Longest Path Value (to the Rescue)

Ron Winter Consulting LLC

Abstract

This paper presents the origin and inherent difficulties in using Total Float as the sole basis for determining the relative importance to project completion between various activities in a CPM network. The concept of a 'longest path' is presented as solution to this problem and this theory is extended to include all activities in a CPM network. New concepts like inter-activity float (labeled 'slack') and logical order are presented that lead to the calculation of a Longest Path Value to be used in place of Total Float.

Background

Some people have said that the newer features of modern CPM schedules obfuscate and allow for bad schedules. Some even believe that these features cause bad schedules. Some advocate the return to simple ADM without constraints and multiple calendars. They cite the many problems this causes to the understanding to the Total Float calculation. Looking backward is only one approach to addressing the problem.

Another approach is better training. But this is a partial solution. You can't fix a problem by just assuming a different environment. This just does not match with reality. The reality is that every year, thousands of new people are introduced to CPM scheduling without training and we need to consider this fact in our solution.

A third solution is to improve the CPM tools that we use. By this, I don't mean making the software easier to use. That, in itself has not lead directly to better scheduling. I am saying that we need new concepts and measurements that overcome the weaknesses of the current methodology. Just because CPM is over 50 years old, this does not mean that the entire width and breath of this 'science' has been fully defined. Let us look at current standards and how they developed.

Introduction

First actual application of a Critical Path Method (CPM) program was developed by a joint venture of E. I du Pont de Nemours and Company and the Sperry-Rand Corporation and run on a UNIVAC I computer in 1957 [1].

CPM is an analytical method of scheduling tasks (called “activities”) in a project. This is preformed by logically breaking a project into activities, estimating each activity’s duration, and linking each activity into a logical framework. Using a formalized set of procedures,

Performing the ‘forward pass’ CPM calculations to a logical network of activities tells you the earliest time in which a project can be completed. The date each activity is scheduled to begin is know as the ‘Early Start’ and the date that each activity is scheduled to end is called ‘Early Finish.’

If you repeat the CPM computational process but this time start with the last activity in the network and work backward in time, the overall project length should not change but the start and finish dates of individual activities may change. This backward CMP computation is called a ‘backward pass.’ The start dates are now called ‘Late Start’ dates and the activity finish dates are called ‘Late Finish’ dates. For any given activity, the difference between the early dates and the late dates is called “Total Float.”

Total Float (float) is the amount of time an activity can be delayed without delaying the overall project completion time. This calculation is the “heart and soul” of the entire CPM process as it put individual activities into context as to their importance toward the timely completion of the scheduled project.

The float of an activity is computed by subtracting its early finish from its late finish, or by subtracting its early start from its late start [2]. Unfortunately, sometimes you get different answers depending upon which method you use. Sometimes you don’t get the value for float that common sense tells you that you should get.

Something’s Wrong

This disparity between numerical float answers depending upon your choice of computational methods is especially true with Hammock and Interruptible Activities.

Hammocks are ‘pseudo-activities’ that summarize the time interval and Total Float of a group of other activities. If the activities at the start of the time interval have a low Total Float value but the ones at the end have high Total Float, which values do you use for the Hammock?

Interruptible Activities are activities that are constrained to start at one period and constrained to finish on another. If the two opposing constraints do not neatly fit with the time period allotted for that activity (which occurs frequently,) then the start of the constrained activity is delayed to satisfy the finish constraint. In this

case, the duration difference between the start dates may be different than the duration difference of the finish dates. This feature is further complicated by the ability to allow these 'Continuous' activities to become 'Interruptible' activities, thereby fixing the float problem at the expense of having the duration of the Interruptible Activity automatically stretched to fit.

Next, activities operating under a different work calendar than its logical neighbors may have a different float value that what you would expect. For example, say we are going to pour a concrete pad and then begin building on it as soon as it is ready. The concrete is poured during the work week but it cures continuously. Concrete doesn't take the weekend off while it is curing. Now say that the cure time is reached during the weekend. The next activity cannot start right away because we have to wait for a weekday to begin building. In this simple, three-step process, the float of the middle activity is different from the other two.

'Everyone' knows that completed activities do not have float. Unfortunately, 'everybody' is wrong. It exists and would be useful in the case of activities completed out of sequence. The CPM programs that we use just refuse to display it. This value would be useful in certain circumstances.

Schedules that have been resource loaded and then leveled also do not show the correct float, nor does the P3 'Longest Path' work correctly in this instance. After the CPM has been calculated, P3 overrides the early and late start dates of leveled activities to schedule them when resources will allow.

Finally, constraints can directly or indirectly change the float of an activity. With constraints, you can make any group of activities the critical path or arrange things so that no activity in the entire project has zero float.

The problem of float further compounds when you relate like activities together in a concept of 'float path.' In theory, activities directly related to each other with the same float value can be grouped together and considered as an entity called a "float path." The most famous of these float paths is the "Critical Path" where the float value is equal to zero (or the lowest float value.) Unfortunately, this literal approach sometimes leaves out activities that would ordinarily be considered on the Critical Path simply due to activities being Hammocks, Interruptible, having different work calendars, completed, or due to constraints.

Longest Path to the Rescue

I attended an excellent presentation by Kenji Hoshino at the 2002 AACE Conference that was partially concerned with this very issue [3]. He explained the problems with the concept of float and proposed that we Scheduling

Engineers stop looking at float and start looking at a newer concept called, ‘Longest Path.’

Primavera Systems, Inc., the makers of Primavera Project Planner (P3) scheduling software defines ‘Longest Path’ as the string of directly related activities that comprise the longest path from the data date to the last activity in the schedule [3]. This definition does not concern itself with “float.” It includes activities that might otherwise be left out by the standard definition of Critical Path.

