CRASHING A SCHEDULE: AN ALGORITHMIC AND PROGRAMMING APPROACH

Najib Gerges PhD, PE, Fredric Ibrahim, and Daniel Hayek
Faculty of Engineering, University of Balamand, Lebanon

ABSTRACT

The purpose of this study is to tackle an important method in the scheduling of a project, crashing a schedule. Crashing a schedule has proved to be a time consuming manual task and susceptible to errors. A need for a clear algorithm is vital. In this study presented would be overviews of the subject and an explanation about the process of manual crashing. Then, it fragments the manual method of crashing into single operations. Each of those operations is analyzed for programmability and a list of programmable tasks is determined. A general algorithm is then stated, and it is programmed in JavaScript along with a user friendly web interface. An example of a new project and another for an ongoing project are then crashed manually and by the software, and the results are compared.

1. INTRODUCTION

When a new project idea arises, it is often messy and unclear. Endless questions turn up like: What? Where? Why? How? How much? Things appear to be so complex, and interrelated. Portny said that "successful organizations create projects that produce desired results in established time frames with assigned resources" (Portny, 2010). Well first things first, a project manager (PM) needs to be assigned. The project manager needs to plan, organize and control the project. He can attain that through answering three basic questions. What is the scope of the project? What is the schedule of work? And finally, what are the required resources? Some basic steps are determining the objectives, constraints and risk factors, initiate strategic planning, develop a set of outlines for the scope of work, a work breakdown structure WBS, and finally prepare and fix project milestones.

On the word 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). But bringing to action those specialists necessitates preparing an initial plan of work. In the planning phase, the PM defines elements of work, commonly referred to as tasks or activities, having defined durations and relationships. Subsequently, he assembles those activities in a certain structure for implementation, called schedule. This schedule serves as the baseline to estimate the total project duration, assign tasks and track and control the progress (Heerkens, 2001). So a project by nature is temporary, with definite start and end time.

The source of most project schedules 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 lag time for each. The total project duration is defined by the critical paths, which are the paths where accumulation of tasks durations is the longest (Levine, 2002).

Nonetheless, in construction projects, many constraints may affect the schedule. Those constraints are events and conditions unlisted in the predefined tasks, and for which the schedule needs to be altered. For instance, the project owner of a ski resort needs it to be operating by early fall season, or a reinforced concrete supplier misses the delivery date. In most cases, those constrains impose a new project deadline or reduced duration. In order to meet the deadline, changes must be made to the schedule of the project. The two main solutions for solving this dilemma are fast-tracking and crashing, or a combination of both. As each project is unique, the solutions to such situation are different and must be creative. When fast-tracking, the project manager overlaps tasks which were initially scheduled linearly, reduces lag times between tasks and splits lengthy tasks into smaller tasks with the intention of congesting more work into a shorter period. For example, a project needs to be entirely designed in order to construct it. But when fast-tracking, the project manager will start constructing a part of the project where he feels that the design is solid, without waiting for the entire design to be done. This method involves risk that could increase the cost and maybe require a rework. 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. For example, if plastering a hundred square meters needs five days for completion; at a labor cost of five hundred dollars. To complete it in three days, it will cost eight hundred dollars. Crashing a schedule, however, can be separated into two categories. The first is crashing of a project schedule that has not yet started, and the second is the crashing of an ongoing project schedule. The general method for both is almost the same, with minor calculation difference between the two. Hence, to crash a schedule, means to crash particular activities from it, resulting in an overall reduced project duration but increased project cost. But then again, the real complication is achieving a maximum reduction in the schedule time, while having a minimum added cost. Many methods may apply, but all are complicated and time consuming. Thus in a world where the business environment cannot wait, and the project managers must take decisions rapidly but rationally, the use of software is a must. Much software implements the CPM method, like Oracle Primavera, Microsoft Project and others. That software provides a quick and precise way of analyzing the project schedule. It makes a forward pass to determine the early start and finish of the activities, then a backward pass to determine the late start and finish of the activities. Once done, floats are calculated and the critical paths are determined. But the crashing feature is not present in those scheduling software. Project managers tend to crash projects based on experience or manually. In both ways, crashing a schedule is a time consuming task, prone to a lot of errors. At this point, the need for a specific algorithm for crashing a schedule is needed, as well as a preliminary programmed sample.

2. CURRENT STATE OF THE ART

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 project. 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 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.

