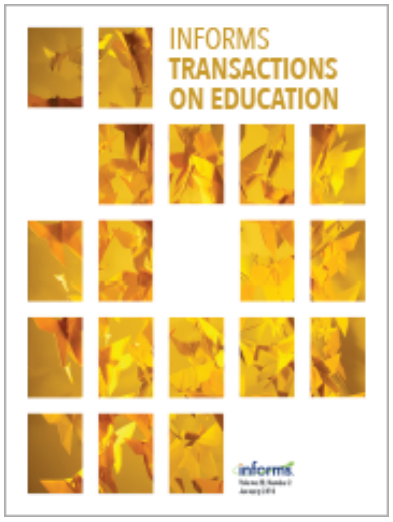


INFORMS Transactions on Education

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A New Approach to Implementing Project Networks in Spreadsheets

Cliff T. Ragsdale,

To cite this article:
Cliff T. Ragsdale, (2003) A New Approach to Implementing Project Networks in Spreadsheets. INFORMS Transactions on Education 3(3):76-85. https://doi.org/10.1287/ited.3.3.76

A New Approach to Implementing Project Networks in Spreadsheets

Cliff T. Ragsdale
Department of Business Information Technology
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061, USA

crags@vt.edu

ABSTRACT

ISeveral approaches have been proposed for implementing project networks in spreadsheets. The approaches suggested to date are overly tedious to implement for problems of realistic size and/or require error-prone manual intervention if precedence relations among activities change or project activities are added or deleted. This paper introduces a new and more efficient approach to implementing project networks in spreadsheets that overcomes the weaknesses found in previous techniques.

Editor's note: This is a pdf copy of an html document which resides at
http://ite.pubs.informs.org/Vol3No3/Ragsdale/

1 INTRODUCTION

Project management (PM) has long been one of the standard topics included in introductory operations research/management science (OR/MS) texts and courses. Indeed, one could argue that PM is among the more important topics covered in OR/MS texts as virtually every business student will become responsible for managing a project at some point in his or her career. It could also be argued that the typical introductory PM presentation is the most simple topic in these books only requiring that students understand predecessor relations among activities (comparable to the prerequisite course requirements familiar to all college students) and be able to add and subtract expected activity durations.

Over the past decade, spreadsheets have become the most popular software tool for teaching introductory OR/MS topics. Ironically, while a spreadsheet seems to offer the perfect environment in which to carry out the simple early and late start and finish time calculations for each activity in a project network, an efficient way to implement these calculations has proven quite elusive. This paper introduces a new, efficient approach for implementing project networks in a spreadsheet using array formulas and circular references.

2 MODELING PROJECT NETWORKS: THE STANDARD APPROACH

Seal (2001) recently noted that the standard approach for solving PM problems in spreadsheets (Moore et al, 2001; Hillier et al, 2000; Ragsdale, 2001) does not permit easy extendibility of the model when activities are added or deleted or when predecessor relationships are altered. As an example, consider the project shown in Figure 1 (adapted from chapter 14 of Moore et al (2001)) representing the physical relocation of an organization's administrative operations.

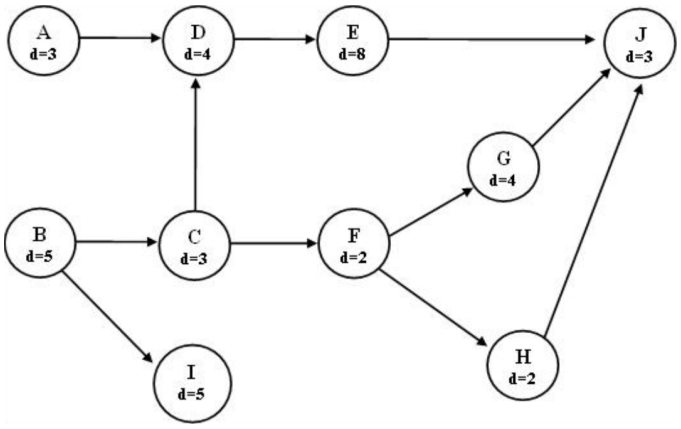
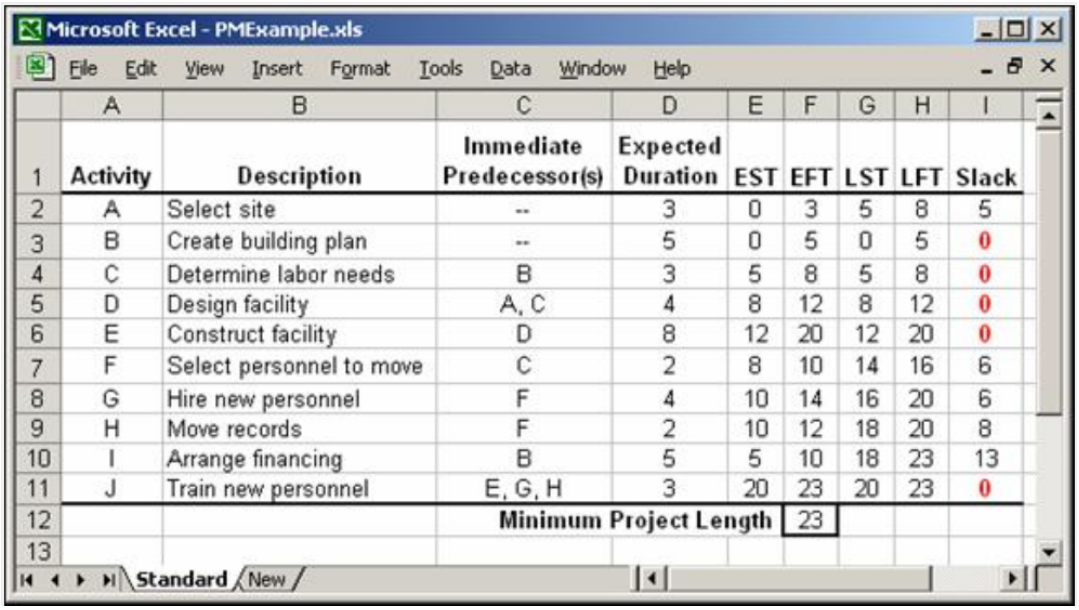