P3 calculates the longest path by identifying the activities that have an early finish equal to the latest calculated early finish for the project. P3 then identifies all driving relationships for these activities and traces them back to the project start date. P3 defines a driving relationship as, ”A relationship between two activities in which the completion of the predecessor activity determines the early dates for the successor activity.” If you have used P3, you have seen the asterisk (“*”) in the predecessor or successor window denoting the fact that a particular relationship was a driving relationship.

Just like the CPM, the Longest Path is a process. It finds the last activity in the schedule. It then travels backward using the driving relationships to identify all activities that are related to the last activity via driving relationships. This list comprises the Longest Path.

In his presentation at the AACE Convention in Portland, Mr. Hoshino said that we should look at the Longest Path and not the Critical Path when managing projects or considering the effects of delays. He went on to say that just like the concept of Near-Critical Activities (those activities with a low float value approaching the Critical Path,) we should also consider looking at Near-Longest Path activities as well. He said that there is only one problem with this concept; there is no known method of determining the Near-Longest Path.

Other experts in this field describe their version of what they would like to see with Longest Path in a different manner [6]. Scheduling and Delay Claims Experts Roger Woodhull and Tom Peters stated that they want to be able to select activities based upon ‘the second longest path’ and the ‘third longest path,’ etc. They are interested in seeing float paths, groupings of activities based upon their contribution to project completion.

The Challenge

The challenge here is to extend the concept of Longest Path so that it can be used by Schedulers much the way float is now used. Schedulers should be able to select, sort, and group activities based upon Longest Path. This concept should be applied to more than just a few activities on the Longest Path, but to all

activities in the schedule just as float is employed. To my mind, this means that we need to invent a method of calculating the Longest Path Value of every activity in the schedule.

To do this, you can't just subtract the Early Finish Date from the Late Finish Date like you do with float. Some other numerical process must be made that mimics the way the Longest Path is computed now.

You Can't Get there from Here

To compute a numerical Longest Path Value, you have to return to the original procedure and try to generalize the process. Starting with the last activity in the network, you note the predecessor activity (or activities) that are "driving' and designate them as being a member of the Longest Path, and then repeat this process until you either run out of driving predecessors or reach the data date.

We are hampered by the fact that we don't have a numerical method for describing the contribution each relationship plays in determining the timing of the successor activity. Until we do, we will never be able to compute the Longest Path Value.

Computing Slack

What do all of the activities on the Longest Path have in common? They all were the source of driving relationships of another activity on the longest path. In other words, driving relationships just exactly fix between the predecessor activity and the successor activity. They cannot have their lag increased by even one day without affecting the timing of the successor activity.

In multi-relationship situations, if a relationship is not 'long enough' then some other relationship will be driving and this 'shorter' relationship will have freedom from constraint. To me, this description sounds a little like the concept of float, only in relationships and not activities. This brings up the question, "Do relationships have "float?"

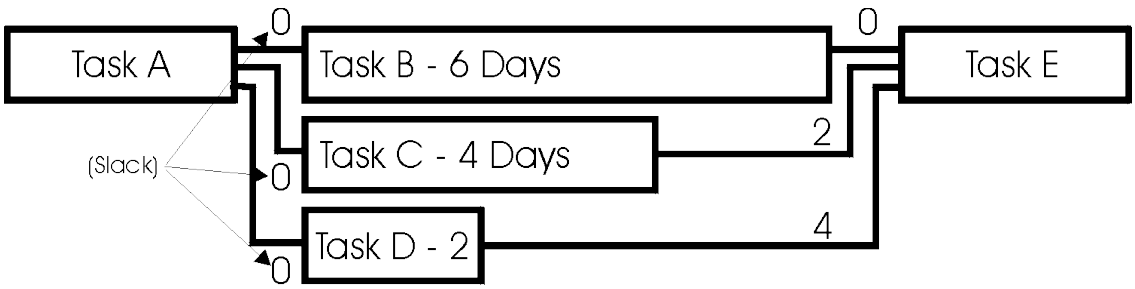
If some relationships are driving and others are not, then they must have their own form of float. Some relationships push the successor activities while others are 'too short.'

I propose that we call the 'float' seen in relationships something else to help us keep this whole thing straight. Why not another term for float that has become disused over the years? I will call the internal 'float' found in relationships between activities, "slack."

slack (slăk) *n*. The amount of inter-activity float a relationship has in relation to other relationships with the same successor activity, measured in the same units as float for the schedule involved.

Slack is the amount of 'unused' time difference between the predecessor and the successor activities. This slack value has nothing to do with the float values of either of the two activities that it relates. It merely indicates how close each predecessor is to becoming the driving relationship for the successor activity.

Perhaps an example will suffice to explain my concept of slack. Note the following CPM fragnet (Figure 1.) Three activities lie in parallel between Task A and Task E. Task B is 6 days in duration; Task C is 4 days in duration; and Task D is 2 days in duration. The numbers outside of the boxes represent the slack of each Finish-To-Start relationship shown. For simplicity, all lags will be assumed to be 0.



[Figure 1] Slack Example

As the three activities in the middle (Tasks B, C, & D) each have only one predecessor (Task A,) the slack for each predecessor relationship must be 0. Each relationship is the controlling operation for each respective activity.

What does it mean when we say that a relationship is a 'Driving Relationship?' We mean that it has a slack of 0. The relationship exactly fills the gap between predecessor and successor. In all cases, there should be one relationship with a slack of 0 (unless an activity constraint overrides this condition.)

Task E, on the other hand, has three predecessors; Tasks B, C, & D. As Task B will take longer to complete than the other two below it, the relationship between Task B and Task E has 0 slack. In other words; it is the driving relationship. The relationship between Tack C and Task E requires 0 duration and has 2 days in which it can complete. The slack of Relationship Task C to Task E must be 2 days. Similarly, as the relationship between Task D and Task E requires 0 days but has 4 days to complete, then its slack value must be 4.

