Criticality of Tasks within Project Management[1]

A.K. Peternella

June 20, 2016

Abstract

Project managers are charged with the responsibility of managing all the tasks within a project to ensure that the project is completed within an agreed deadline, with a certain quality, and for an agreed price. To know how much attention they need to invest in each task, managers need a measure that can capture the feature of 'criticality' of the tasks of a specified project. Criticality is defined as the importance of managing the duration of a task.

Inspired by the pessimistic bankruptcy game for the problem of cost sharing, this article explores the idea of using the combination of cooperative game theory and the Shapley value as a method for capturing the expected features of criticality. Those features are: correlation between task duration and duration of the entire project, probability of being on 'the critical path', relation to other tasks, and task dependencies.

The research presented will show that pessimistic games in their purist forms only further formalize the concept of cruciality, which is the importance of managing the duration-uncertainty of tasks. An adapted method is created, which will be proven to capture the desired features of criticality, and serve as proof of concept.

1 Introduction

This article will focus on project management i.e. the scheduling of (sets of) interrelated tasks. A project is an organization of tasks intended to a specific objective. 'Projects generally involve large, expensive, unique or high risk undertaking which have to be completed by a certain date, for a certain amount of money, with some expected level of performance.'[4]

An example of this would be the process of making paper. Starting from the cutting down of trees and transporting the wood to the factory, to packaging and distribution of the paper. All the tasks need to be managed. Some tasks are necessary and some are optional. Some tasks carry more risk of delay than others and some have (many) other tasks depending on it. This naturally leads to some tasks being more important than others and need special attention regarding management. To express this we would like to have some kind of feel, a comparable measure, for the phenomenon of different importance levels among tasks. This importance level we will call the criticality of a task. The main focus of this article will be on finding a method to define and measure the feature of criticality of tasks within project management.

There have been some attempts at defining the feature of criticality, but these attempts have only resulted in the formalization of the feature of cruciality. 'Cruciality' is defined as the importance of managing the duration-uncertainty of an activity while 'criticality' is defined to be the importance of managing the duration of an activity [5].

To achieve this cooperative game theory techniques will be incorporated where the influences between the tasks are measured and cooperative game solutions will be applied to measure criticality. The cooperative game technique and solutions are provided within the cooperative game package in Matlab, which is the program used for the implementation and testing of this concept.

The main goal is then to test how efficient this approach is at measuring criticality. Looking at some features that are considered relevant in the constitution of the degree of importance of a task and see if the developed measure reflects such features.

2 Modeling

In this section the focus is set on making a model for the purpose of assessing the criticality level of the tasks within the project. The model is then implemented in Matlab. This program was chosen for its simplicity with regard to implementing the model and for its computational power. To the implemented model the intended cooperative game techniques will be applied and tested. More details of the cooperative game

[1] This thesis was prepared in partial fulfillment of the requirements for the Degree of Bachelor of Science in Knowledge Engineering, Maastricht University, supervisor: Jean Derks.

techniques will be discussed later in this section.

2.1 Project

'A project can be defined as a set of activities with an estimated duration for which a precedence relation among them is known.' [2]

The first issue is deciding on a type of project to be modeled. There are many different models that could be used within project management. This article will focus on the PERT model. A PERT model breaks a project down into different tasks, which are linked based on their dependencies of other tasks. From the connected tasks a graph is formed. In PERT models the nodes of the graph of a project are the different phases of the project, whereas the edges of the graph are the tasks.

The straightforward thought of criticality in a project, where all tasks must be completed, is that the critical tasks are on the longest path ('critical path'). This is because the longest path is determinant of the total duration of the project. All other tasks can be completed in parallel to the ones on the longest path without influencing the total duration. However, PERT models have a stochastic structure implying that (almost) any task can have some degree of criticality [3]. Elmaghraby explains this in her paper by including the fact that some, if not all, tasks can have some delay occurring to them. The delay occurred can then change the critical path by having a new chain of tasks determine the total duration of the project. This means that there isn't one fixed set of tasks that is critical for the management of the total duration of the project. All the tasks within the project have some degree of criticality, because, if delayed, they could potentially also delay the total duration of the project.

2.2 Cruciality vs Criticality

With this incorporated notion of uncertainty we arrive at the core of Williams' [5] argument as to why previous attempts at defining the criticality of tasks have not been successful. Williams makes clear that the long established definition of criticality, which is 'the probability of the task being on the critical path', is not a very useful definition. Williams found that this definition did not give the logical information expected by project managers.

