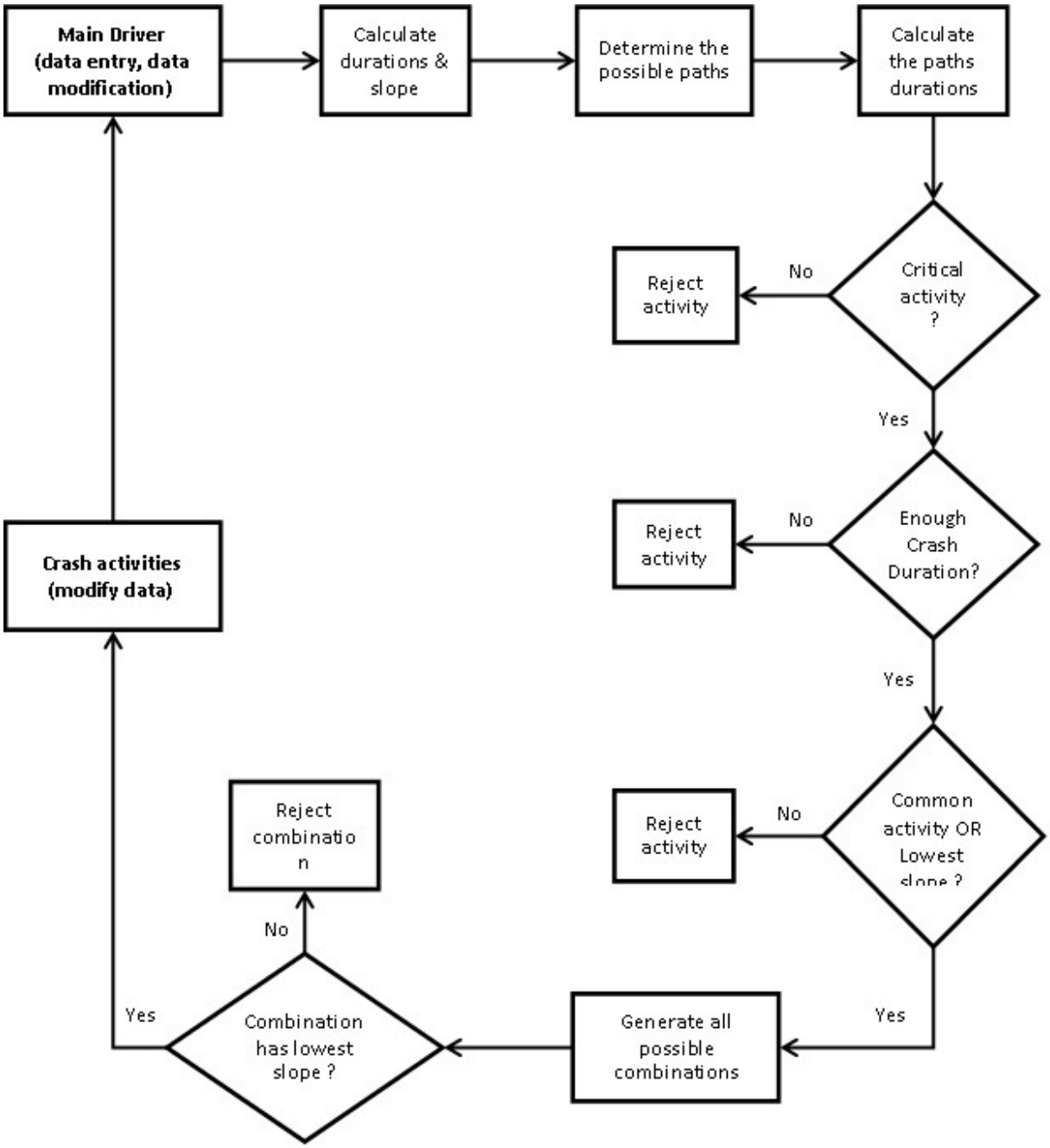


**Figure 1.** CRASH Model Flow Chart

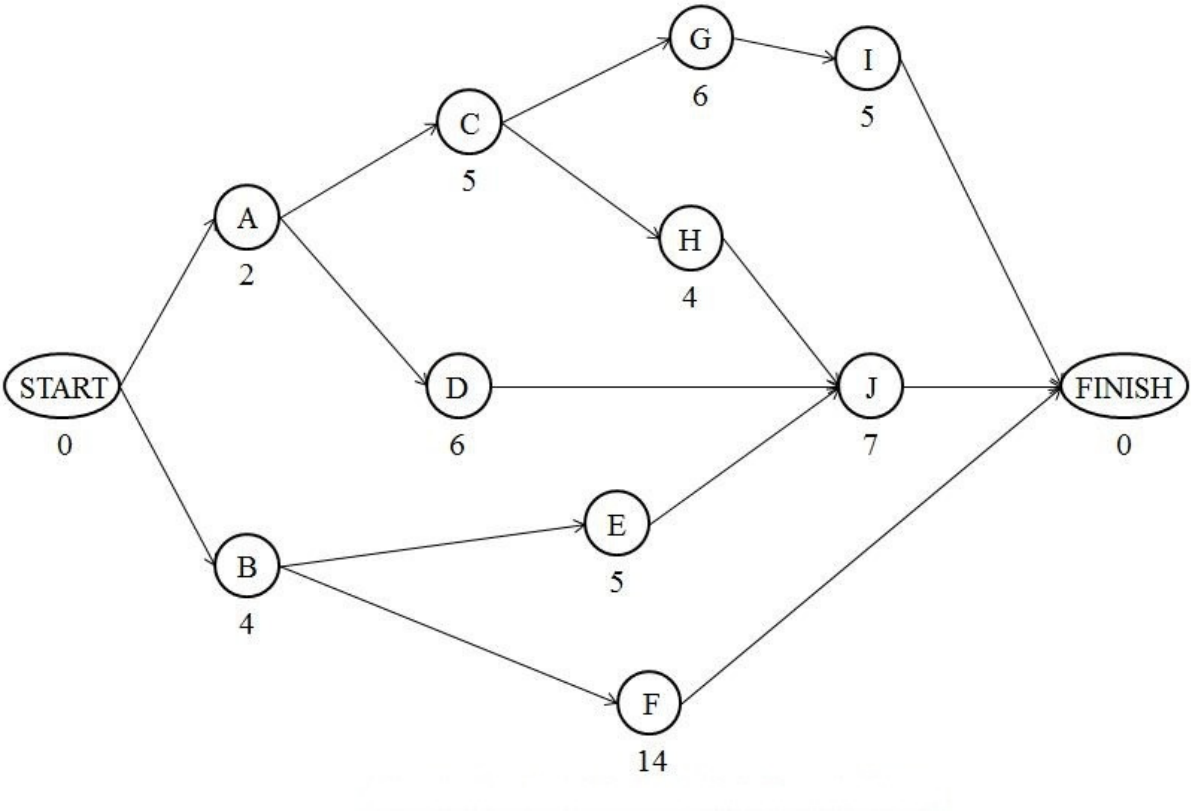


**Figure 2.** Case Study 1 Activity on Node Schedule

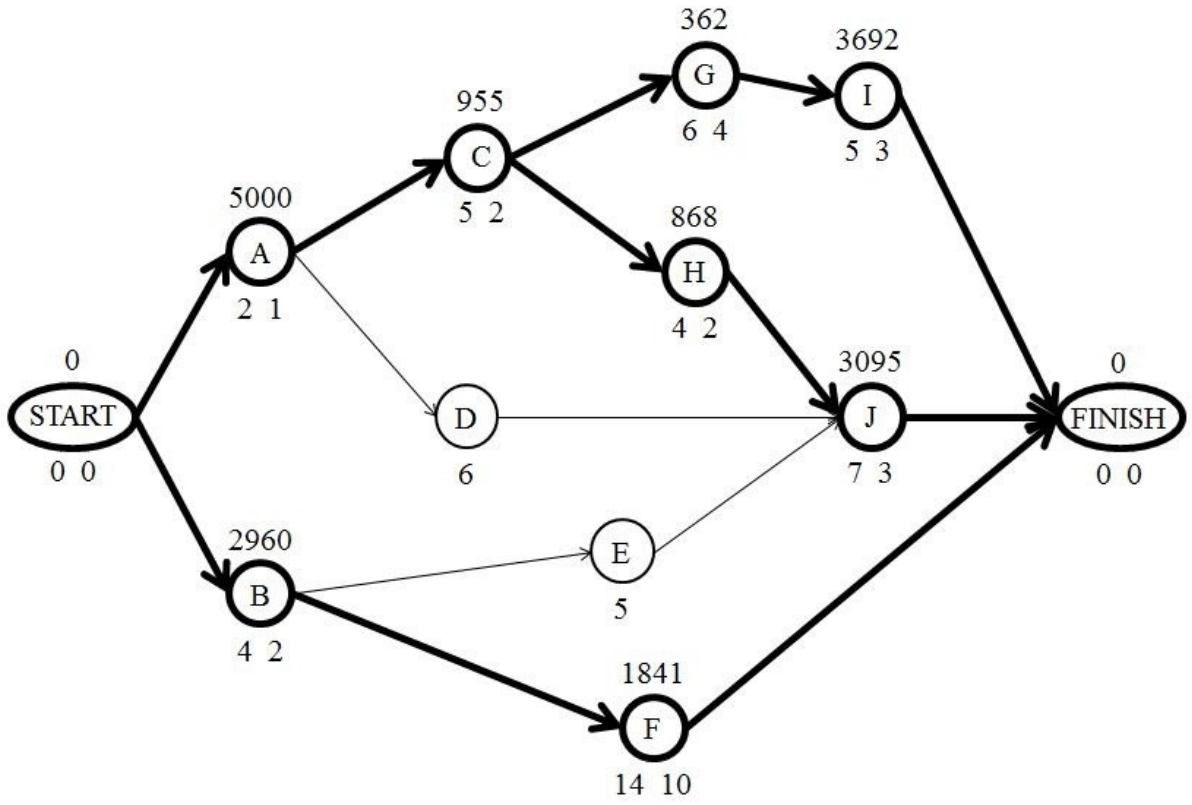


**Figure 3.** Case Study 1 Critical Paths With Crashing Slopes


## Crashing

| | Name | Duration | Cost | Crash Duration | Crash Cost | Slope | Percent Complete | Successors |
|---|---|---|---|---|---|---|---|---|
| ✎ ✖ | START | 0 | 0 | 0 | 0 | NaN | | A, B |
| ✎ ✖ | A | 2 | 5000 | 1 | 10000 | 5000 | | C, D |
| ✎ ✖ | B | 4 | 6080 | 2 | 12000 | 2960 | | E, F |
| ✎ ✖ | C | 5 | 3135 | 2 | 6000 | 955 | | G, H |
| ✎ ✖ | D | 6 | 1704 | 4 | 4000 | 1148 | | J |
| ✎ ✖ | E | 5 | 4000 | 3 | 8000 | 2000 | | J |
| ✎ ✖ | F | 14 | 6636 | 10 | 14000 | 1841 | | FINISH |
| ✎ ✖ | G | 6 | 876 | 4 | 1600 | 362 | | I |
| ✎ ✖ | H | 4 | 864 | 2 | 2600 | 868 | | J |
| ✎ ✖ | I | 5 | 6615 | 3 | 14000 | 3693 | | FINISH |
| ✎ ✖ | J | 7 | 11620 | 3 | 24000 | 3095 | | FINISH |
| ✎ ✖ | FINISH | 0 | 0 | 0 | 0 | NaN | | |