Computing Longest Path Value

Now that we understand the concept of relative float in relationships (now called slack,) let's use an example to show how Near-Longest Path can be calculated. Figure 2 shows a different CPM fragnet that we will use to illustrate our concept. The numbers listed below each activity box are the activity's duration. For simplicity, let's assume that all relationships are Finish-To-Start with 0 lag.

Using standard CPM principles, we calculate the Early Start and Early Finish day numbers. The numbers listed above each activity box are the early start workday number (on the upper left) and the early finish workday number (on the upper right.)

[Figure 2] Forward Pass

Now we need to compute the slack value for every relationship in our fragnet. Figure 3 shows the results of comparing the Early Finish of the predecessor activity with the Early Start of the Successor. The numbers shown between the relationships are the result of evaluating the start of the relationship and its end. This value is called slack and represents how close the relationship is to becoming the driving relationship.

[Figure 3] Slack

I propose that we calculate the longest path value of every activity in a CPM schedule. This value will represent the number of units (be it days or hours) that

each activity has free before being included on the longest path. We can then select activities on the near-longest path by value.

The Longest Path value for each activity is calculated by progressively evaluating each predecessor activity, starting with the finish activity and adding the slack to the successor's Longest Path value to give the current activity's Longest Path value. In the case of multiple successor relationships, if the number derived is less than the current value then the new number is substituted for the old. Then we process every relation until all relationships have been evaluated. What could be easier?

In our example, we start out by assigning the Longest Path value of "0" to the Finish Activity, Task D. We then work backwards through the CPM network adding the current Longest Path value to the slack value to derive the Longest Path value of the predecessor. If the number calculated for slack is higher than an existing value derived from some other relationship, we ignore that calculation and stick with the lower number.

Figure 4 shows the results of this Longest Path process. The numbers shown above each activity represent the Longest Path value for that activity. As expected, the Longest Path falls from Task A, to Task B, to Task C, and then Task D. We know this because their Longest Path value is "0." Task F is the closest activity to the Longest Path without actually being on it with a Longest Path value of "1." Task E has a Longest Path value of "2."

[Figure 4] Longest Path

Just as with the near-critical path, we would pick a value to differentiate 'normal' activities from Near-Longest Path activities and select for this value or less. The major difference between near-critical path activities and Near-Longest Path activities is that all of the activities that near-critical path misses are included in Near-Longest Path.

Implementation of Longest Path Value

The key to computing the longest Path values is the calculation of slack. While the process of computing slack may seem simple enough, its implementation is quite complicated.

The rules for calculating relationship lags are very technical and not widely known. You can't assume that the same activity calendar is being used for both the predecessor and successor activities. I will assume [A] that work day numbers for a relationship are computed using the calendar of the predecessor activity.

You must also take into consideration the lag value. Don't forget that there are four different relationship types with different rules for calculating the lag. Finally there are exceptions to rules like those found with milestone activities.

In the long run, what we need is a computer program to first calculate the slack of every relationship and then progressively calculate the Longest Path value using the calculated slack value.

To test the feasibility of implementing my theories, I developed a program called "Longest Path Software" [5]. It looks at a CPM schedule and computes the slack value for each relationship and saves this to a table. It then calculates the Longest Path value and stores this back in the original schedule under the Custom Data Item called, "PATH".

The Pot Holes

I hit a couple of 'pot holes' on the road to success with this experiment. In particular, activities completed out-of-sequence confused the calculation of Longest Path Value. I wondered how prevalent this out-of-sequence condition really was. In other words, does out-of-sequence activity progress happen often in real life?

To answer this question, I made a random sampling of past and present projects, plus a large number of sample schedules sent to me by others. Of more than 50 projects reviewed, within a couple of months of project start, all of the schedules checked had some sort of combinations of activities completed out of sequence. Now I understood a major reason that schedules are so hard to trace.

Further research indicated that the confusion that I was having was in accepting the actual start and finish dates as listed by P3. A lesser-known rule in calculating the CPM of any schedule with actual status is that you must ignore the actual dates and deal with the completed activity as if it were an unstatused

activity with zero remaining duration. (Only after that is done do you go back and insert the actual dates.)

Using the actual dates reported by P3 did not allow the software to correctly compute the retained logic in instances involving out-of-sequence progress. My program could not overcome this missing information without another step. I determined that I would have to compute the forward pass of the Critical Path Method computation instead of letting the third-party software do it.

This is not as simple of a process as you may remember from school. In the modern world, each activity can have a different calendar to clock progress. This eliminates the simple procedure of computing the entire network using day numbers.

In addition, you must deal with the issue of constraints. Currently, the Longest Path calculation that P3 uses doesn't consider late constraints as the 'backward' portion of the CPM calculation does. For example, Finish-No-Later constraints do not affect the outcome of the Longest Path. But the Longest Path calculation used by P3 does consider early constraints.