Emalghraby [3] agrees with this view and adds that the index in many realistic circumstances go against project managers' expectations. In the case where there is a project with two tasks of different duration, both of which on the critical path (the only path), this measure would imply that both tasks are equally critical. However, it is intuitively clear that the task with longer duration is more critical because the total duration time of the entire project is more dependent on it than on the task with shorter duration.

Williams went on to coin the term 'cruciality' and defined it as the correlation between a task duration and the duration of the entire project. The difference between 'cruciality' and 'criticality', Williams makes clear, is that 'cruciality' is the importance of managing the duration-uncertainty of a task while 'criticality' is the importance of managing the actual duration of a task.

Elmaghraby [3] also delved into this notion of cruciality. She introduced the 'cruciality index' (CRI) as an index giving the linear correlation between task duration and total project duration. CRI showed some advantages in reflecting the importance of tasks within a project, suggesting a degree of dependency of the total project duration on the task durations. Also CRI manages the stochastic nature of tasks in PERT models. CRI however, as Elmaghraby points out, measures a linear dependency between task durations and the whole project duration and this doesn't always need to be the case.

To achieve an accurate measure for criticality a method is needed that not only considers the probability of a task being on the critical path, but also considers the actual duration of the tasks in relation to the duration of the entire project, as well as that of all the other tasks in the project. Moreover the dependencies among the tasks should be noticeable in the criticality index. These features are recognizable when using cooperative game theory concerning cost-/profit allocation.

2.3 Cooperative game techniques

Cooperative games are used for the distribution of some total value in a 'fair' way among the players of the game. The 'value' can be anything measurable and divisible, for example: cost, profit, time, etc. The players are the people or things the value is to be shared over. A 'fair' distribution of the value in a cooperative game is one which meets the requirements of efficiency, the null-player rule and symmetry. A distribution is efficient when the sum of the value of all the players equals the total value. The null-player rule states that, for example in the profit distribution, a player that does not contribute anything to the total profit cannot receive part of the profit distribution. Similarly, a player, in a cost distribution game, cannot bare any of the costs if it did not contribute to the total delay. Finally, symmetry simply means that two identical players should receive equal shares of the total distribution. The specific type of cooperative game dealt with in this article is a characteristic function game.

A characteristic function game $G(N, v)$ consist of a

set of players N and a function v that takes every non-empty coalition of players ($v : 2^{[N]} - 1 \Rightarrow \mathbb{R}$) and assigns it a value. In figure 1 an example game is presented for three people (A, B and C) taking a taxi together from starting point S. The assumption is made that the destinations of A and B are on the way to the destination of C (no detours).

S	$v(S)$
$\{A\}$	6
$\{B\}$	13
$\{C\}$	40
$\{A, B\}$	13
$\{A, C\}$	40
$\{B, C\}$	40
$\{A, B, C\}$	40

Figure 1: Example for a cost distribution game with 3 players with the table of contributions.

A, B and C need to pay 6, 13 and 40 respectively to take a taxi alone to their respective destinations. In the table in figure 1 the contributions of the coalitions are shown. The contributions will be the key for calculating the fair share of the cost for the taxi for each person when taking the taxi together. This is done by applying the Shapley value.

Shapley value

The Shapley value is used in a characteristic function game to calculate the amount each player would pay (in a cost game) in each permutation of an ordered set of the player. The results are then averaged and we arrive at the Shapley value. The function for the Shapley value looks as follows:

$$\phi_i(G) = \frac{1}{n!} \sum_{\pi \epsilon \Pi_n} \Delta_\pi^G(\mathrm{i}) \qquad (1)$$

The way the Shapley value works is illustrated in table 1. The first column (π) contains all the ordered permutations of set $\{A, B, C\}$. In the first row, for example, A would pay first, B second and C last. Using the contributions illustrated in figure 1, the calculations are made. Player A first looks at the the contribution of the set containing only A, $\{A\} = 6$, and pays 6. Then when player B will pay, it will look at the contribution of the set containing A and B. $\{A, B\} = 13$, but since player A already payed 6, player B will pay the remaining 7. Similarly, when player C will pay, it sees $\{A, B, C\} = 40$ and that 13 has already been payed by A and B, so it pays the remaining 27. All the other rows are then filled out in the same manner. The final row ϕ contains the average cost for each player over all the order permutations.