3. ANALYSIS AND ALGORITHM

3.1 The Traditional Crashing Method

The most basic and correct crashing method is the manual method, where activities are crashed on a day by day basis. This method, although resulting in correct results, is very time consuming and can create an obstacle for project managers with limited time. The basic information needed to perform this method are five items, along with the project network diagram. First, 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. To determine the normal duration and the crashed duration, he must determine the budgeted quantity of work as well as the normal rate of productivity and the maximum rate of productivity. Then, he calculates the durations according to the following formulas (1) and (2).

$$Normal\ Duration = \frac{Budgeted\ Quantity}{Normal\ Rate\ of\ Productivity} \tag{1}$$

$$Crahsed\ Duration = \frac{Budgeted\ Quantity}{Maximum\ Rate\ of\ Productivity} \tag{2}$$

Likewise, the project manager needs to calculate the normal cost and crashed cost after determining the cost of material, equipment and manpower for each (3) and (4).

$$Normal\ Cost = Material\ Cost + Equipment\ Cost + Manpower\ Cost \tag{3}$$

$$Crashed\ Duration = Material\ Cost + Equipment\ Cost + Manpower\ Cost \tag{4}$$

Afterwards, the slope is calculated, which is the daily increase in an activity cost. This slope is calculated using formula 3.1.

$$Slope = \frac{Crashed\ Cost - Normal\ Cost}{Normal\ Duration - Crashed\ Duration} \tag{5}$$

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 formula (6), after determining the actual quantity of work performed.

$$Completion\ Percentage = \frac{Actual\ Quantity}{Budgeted\ Quantity} \times 100 \tag{6}$$

3.2 Analysis of Programmability

Converting this 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. 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

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 determined 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 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 a huge computational power and a large available storage space.

3.3 Problem Constraints

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, 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. SYNTHESIS OF RESULTS

4.1 The Crashing Algorithm

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 function is the programmed in interaction with the “Driver” function. 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. 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

Below is an example of the process:

Table 1: List of Activities and Successors

Activity	**Successors**
START	A, B
A	C, D
B	D, E
C	FINISH
D	FINISH
E	FINISH
FINISH	-

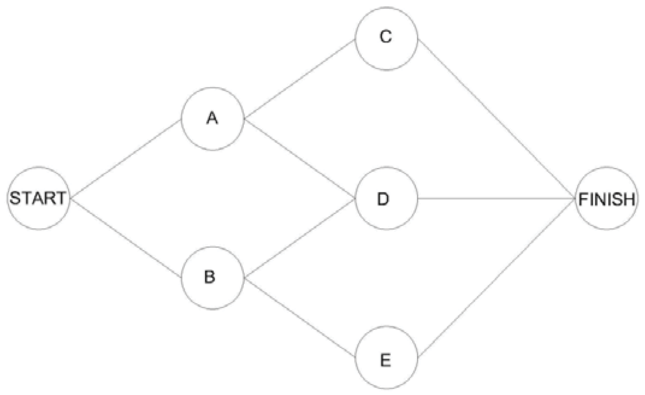


Figure 1- Network Diagram

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. 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. 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.
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. 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 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. For example, if the set is {C, A, C, D} the sum of slopes will be: Slope C + Slope A + Slope D. 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.

4.2 The Programming

The crashing algorithm is implemented as a command-line program in C++. To activate the program, the software sends to it a file name of a file containing the network diagram data and outputs a list of the activities that should be crashed. To improve user experience, a simple web interface was added to the program in a way to allow users to produce the network diagram file in a simple way and call the C++ program. The network diagram is represented as a text file in JavaScript Object Notation (JSON) which is data that consists of strings, numbers, arrays, and objects. A string is any text encapsulated in double quotes. A number is any number that can be expressed to a compute in scientific notation (e.g. 1.23e-5). The network diagram is stored as an array of objects. Each object contains an activity object and an array of the activity's successors. The activity object consists of the activity's name, duration, cost, crashed duration, and crashed cost. The network diagram file is read at runtime by the C++ program and parsed into a graph. The Boost Graph Library[1] is used to represent the network diagram in memory as an adjacency list. An adjacency list is an array where each element represents an activity, and contains a pointer to a linked list of that activity's successors. Adjacency lists are an efficient representation of network diagrams because they allow direct, linear-time access to each activity. Traversing the network diagram is equivalent to iterating over the array. The web interface allows a user to input activities along with their data and converts them into the JSON representation of the network diagram. This representation is then posted to a web server which saves it in a text file and calls the C++ program with that file as the input. The C++ program runs the algorithm on that file and prints a JSON array containing names of the activities that should be crashed. The web server then sends this array to the web interface which updates the user's view of the data.

4.3 Program Testing

A case study project to test the program is presented in Table 2 and would be explored.

[1] (http://www.boost.org/doc/libs/1_49_0/libs/graph/doc/index.html)

Table 2: Example - New Project

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

First, the network diagram should be built using the CPM method.