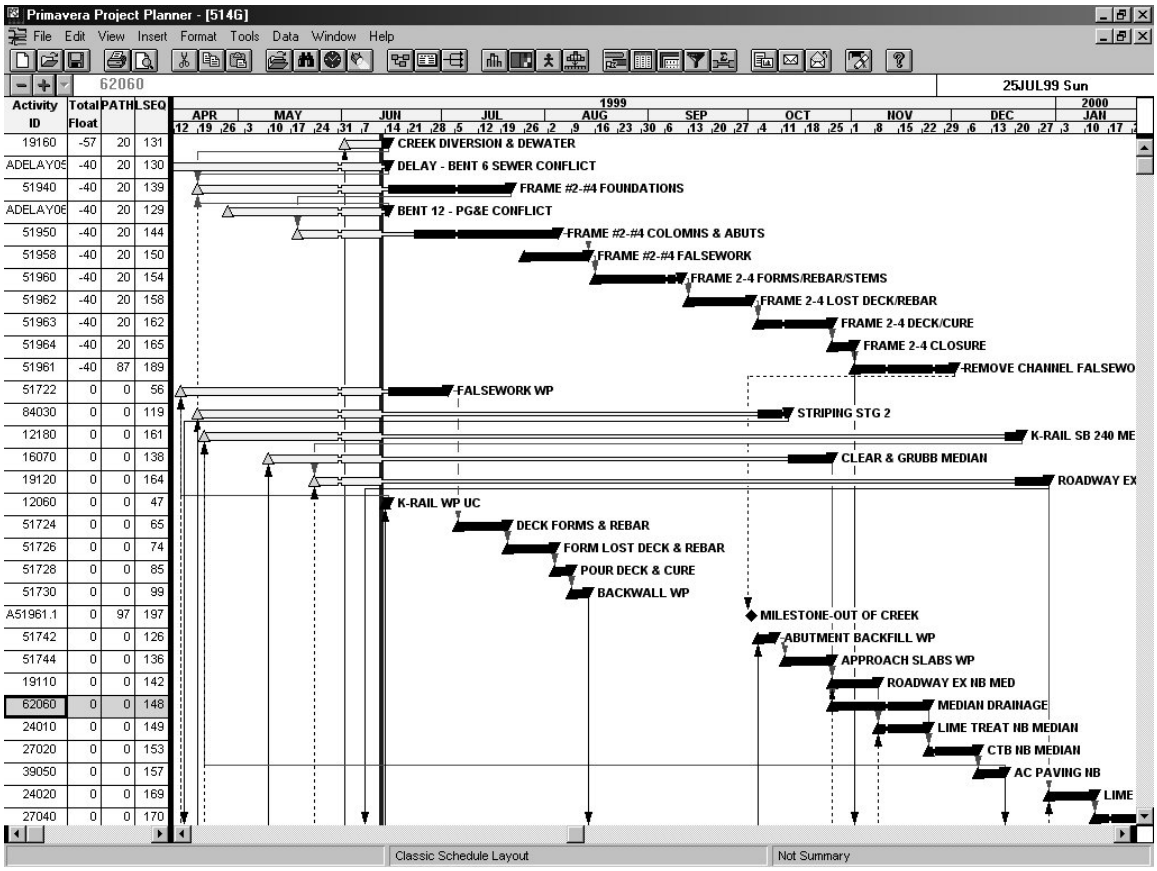
With P3, you can set a Start-No-Earlier-Than constraint on an activity that will force it to be critical and P3 will confirm that the Longest Path begins with that activity and not necessarily starting from the data date. This 'forced' finding seems inherently wrong as the Longest Path should start at the Data Date and proceed to Project Completion regardless of artificial constraints.

The purpose of computing the Longest Path is to identify the work that is controlling the completion of the project. If constraints cause the Longest Path to not function correctly, then they should not be used to determine the Longest Path.

For Longest Path Software, I added the calculation of the early CPM dates to the procedure, but without consideration of constraints. The result is the elimination of the occasional odd inclusion and a cleaner look.

The Results

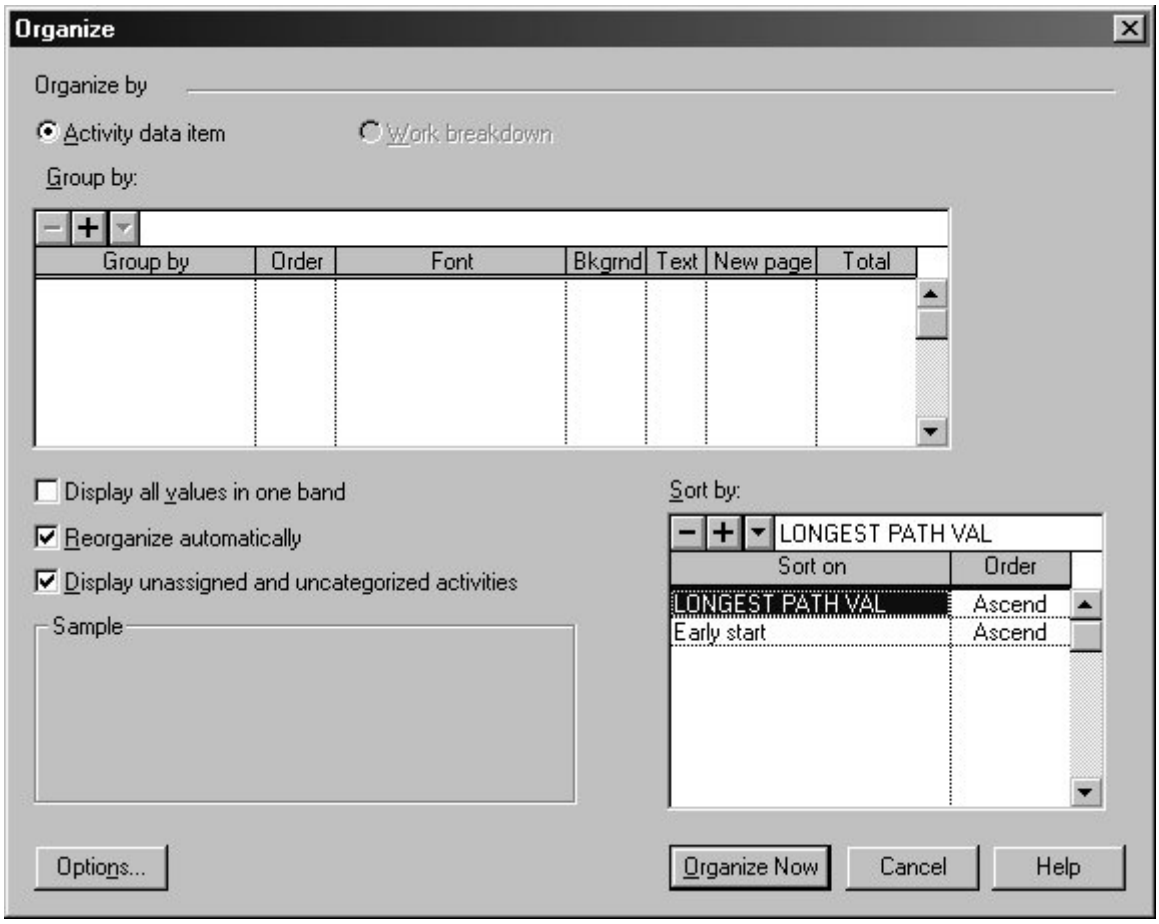
Figure 5 shows a project organized in the standard method of sorting on Total Float and then on Early Start Dates. The first 11 activities have –40 days of float. The remaining activities shown have 0 days of float.