π	A	B	C
(A, B, C)	6	7	27
(A, C, B)	6	0	34
(B, A, C)	0	13	27
(B, C, A)	0	13	27
(C, A, B)	0	0	40
(C, B, A)	0	0	40
ϕ	2	5.5	32.5

Table 1: Shapley value calculation

The concept of applying cooperative game theory to a project, inspired by the pessimistic bankruptcy game discussed by Branzei et al.[1], will be discussed in the next section.

2.4 Expectations

As discussed before, to find out if the application of cooperative game theory and the Shapley is useful for measuring criticality of tasks, the results need to show a number of features. The obvious one is the distinction between criticality and cruciality. The intention is to have the cooperative game and Shapley value be based on the duration of the tasks. Looking at the results found in the previous example given, it is clear that the Shapley value applied to a cooperative game gives all the players (which would be the tasks of a project) a value in relation to its value (duration). However, some attention needs to paid to the stochastic nature of the PERT model. Experiments need to be conducted to determine the influence of uncertainty.

Another important feature the measure of criticality needs to adhere to is that of dependencies among tasks. Tasks that have multiple other tasks depending on its on time completion are naturally expected to be more critical. Any delay caused to such tasks would propagate to the next tasks and maybe even trigger more delays. The calculations for the contributions of a characteristic function game are done based on sets. Thus the values can be assigned/calculated based on the paths of the project. Tasks that are on multiple paths in the project will have more non-zero entries in the table when calculating the Shapley value, and therefore a higher value when averaged over all entries. The

expectations as to whether cooperative game techniques can capture this feature are therefore quite positive.

3 Experiments

To start the experimentation on whether cooperative game techniques will capture the expected features of criticality, an example project needs to be created based on the model established in the previous section. Figure 2 and table 2 show the example project that will be used throughout this section.

In the graph in figure 2 an example project can be seen with 5 phases (A, B, C, D and E) and 6 activities. In table 2 the minimum duration for each task can be read, in the second column, as well as the maximum delay time of each task, in the third column. This example graph has been chosen because it seems simple, yet it contains some interesting properties which will show to be helpful for testing purposes.

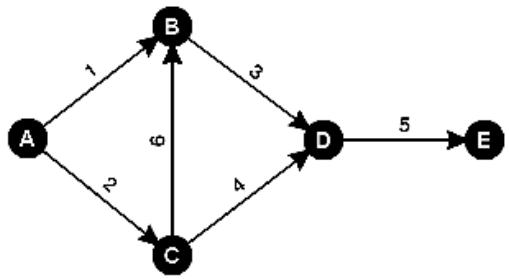

Figure 2: Example project with 5 phases and 6 tasks labeled.

Task	Duration	Possible delay
1	7	2
2	5	2
3	6	2
4	4	4
5	1	2
6	0	10

Table 2: Base case duration and delay of the tasks.

The actual duration (a_i) of an activity i is calculated as the base duration (b_i) plus some random duration within the possible delay window (pd). This is shown is function 2. In function 3 the calculations for the 'phase arrival times' (PAT) is shown. Those are the start times for each of the phases of the project. The details of this function will be explained later in this section. Lastly, what function 4 simply says is that the total project duration is equal to the arrival time of the last phase.

$$a_i = b_i + rand() * pd \tag{2}$$

for j=1 : m (repeated while $\exists\ p_j$ that has been updated)

$$p_j = max\{p_j,\ (p_{i-1} + a_{p_{j-1},pj})\} \tag{3}$$

$$E = p_m \tag{4}$$

Branzei et al.[1] discuss in their paper the similarities of the problem of cost allocation to the theory applied in cases of company bankruptcy.

3.1 Pessimistic bankruptcy game

In their paper, Branzei et al, discuss project activities as having a certain expected time for completion. Those are the minimum times, which together with the graph of the project constitute the expected start and end times for each activity (b_i and e_i respectively). Of course things do not always go as planned so they used actual ending times (e^*) as well to deduce delays occurred. These are then compared to the total delay of the project (D) to evaluate how much any delay for each activity affects the project, if at all.

From these parameters they set up two functions:

$$D = (E^* - E)_+ \tag{5}$$

$$d_i = ((e_i^* - b_{+i})_+ - (e_{-i}^* - b_i)_+)_+ \tag{6}$$

Function 5 simply states: the total delay of the project is the actual ending time (E^*) minus the scheduled ending time (E). The $_+$ after the bracket simply means that if the result of $E^* - E < 0$ the total delay is then 0.