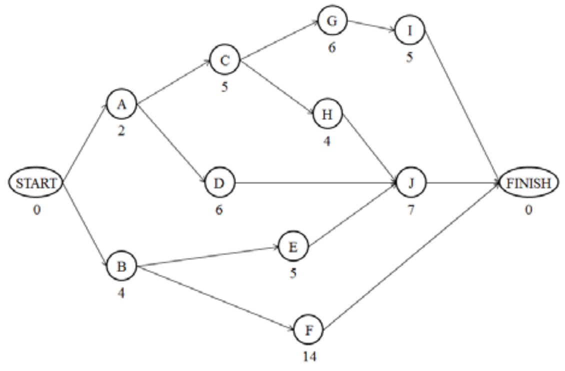


Figure 2: Network Diagram – Manual Crashing – New Project

Afterwards, the critical paths must be determined, and listed as below:

START-A-C-G-I-FINISH : 18 days

START-A-C-H-J-FINISH : 18 days

START-B-F-FINISH : 18 days

For crashing this schedule, the critical paths are the only paths that are needed. In the following figure, the critical paths are thickened.

Now, the slopes must be indicated on the network diagram along with the crashed duration of every activity. Let the slope be above the node, with the original duration under the node on the left and the maximum crashed duration under the node to the right.

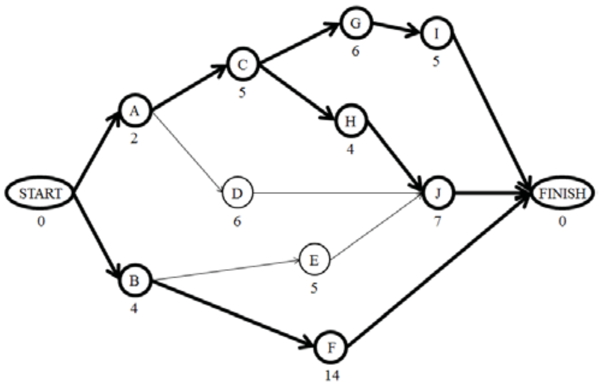


Figure 3: Critical Paths

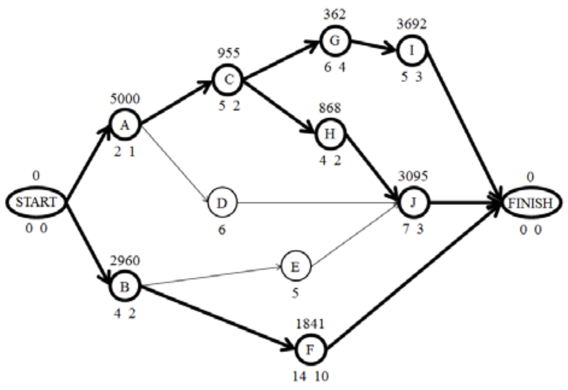


Figure 4: Critical Paths with Slopes and Maximum Crashing

Duration 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. For this example, the following are the crashing combinations to consider for maximum crashing of this schedule:

C + F : 2796$/day; C + B : 3915$/day; G + F + J : 5298 $/day; G + F + J : 5298 $/day; and A +B : 7960 $/day

After testing the software for a new project, it should be tested for an ongoing project. The same example used as a new project will be altered and used as an ongoing project. The example is stated in Table 3.
After adjusting the network diagram and removing the completed activities and adjusting the original crashing time, the following diagram shall be drawn (Figure 5).
At this stage, the same procedure applied before in section 4.3.2 shall applied to this network diagram, and the following combinations are the combinations of activities to be crashed:

H + E : 2868$/day; J + G : 3457$/day; F + J + I : 8628$/day; and F + J + I : 8628$/day

This example was loaded into the software and crashed within seconds and screenshots are shown in Figures 6 and 7.