[Figure 5] Example of Schedule before Longest Path

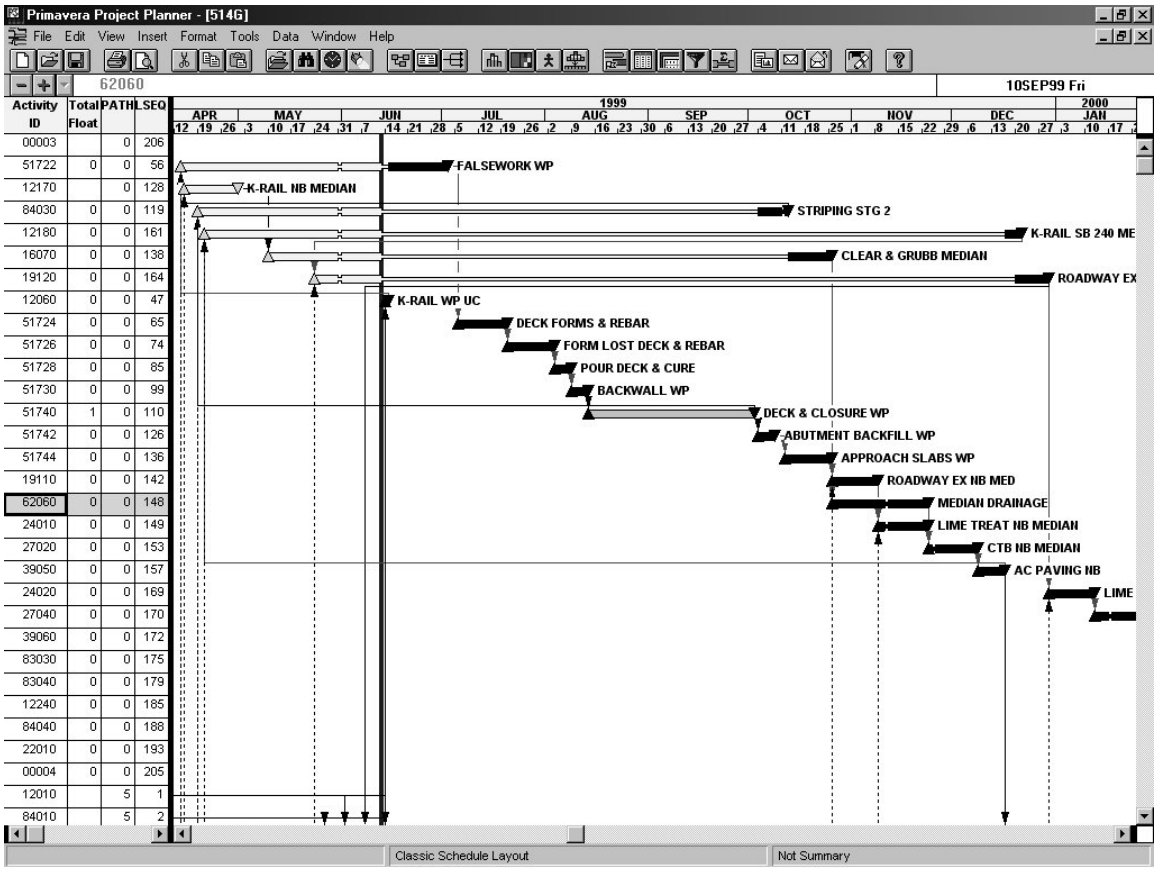
Notice the gaps in the above sequences. Also note that while the layout gives one a wonderful impression of progress as the start of each activity progressively sweeps to the right, the logic lines are, well ‘confused.’

I ran LONGEST PATH Software using this schedule as the subject. It calculated the Longest Path Value of each activity and automatically inserted it back into the schedule as the Custom Data Field, “PATH”. I then returned to P3 and the schedule and selected ‘Organize.’ I clicked on the portion of the Sort by: box containing Total Float and replaced it with Longest Path Value as shown in Figure 6 below.



[Figure 6] P3 Organize Layout Set-up for Longest Path Value

I clicked on the ‘Organize Now’ button and the result is captured in Figure 7 below.



[Figure 7] Example of Schedule after Longest Path

Gone is the entire 11-activity sequence showing –40 days of float. The negative float was caused by a constraint and never had any relation with project completion. Gone also was the gaps in the logic path for float equal to 1. The long activity in the middle was on a different activity calendar and had slightly different float.