Function 6, activity delay, is a bit trickier so it will be explained in parts. $(e_i^* - b_{+i})_+$ is calculating whether the current activity is finishing later than the scheduled start time of its followers. If so, the value will be positive, 0 otherwise. Similarly, $(e_{-i}^* - b_i)_+$ is calculating whether the previous activities finished later than the scheduled start time of the current activity. The difference of these two calculations ensures that any delay caused by previous activities are not counted as delay for the current activity. Again, if this value is < 0 the activity delay is simply 0.

PROP/CER

Here the similarities to the bankruptcy problem arise. 'A bankruptcy problem is described by a pair (E, d) where E is the estate and $d = (d_i)_{i\epsilon N}$, where d_i as the claim of

customer i, with $0 < E \leq \sum_{i=1}^{n} d_i$. Here the problem is how to divide the estate E among the claimants.'[1]

The proportional rule (PROP) is a method used to calculate the proportional share of the total delay, based on the individual activity delays. By looking at function 7, this is done by normalizing all the delays $\left(\left(\sum_{i=1}^{n} d_i\right)^{-1} d_i\right)$, and then multiplying each with the total delay (D).

The constrained equal reduction rule (CER) works in a slightly different manner (function 8). It basically takes the difference between the sum of delays and the total delay and dividing that by the number of activities $\left(\left(\left(\sum_{i=1}^{n} d_i\right) - D\right)/n\right)$, calling this value β, and reducing it from the delays $(d_i - \beta)$. Once again, if this value would be negative, the value is set to 0.

$$PROP_i(D,d) = \left(\sum_{i=1}^{n} d_i\right)^{-1} d_i D, i = 1, ..., n. \tag{7}$$

$$CER_i(D,d) = (d_i - \beta, 0)_+, i = 1, ..., n. \tag{8}$$

These similarities seem to show a good basis for the calculation of criticality, sharing the total cost (delay duration) among the activities that contribute to it. However, note that if $\sum_{i=1}^{n} d_i = D$ (meaning all delays are sequential and on the critical path), PROP would just return a value equal to the delays. Meaning all other activities would not carry any value, which as mentioned in section 2.2, is not what we expect from a criticality measure.

Pessimistic game

A pessimistic game, as discussed by Branzei et al,[1] is a different approach to the problem of sharing the delay cost by means of coalitions of activities, instead of individual activities.

$$c_{(D,d)}(S) = min\left\{\sum_{i \epsilon S} d_i, D\right\} for\ each\ coalition\ S \subset N. \tag{9}$$

The contributions corresponding to the delay problem (D,d) are calculated according to function 9. The contribution of a certain coalition S is $\sum_{i \epsilon S} d_i$. If that value is greater than the total cost, D, the contribution is then equal to D. the reason for this is that otherwise more cost would be shared than actually has been induced to the project. To these contributions the Shapley value can be applied.

The fact that in a pessimistic game the cost is shared based on the joint contributions of delay of set of activities gives hope for a better approximation to a criticality measure. However, as mentioned in section 2.3, explaining the Shapley value, the null-player criteria still holds for activities with delay 0. So any activities without delay would still have Shapley value 0 in the end. To tackle this the next experiments have been conducted over the example project (figure 2) where all tasks have some randomized delay within their maximum delay window. For each run the cooperative game is set up and the Shapley values calculated. The process is repeated over 10000 runs and then the Shapley values for each task is averaged over all runs.

To test if the results of the pessimistic game really captures criticality, as suggested by Williams [5], the measure needs to show some correlation of the task duration and the total duration of the project. In the graph in figure 3 the results are shown to the experiment of changing the base duration (possible delay stays the same) from $0 - 20$, with 1 unit steps, of activity 4.

Activity 4 was chosen specifically for its features. First of all, it has low dependency, meaning there few (1) tasks depending on it. Secondly, it does not lie on the critical path (defining the total duration), so there is some room for increase without immediately gaining a lot of criticality. And lastly, it can and will become part of the critical path within the testing range.

Results

As can be seen in the graph in figure 3, the results show that the value has the tendency to stay constant as the duration increases and the total project duration stays the same. The first part of the graph is constant near value 0, when the activity is not in the critical path. The last part of the graph is also nearly constant around value 1.1 when the activity is nearly always in the critical path. The interesting slant in the middle is simply the average of the two extremities over the 10000 runs, sometimes it falling on the critical path, sometimes not.

It seems that the pessimistic game fails to capture criticality in two ways. Firstly, it fails to assign a criticality value higher that 0 for activities that have duration > 0. Secondly, it fails to give a correlation between an activity duration and the total duration of the project. The reason for this is believed to be because the base for the calculations are the delays and not the actual durations.