**Figure 4.** Case Study 1 CRASH Run Screenshot


**Figure 5.** Case Study 1 CRASH Result Screenshot

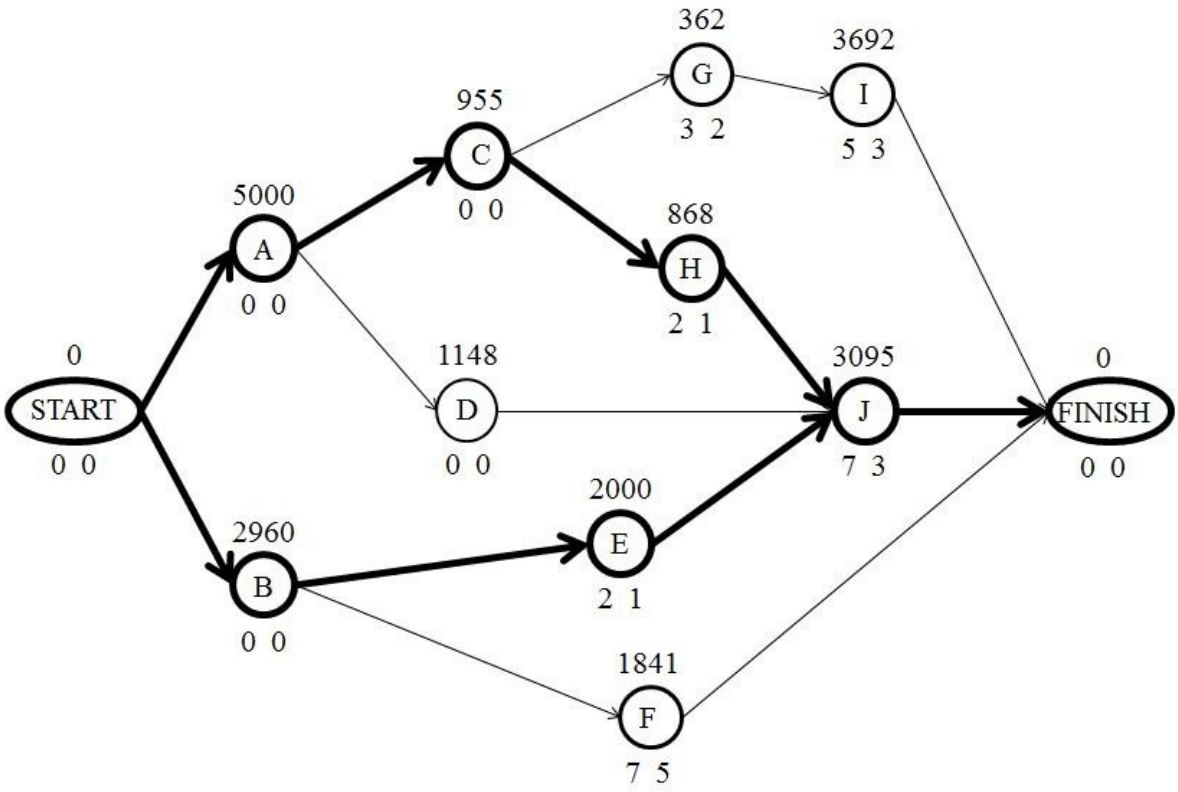


**Figure 6.** Case Study 2 Critical Paths With Crashing Slopes


# Crashing

| | Name | Duration | Cost | Crash Duration | Crash Cost | Slope | Percent Complete | Successors |
|---|---|---|---|---|---|---|---|---|
| ✎ ✖ | START | 0 | 0 | 0 | 0 | NaN | | A, B |
| ✎ ✖ | A | 0 | 5000 | 0 | 10000 | 5000 | 100 | C, D |
| ✎ ✖ | B | 0 | 6080 | 0 | 12000 | 2960 | 100 | E, F |
| ✎ ✖ | C | 0 | 3135 | 0 | 6000 | 955 | 100 | G, H |
| ✎ ✖ | D | 0 | 1704 | 0 | 4000 | 1148 | 100 | J |
| ✎ ✖ | E | 2 | 4000 | 1 | 8000 | 2000 | 60 | J |
| ✎ ✖ | F | 7 | 6636 | 5 | 14000 | 1841 | 50 | FINISH |
| ✎ ✖ | G | 3 | 876 | 2 | 1600 | 362 | 50 | I |
| ✎ ✖ | H | 2 | 864 | 1 | 2600 | 868 | 50 | J |
| ✎ ✖ | I | 5 | 6615 | 3 | 14000 | 3693 | | FINISH |
| ✎ ✖ | J | 7 | 11620 | 3 | 24000 | 3095 | | FINISH |
| ✎ ✖ | FINISH | 0 | 0 | 0 | 0 | NaN | | |

**Figure 7.** Case Study 2 CRASH Run Screenshot

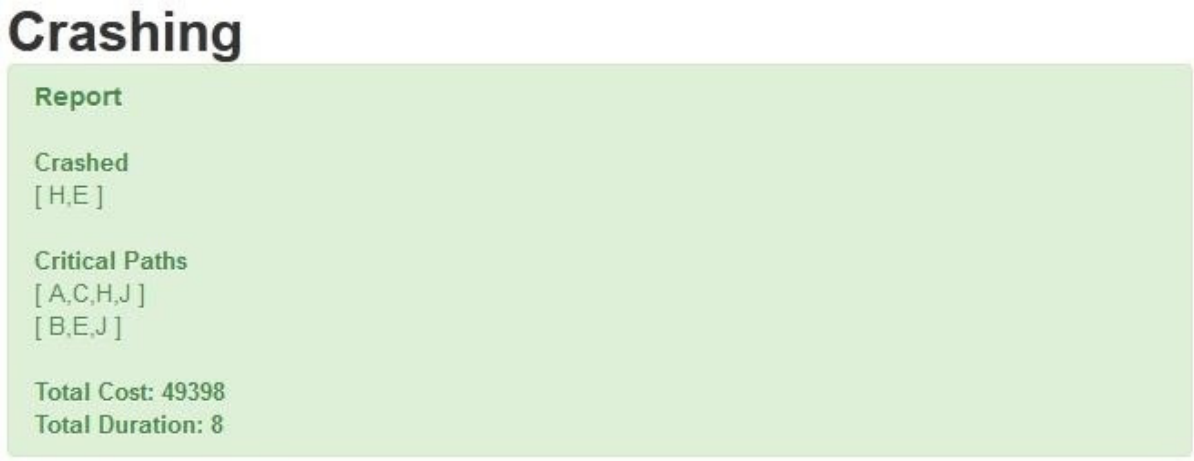


**Figure 8.** Case Study 2 CRASH Result Screenshot