Serendipity

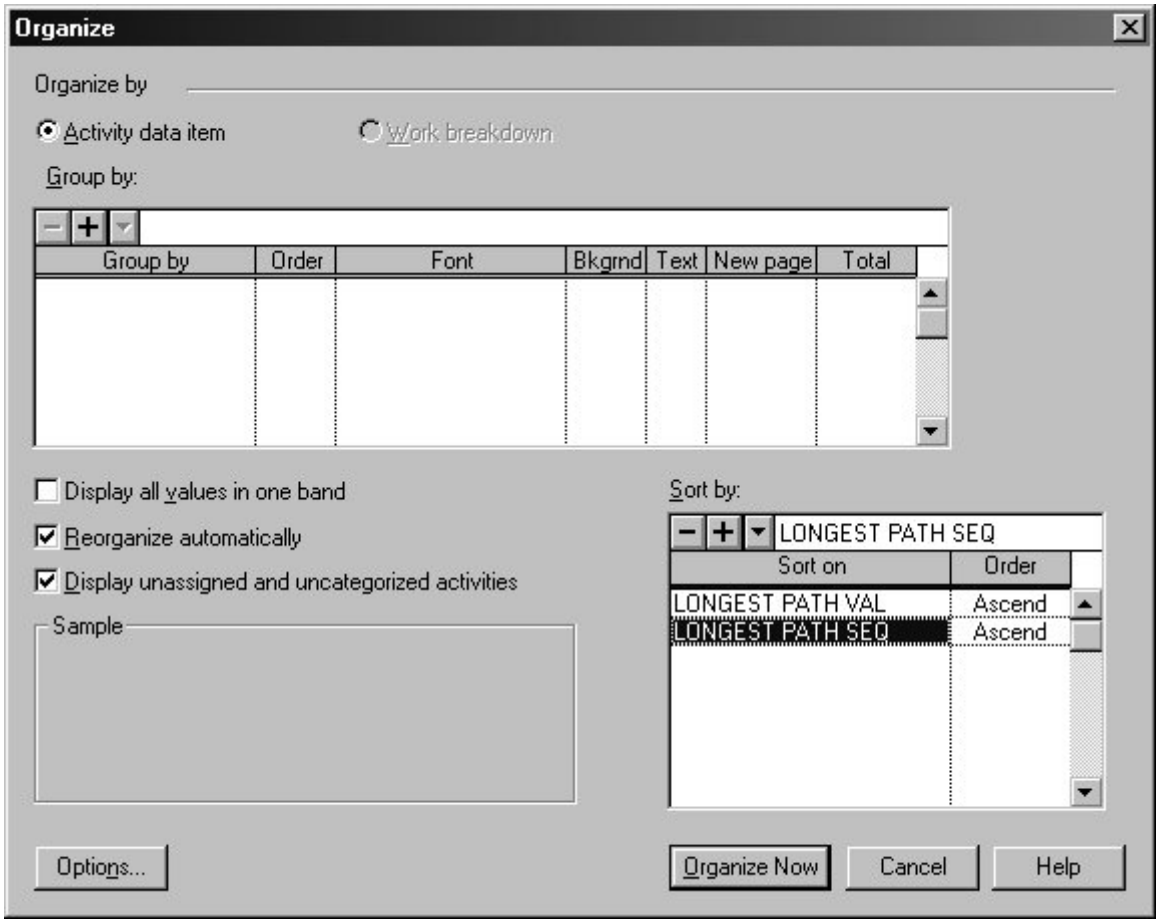
While I was working on computing the Longest Path, something else suggested itself. While Longest Path Value is organized by logical relationships, the secondary sort on Early Start Dates only approximates the logic layout. Now out-of-sequence activities were shown, but in the wrong order. Switching to sorting by Early Finish wouldn't solve timing problems caused by negative lags.

Then it hit me, didn't I just compute the Longest Path value in reverse logical order? Couldn't I capture this order information, reverse the ordering and assign it to each activity as Longest Path Sequence number? Then P3 would know the logical order of each activity and we wouldn't need early start dates.

The value of this sequence number is a little obtuse to spot right away. Consider the manner that Longest Path Value is calculated. Only after all predecessors to an activity have been considered and the lowest Longest Path Value is recorded is that activity eligible to be put in the Longest Path list. This is where the sequence number is recorded. Some times (or even often,) the next activity in a logical chain is not computed immediately after the preceding one. This is because in a network, multiple paths are being considered simultaneously.

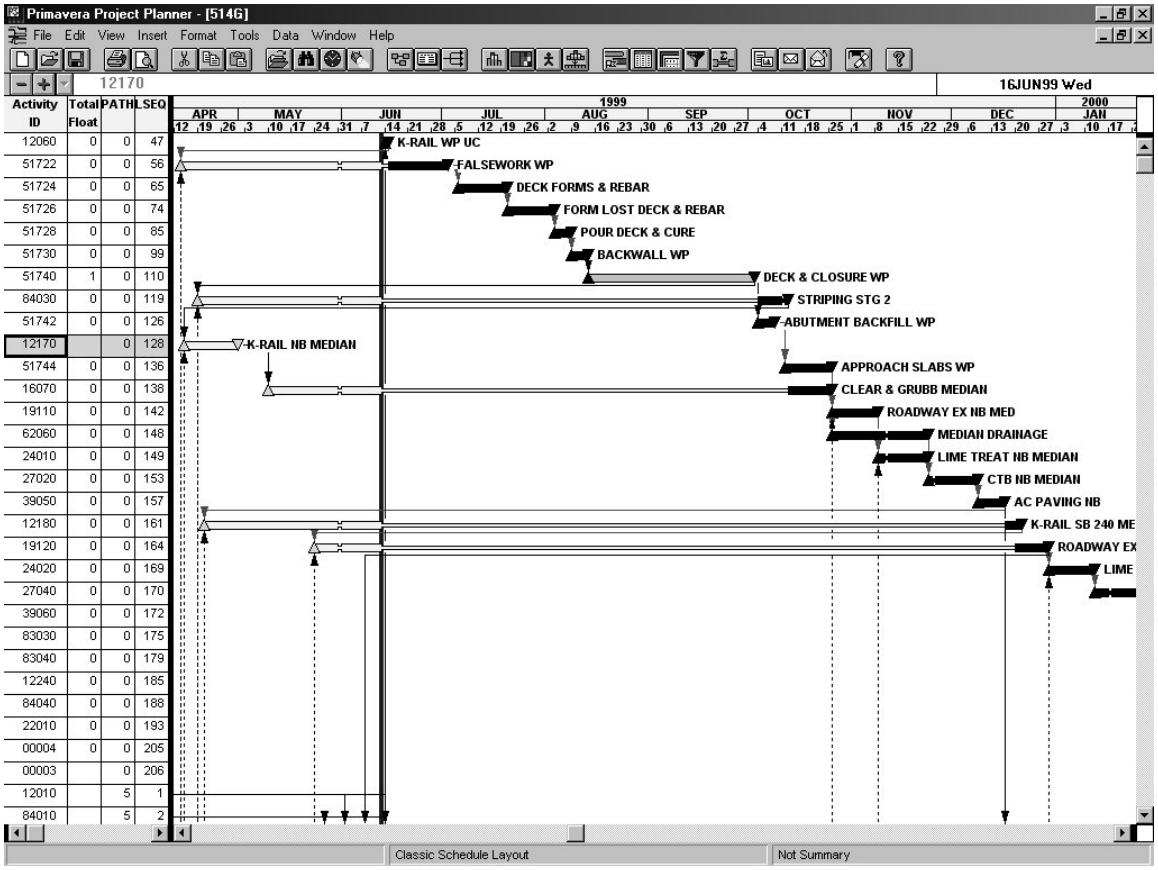
But this 'confusion' doesn't matter. The point is that the Longest Path Value of every predecessor is always located before the successor in the list and thus its sequence number is always less (because it was 'reversed.') The numbering schema for a chain of activities may not be continuous, but they are always in logical order somewhere in that list. By the way, I reversed the order of the Longest Path Sequence list so that activities occurring earlier would have lower numbers, just like dates.