Looking at the graph in figure 4 where the same test was performed with, instead of varying duration, varying delay uncertainty, we get very different results. There appears to be a strong correlation between the delay uncertainty and the value received for the Shapley value applied to the cooperative game.

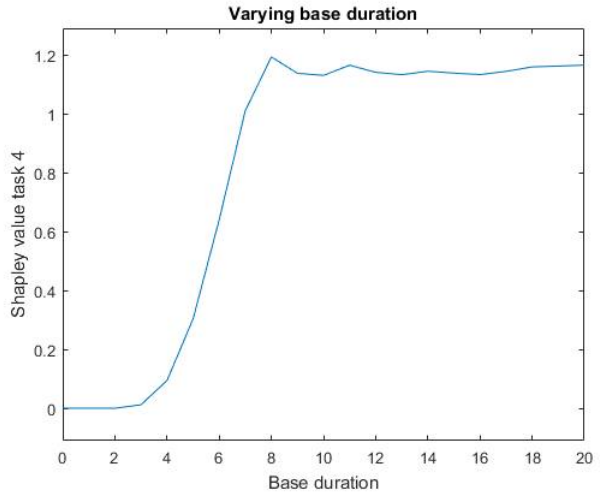

Figure 3: Results of experiment pessimistic game with varying base durations.

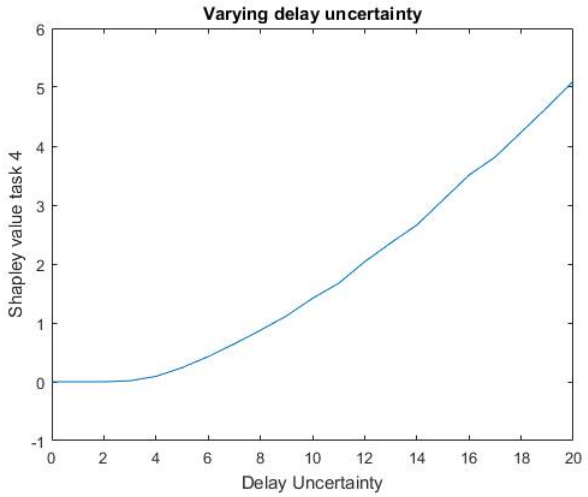

Figure 4: Results of experiment pessimistic game with varying delay uncertainty.

Failure to define criticality

It is clear that the pessimistic delay game does not define criticality of tasks, nonetheless, the pessimistic delay game shows some desired features. By making a modified version of the pessimistic delay game a better approximation could be achieved.

The previous experiment has shown that the pessimistic delay game leans more toward defining cruciality, instead of criticality. The pessimistic delay game has shown a strong correlation between the delay uncertainty and the value it receives from the Shapley value calculations. Recalling Williams' [5] statement, cruciality is the importance of managing the duration uncertainty (in this case delay). The modified version needs to be based on the actual duration of the tasks.

3.2 New approach

The new approach that will be taken to this problem will still be using cooperative games, to include the feature of joint contribution, and the Shapley value, for the feature of 'fair' share assignment. However, now the delay calculations will be disregarded and the actual task durations will be incorporated straight away.

Application

Changing the basis for the cooperative game from delay to actual duration (a) makes for a simplification of the problem, considering the individual delay calculations need not to be performed. The 'total value' is now, of course, simply the total actual duration (A), which can be calculated with the same method for the calculation of the total delay. The pessimistic game with this change looks now as follows:

$$c_{(A,a)}(S) = min\left\{\sum_{i\epsilon S} a_i, A\right\} for\ each\ coalition\ S \subset N. \tag{10}$$

An experiment has been conducted with this set up using the same parameters as when calculating the pessimistic delay game (10000 runs, random delays assigned to each activity in each run according to their delay window). It is important to note that a seed has been used to ensure the random number generator (rng) used in Matlab would be using the same ordered set of random numbers for all different tests. A method that uses random numbers, performed twice with the same seed, will have the same results. So two different method using the same seed would really show the differences of the two methods in equal circumstances.

In figure 5 we can see the same varying base duration test as done in figure 3, but this time with all of the tasks. (Note that each line on the graph represents one test where the corresponding task has been assigned varying duration, while the other tasks remain as stipulated in the base case. Basically, six experiments have been conducted and fused into one graph.) Similarly figure 6 shows the varying delay uncertainty test also done with all six of the task.