Table 3: Example - Ongoing Project

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

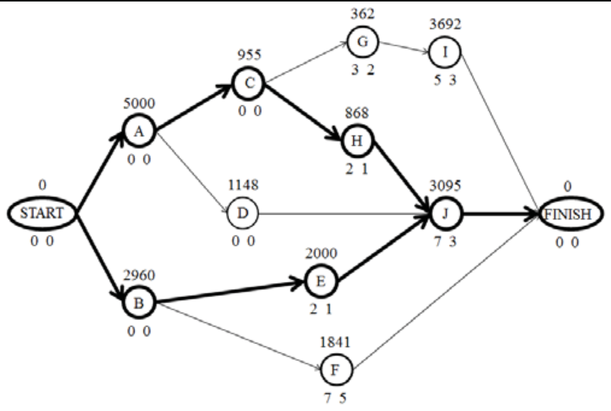
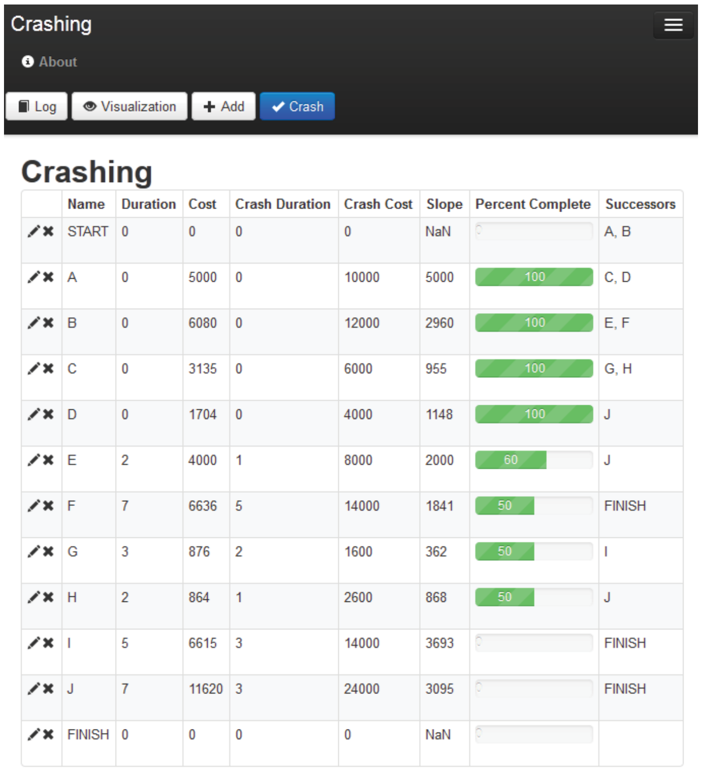


Figure 5: Critical Paths - Ongoing Project

5. CONCLUSIONS AND RECOMMENDATIONS

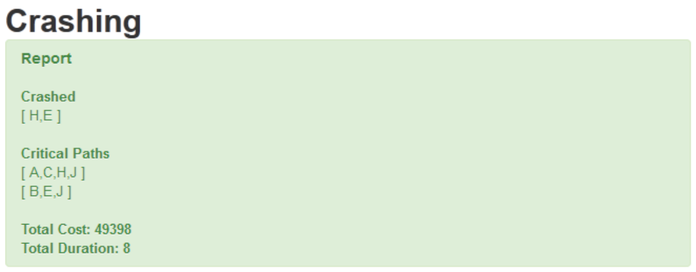
The results of this study proved that the task of crashing a schedule can be programmed. Using various techniques of combination mathematics, programming optimization and in depth pruning techniques, the software generated has an edge on the manual crashing technique, since the time used to finish the process can now be reduced to fractions of a second. Moreover, the programming of the software was accomplished through using various programming languages like C++, JAVA JSON and PHP. The C++ software ran the algorithm, the PHP web interface created an easy input method for the user, and the JAVA JSON linked the input from the web interface to the algorithm, and vice versa for the output. Proving to be complicated to use multiple programming languages and since JAVA Script is more practical, the whole software was converted to Java Script. Finally, it is recommended that this software will be tested further more to determine any residual errors or bugs. Also, it is recommended that this software gets affiliated or used as a module in some of the major scheduling software like Primavera or Microsoft Project. This way, the full potential of the software will be utilized in complicated projects where manual crashing takes hours or days. This way, thousands of dollars in savings may be made by crashing the correct item.



Crashing

	Name	Duration	Cost	Crash Duration	Crash Cost	Slope	Percent Complete	Successors
✎ ✖	START	0	0	0	0	NaN	0	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	0	FINISH
✎ ✖	J	7	11620	3	24000	3095	0	FINISH
✎ ✖	FINISH	0	0	0	0	NaN	0	

Figure 6: Software Crashing - Ongoing Project



Crashing

Report

Crashed
[H,E]

Critical Paths
[A,C,H,J]
[B,E,J]

Total Cost: 49398
Total Duration: 8

Figure 7: First Output from Software – Ongoing Project

REFERENCES

[1] Portny, S. (2010). *Project management for dummies.*(3rd ed.). Indiana: Wiley Publishing.
[2] Sayles, L.S. (2011). *Project management quotes*.
[3] Heerkens, G. (2001). *Project management.* (1st ed.). New York: McGraw-Hill.
[4] Levine, H. (2002). *Practical project management – Tips, tactics, and tools.* (1st ed.). New York: John Wiley & Sons.
[5] 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.
[6] 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.
[7] Brunnhoeffer, G.C. (2010). *Crashing the schedule – An algorithmic approach with caveats and comments.*