To test this theory, I added the creation of a second Custom Data Item (LSEQ) in the schedule and exported the Longest Path Sequence Number to the schedule at the same time I exported the Longest Path Value. Figure 8 shows the same P3 Organize Window, this time using both of the new criteria.



[Figure 8] P3 Organize Layout Set-up for Longest Path Sequence

When we first sort by Longest Path Value and then by Longest Path Sequence, an amazing metamorphosis occurs. Continuous activity chains are displayed, regardless of status, conditions, or other events. Figure 9 demonstrates the results of replacing Sort by: Early Start with Longest Path Sequence.



[Figure 9] Schedule Sorted by Longest Path Value and Longest Path Sequence Number

How is this result different from sorting by float and then by Early Start Dates? In addition to including activities that normally get left out (as explained earlier,) you now see the planned, logical sequence. Instead of seeing activities that started early apart from their predecessors and successors, you now see them in order with all of their predecessors above then and their successors below. Tracing logic and causality of in-progress work becomes almost too easy.

Unexpected Bonus

While using LONGEST PATH Software, I came across an unexpected bonus. I noted that not only were activities with early starts shown in their logical location, but completed activities in the logic chain were shown right were they belonged all along. Longest Path Value calculations include completed activities!

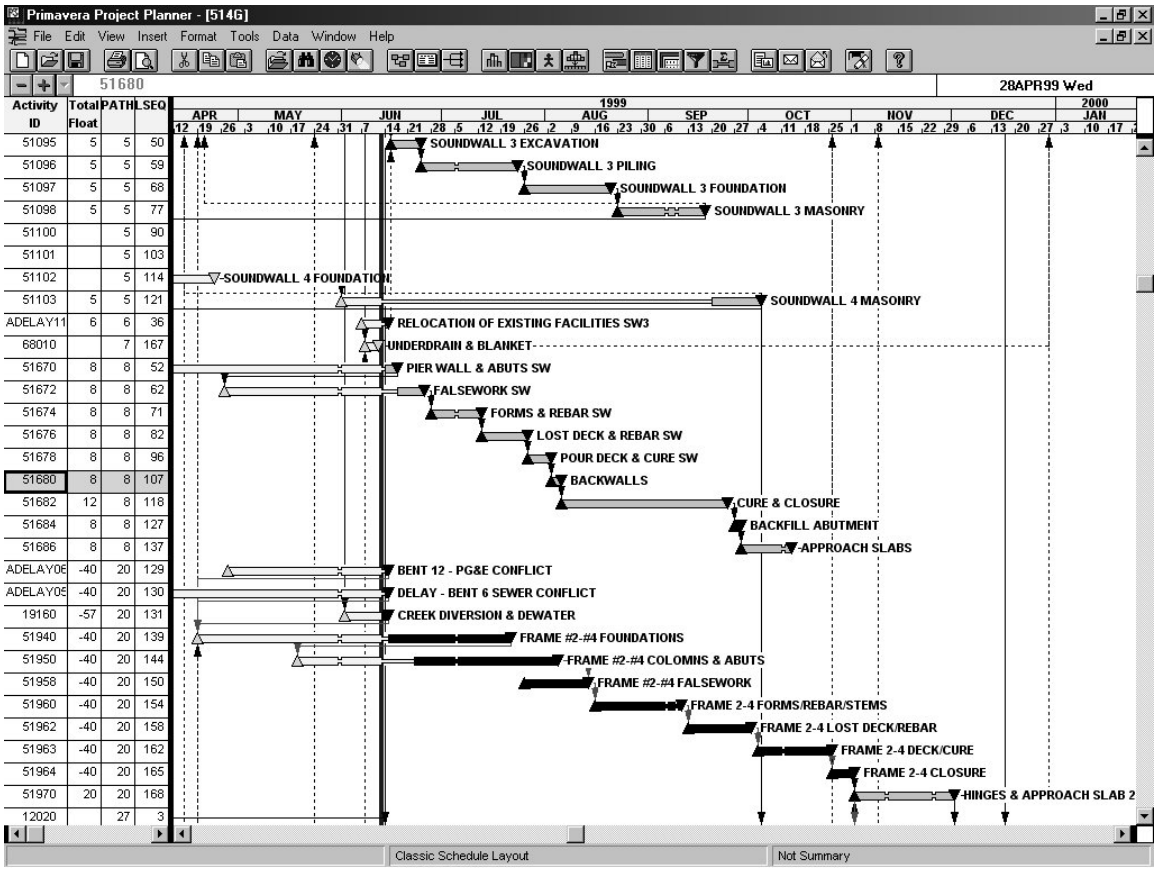
Using float to isolate the critical path, completed activities don't show float and thus do not show up on the same section of the listing as do the uncompleted activities in the same logic path. The 'Seasoned Scheduler' knows the confusion of trying to trace the critical path through activities that were completed out-of-sequence. Just because an activity is complete and does not show any float value doesn't mean that it is not a crucial piece of the critical path.