As can be seen in figure 5 the method seems to have a more logically desirable correlation between activity duration and total duration time, with respect to the other tasks. Although the lines travel a similar path it does not mean the tasks have similar criticality value since the actual value lies at the expected average base duration for each task (for example, task 5, the duration is between 1 and 3, so the expected duration is 2). The

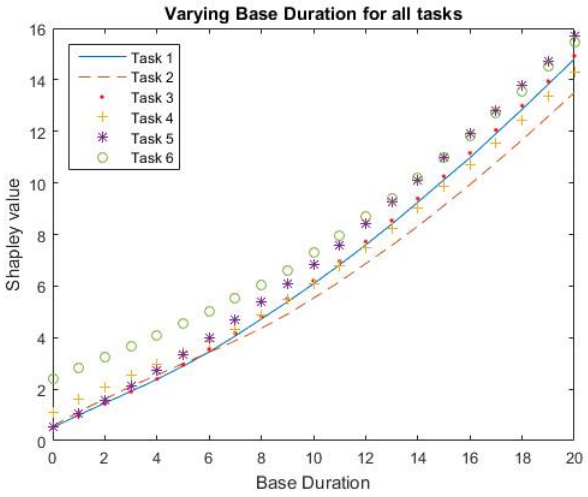

Figure 5: Testing criticality based on task duration.

differences based based on the graph only start to show up for higher values. Figure 6 suggests that there still is some

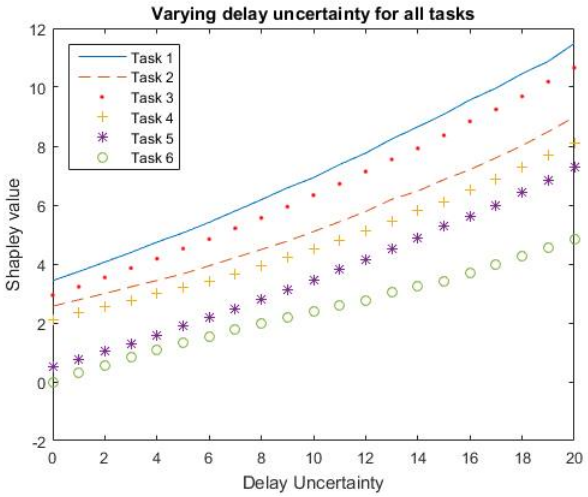

Figure 6: Testing criticality based on task delay.

correlation of the delay uncertainty and the criticality value. This is understandable considering the delay uncertainty inevitably has a consequence for the actual duration. Nonetheless, it is clear that the correlation is now reduced as the graph shows no sudden steep increase in the graph. The values have a steady, near linear, incline.

Also a bit more clear from this graph is the difference of the tasks in criticality. The task criticality order is almost completely preserved throughout.

Short coming

The modified pessimistic game has the features of correlation of activity duration and total duration, and in relation to the other task durations incorporated for the calculation of the criticality value. Also the probability of the task being on the critical path has been accounted for through the cooperative game. The only feature not yet accounted for is that of task dependencies. The graph in figure 5 already hints to this as a bigger difference between the tasks criticality value would be expected because of dependencies.

Through a simple analysis of a single coalition it can be shown that dependency is not yet accounted for in this modified pessimistic game. In figure 7 we can see the example coalition $S = \{1, 2, 4, 5\}$ (see also figure 2). The minimum durations are used for simplicity.

In this coalition the sum of the durations $\{\sum_{i \epsilon S} a_i) = 7 + 5 + 4 + 1 = 17$. The actual duration $A = 7 + 6 + 1 = 14$ (critical path in the case of minimum durations is $\{1, 3, 5\}$). That would mean that the corresponding value for that coalition is: $c_{(A,a)}(\{1, 2, 4, 5\}) = min\{17, 14\} = 14$. However, if the corresponding value would be calculated in the same fashion as the total duration, with the activities not part of the coalition having value 0, the result would be $A = 5 + 4 + 1 = 10$ (critical path of the coalition is $\{2, 4, 5\}$).

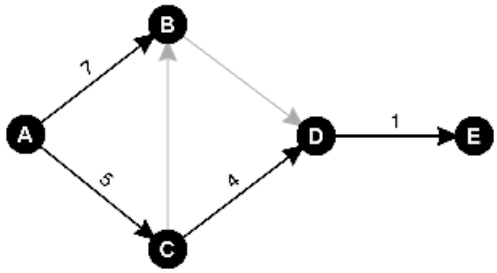

Figure 7: Example coalition with duration values along the edges.

From this we can deduce that, for some coalitions in the (modified) pessimistic game, more value is being shared to activities, as the coalition contributions are not being calculated in the most accurate way.

Adapted method

An adapted method needs to be created which calculates the contributions of the coalitions in a more accurate manner to better represent the dependencies among the tasks. The set of equations 11, 12 and 13 establish the approach taken in the adapted method.

Vector t, the 'task coalition values', hold the values assigned to each task in a specific coalition, being equal to the actual duration of the task if it is in the coalition, and 0 otherwise. These values are then used to calculate the phase arrival times (p).

Function 12 is a repeated loop where at each iteration updates the values for the start time of the phases of the project. These are calculated as the maximum between the value they already have assigned (p_j) and the value of the preceding phase summed with the task coalition value of the task connecting that preceding phase and the current one ($p_{i-1} + t_{p_{j-1},pj}$). The calculations are done for all m phases and then this function needs to be repeated for as long as there is a phase in the project for which its value has been updated. An updated phase value could mean that that value still needs to be propagated further through the graph.

Finally, the last equation simply states that the 'coalition duration' of set S is equal to the final arrival time of phase m (assuming phase m is the last phase, i.e. where the project is considered completed).

$$t_i(S) = \begin{cases} a_i, & \text{if } i \epsilon S \\ 0, & \text{otherwise} \end{cases} \tag{11}$$

for j=1 : m (repeated while $\exists\ p_j$ that has been updated)

$$p_j(t(s)) = max\{p_j,\ (p_{i-1} + t_{p_{j-1},pj})\} \tag{12}$$

$$cd(S) = p_m \tag{13}$$

Figure 8 shows the experiment that will be performed for testing how the adapted method compared to the modified pessimistic game. Three scenarios are depicted where task 6 has a different ending phase. The top one, the same as in the example graph in figure 2, has 2 tasks depending on task 6. In the middle graph the end phase is changed to D and then only has the finishing task as a depending task. In the bottom scenario, task 6 is linked directly to the final phase, meaning there are no tasks that depend on it.

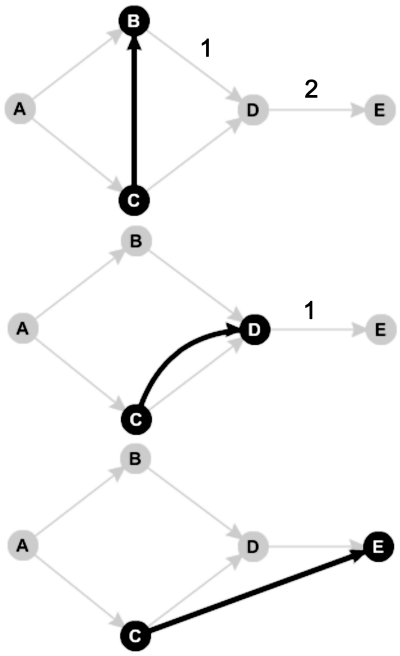

Figure 8: Changing a tasks' link to test criticality based on dependency.

Results

Table 3 shows the results to the dependency experiment. Both methods have been run 10000 times with the three different dependency scenarios using the same seed, for the random number generator, for all 6 combinations. The resulting criticality indexes have then been averaged over the runs and then normalized.

The results show the differences between the two methods very clearly. In all three scenarios, the modified pessimistic game gives nearly the same results for all tasks. The change in dependencies shows no significant change in the criticality indexes of M1 (modified pessimistic game). On the contrary, M2 (the adapted method) shows change in the criticality indexes as expected when changing the ending phase of task 6.

Task 6 shows a very significant drop in criticality when its number of depending tasks is reduced from 2 to 1 and when reduced from 1-0. The switch in scenarios is also accurately readable in the criticality indexes of the other tasks. Task 1, for example, shows a steep increase, as with the change in dependency from the first scenario ensures task 1 has a much higher probability of being on the critical path. Task 3 shows first a big decrease as there are now fewer paths (coalitions) that provide it with criticality. When task 6 changes from 1 to 0 depending tasks, it looses some more criticality. These are transferred, mainly, to the tasks on the most probable critical path ($\{1, 3, 5\}$), hence the increase in criticality of tasks 1 and 3.