To trace the critical path through activities completed out-of-sequence, the Scheduler must 'page-down' the screen to the end of the listing. Then he or she will follow the logic back up the screen to the next critical path activity. In fact, most Schedulers do not know that any lags assigned from the completed activity and its successors are still in effect even thought the activity is complete. This is one of the sources of 'Ghost Duration' in schedules that occur from time to time.

Did you see the difference in how activities completed out-of-sequence are displayed in Figure 9 above? The highlighted activity (which is "K-RAIL NB MEDIAN") was completed out-of-sequence but is now displayed after its Longest Path logical predecessor and before its Longest Path logical successor.

Near-Longest Path

At the start of this paper, I said that I was looking to find the "Near-Longest Path." How well did I do? Figure 10 below shows the results of paging-down one screen in our trial schedule.



[Figure 10] The Near-Longest Path

The first chain has a Longest Path Value of 5, the second has a Longest Path Value of 8 and the third has a Longest Path Value of 20. Notice that the third chain contains members of the -40 float group.

Final Addition

I added one last 'bell and whistle' to the software to correct for one of the most glaring errors that the P3 implementation of Longest Path makes. P3 assumes that the last activity in the schedule (the one with the latest Early Finish Date) is the end of your Longest Path. This isn't true on many real-world construction projects.

On many projects there are specified requirements in addition to project completion. Sometimes there is a landscape maintenance period. Sometimes the submittal of As-Builts, equipment documentation, or other paperwork is listed after project completion. Sometimes there is off-site work, removal and reclamation of replaced equipment or materials or even non-essential punch-list work to be performed after the formal project completion. None of these should be considered when determining the Longest Path.

In short, Substantial Completion should define the end of the Longest Path. I added the option in my software to designate a Substantial Completion Activity. This activity would then define the anchor of the Longest Path and the basis for all Longest Path Values. Any activity that occurred after this activity would be removed from the Longest Path.

Principles of Longest Path Value

The CPM 'Forward Pass' is made without consideration of the activity's completion status, only considering remaining duration. The computed early start and early finish dates are retained for all activities.

No activity will be scheduled to occur before the current schedule data date.

No logical relationships are ignored, even those to completed activities.

The CPM 'Forward Pass' is made without consideration of constraints.

All activity calendar assignments are completely observed.

Slack is computed using the calendar of the predecessor activity.

Conclusion

CPM Theory is over 50 years old and is still evolving today. For proof of this I offer the new concepts of "Slack," "Longest Path Value," "Longest Path Sequence," and the "Longest Path Substantial Completion Activity."

Using Longest Path Value gives the Scheduler a means of looking at the criticality of the schedule by considering a chain of activities and not just individual activities. This process is an improvement over using float in that it doesn't overlook activities with different calendars, interruptible activities, out of sequence work, or even completed activities. Results are not dependent upon <u>any</u> CPM calculation rules settings.

A general rule of thumb is that you can use Longest Path Value anywhere you used to use float. You can use Longest Path Sequence anywhere you would use Early Start Dates or Early Finish Dates.

Using Longest Path Sequence further reinforces the trend of organizing activities by their logical sequence instead of using dates. When used with Longest Path Value, Longest Path Sequence properly places activities with out-of-sequence

progress and complete activities where they logically belong in the sequence listing, making logic easy to trace.

The concept of designating a Longest Path Substantial Completion Activity in your schedule adds intelligence and meaning to individual schedules. Directing the Longest Path is just the start of the usefulness of this concept. Many a project was ruined by the failure to consider what constituted the end of the project.

Footnotes

[A] Primavera Project Planner (P3) by Primavera, Inc. uses the rule that the predecessor activity's calendar is used in lag duration calculations. Primavera Enterprise (P3e/c) by Primavera, Inc. has several possible settings and used the successor activity's calendar by default before Version 3.5 and now adopts the P3 setting as a default. Project by Microsoft, Inc. uses the default calendar for all lag calculations.

References:

[1] Spinner. Elements of Project Management, M. Spinner, copyright 1981 by Prentice-Hall, Inc. Englewood Cliffs, New Jersey 07632, Page 3.

[2] Bates. "Scheduling Techniques," by Jennifer Bates, CCE. AACE International Skills & Knowledge of Cost Engineering, Chapter 9. ISBN: 1-885517-36-X.

[2] Hoshino. "Catching the Elusive As-Built Critical Path," by Kenji P. Hoshino. 46th Annual Meeting of AACE International at Portland, June 25, 2002, CDR.08.

[4] Primavera. Primavera Project Planner Reference Manual, Version 3.0. Primavera Systems, Inc., Three Bala Plaza West, Bala Cynwyd, Pennsylvania.

[5] Winter. Longest Path Software. Copyright 2002 by Ron Winter Consulting.

[6] Woodhull. "Claims Avoidance, An Alternative Approach," Primavera Users Conference, San Diego, October 23, 2002 – Roger Woodhull, PE, JD and Tom Peters, PE.