	Tasks					
M	1	2	3	4	5	**6**
M1(2)	.2357	.1745	.2057	.1723	.0587	**.1531**
M1(1)	.2379	.1757	.2076	.1750	.0608	**.1430**
M1(0)	.2381	.1758	.2078	.1751	.0609	**.1423**
M2(2)	.1551	.2086	.2880	.0927	.0986	**.1569**
M2(1)	.2541	.1779	.2236	.1248	.1170	**.1027**
M2(0)	.2669	.1741	.2348	.1313	.1037	**.0893**

Table 3: Results of experiment showing the influence of dependency on task 6, to its criticality value. 'M1' is the modified pessimistic game and 'M2' is the adapted method.

4 Conclusion

As hoped, cooperative game theory, specifically the Shapley value has indeed shown a power to capture the features of criticality. The pessimistic bankruptcy game, in the scope of cost sharing, had been a useful inspiration for the application of cooperative games techniques to deduce joint contributions to the duration of a project. However, pessimistic games in their purist form have only shown to formalize the concept of cruciality.

By modifying the pessimistic game, from using activity delay to task durations, more features of criticality have been captured. The modified pessimistic game managed to provide the desired correlation between task durations and the duration of the entire project and the probability of a task being on the critical path.

The only apparent feature the modified pessimistic game was lacking is that of dependencies of tasks. This problem was tackled by making an adapted method which better estimates coalition contributions. The adapted method has shown to accurately simulate the effects of dependencies of tasks. The manner in which the adaptation had occurred, ensured that the other features of criticality were preserved. This brings as a conclusion that the adapted method captures all the desired features, stipulated in section 2, of criticality.

Future Work

This article serves mainly as proof of concept. Further research and analysis needs to be done on cases where this concept might not hold, or for what values/value differences the results might differ from what would be expected.

Research should be conducted for methods that further improve in more accurate estimations of the contribution of coalitions to the total durations.

References

[1] Branzei, Rodica, Ferrari, Giulio, Fragnelli, Vito, and Tijs, Stef (2002). Two approaches to the problem of sharing delay costs in joint projects. *Annals of Operational Research*, Vol. 109, pp. 359–374.

[2] Castro, Javier, Daniel, Gmez, and Tejada, Juan (2007). A project game for pert networks. *Operations Research Letters 35*, pp. 791–798.

[3] Elmaghraby, Salah E. (2000). On criticality and sensitivity in activity networks. *European Journal of Operational Research*, Vol. 127, No. 2, pp. 220–238.

[4] Steiner, G.A. (1969). Top management planning macmillan. *Quoted on page 481 of Project Management Handbook, 2nd edition.*

[5] Williams, Terry (2003). The contribution of mathematical modelling to the practice of project management. *IMA J Management Math*, Vol. 14, No. 1, pp. 3–30.

A Code for the adapted method

```
N=[1 2;1 3;2 4;3 4;4 5;3 5]'; % task graph
dur=[7 9; 5 7;6 8;4 8;1 3;0 10]'; %task min and max durations
num=2^(length(N(1,:)))-1; %number of coalitions
n=size(N,2); % number of tasks/links
S=max(N(2,:)); % number of phases (last stage is finishing phase)
a=dur(1,:); % actual task durations (min durations as default)
temp=zeros(1,n); %

%if 0 Shapley matrix H(S,i)=p(s) if i in S, and -p(s+1) otherwise
c=1; for i=2:n, c=[c 1 c+1]; end % c(S) cardinality of coalition S
l=length(c); % number of non-empty coalitions
p=zeros(1,n); p(1)=1/n; for i=1:n-1, p(i+1)=p(i)*i/(n-i); end
H=zeros(l,n);
for i=1:n
    for j=1:l
        cj=c(j);
        if bitget(j,i)
            H(j,i)=p(cj);
        else H(j,i)=-p(cj+1);
        end
    end
end

%constitution of actual durations
for i=1:n,
    if rand>0 %probability of not having delay
        a(i)=a(i)+rand*(dur(2,i)-dur(1,i));
    end
end
a; % actual duration

v=zeros(1,num); %initializing coalition contribution vector
for T=1:num %running through all coalitions
    temp=a;
    for i=1:n
        if bitget(T,i)==0
            temp(i)=0;
        end
    end
    A=zeros(1,S);
    cnt=1;
    while cnt
        cnt=0;
        for i=1:n % running through all activities
            s=N(1,i); t=N(2,i);
            if A(t)<temp(i)+A(s)
                A(t)=temp(i)+A(s);
                cnt=cnt+1;
            end
        end
    end
v(T)=A(S);
end
Shapley=v*H % Shapley matrix H is computed beforehand
```