Figure 1: Description of example problem.

  ISSN: 1532-0545

Activity	Description	Immediate Predecessor(s)	Expected Duration
A	Select site	– –	3
B	Create building plan	– –	5
C	Determine labor needs	B	3
D	Design facility	A, C	4
E	Construct facility	D	8
F	Select personnel to move	C	2
G	Hire new personnel	F	4
H	Move records	F	2
I	Arrange financing	B	5
J	Train new personnel	E, G, H	3

The standard approach proposed by Moore et al (2001) and others for calculating the early start time (EST), early finish time (EFT), late start time (LST), late finish time (LFT), and slack for each activity is shown in Figure 2. [Click here to download the Excel file for this example. Note: This file contains a macro that turns on Excel's Iteration calculation option. Before opening this file, enable Excel's medium macro security setting by clicking: Tools, Macro, Security, Medium.] This technique requires 24 separate spreadsheet formulas to determine the critical path for a project involving only ten activities. Also note that many of the MAX( ) and MIN( ) functions listed for columns E and H contain a single argument and could be simplified by using a direct reference to the cell in question (e.g., an alternate formula for cell E4 is "=F3"). However, the use of the MAX( ) and MIN( ) functions (even when not necessary) offers a useful cue to remind students that for each activity listed, the value in column E represents the MAXimum EFT of its immediate predecessor activities and the value in column H represents the MINimum LST of its immediate successor activities.



	A	B	C	D	E	F	G	H	I
1	Activity	Description	Immediate Predecessor(s)	Expected Duration	EST	EFT	LST	LFT	Slack
2	A	Select site	--	3	0	3	5	8	5
3	B	Create building plan	--	5	0	5	0	5	0
4	C	Determine labor needs	B	3	5	8	5	8	0
5	D	Design facility	A, C	4	8	12	8	12	0
6	E	Construct facility	D	8	12	20	12	20	0
7	F	Select personnel to move	C	2	8	10	14	16	6
8	G	Hire new personnel	F	4	10	14	16	20	6
9	H	Move records	F	2	10	12	18	20	8
10	I	Arrange financing	B	5	5	10	18	23	13
11	J	Train new personnel	E, G, H	3	20	23	20	23	0
12			Minimum Project Length			23			
13									

Figure 2: The standard approach to project network implementation.

The spreadsheet in Figure 2 computes all the values correctly. However, numerous manual changes to formulas are required if project activities are added or deleted, or if predecessor relations among the activities are altered (in column C). Clearly, this approach to implementing project networks in a spreadsheet is tedious, error-prone, and inflexible and these issues only become more pronounced as the number of activities increase. As a result, this approach, while educational, is impractical for all but the smallest of PM problems.

Seal (2001) proposed a new approach to implementing a project network in a spreadsheet that overcomes many of the scalability and flexibility problems associated with the standard approach shown in Figure 2. However, that approach requires one to create a matrix of predecessor relationships for the project network and involves other complexities that make it rather cumbersome to implement.

Conway and Ragsdale (1997) describe a number of design guidelines for developing effective spreadsheet models. A key principle noted is that a spreadsheet design that results in formulas that can be copied is probably better than one where every formula is unique. A model with formulas that can be copied to complete a series of calculation in a range is less prone to error (or more reliable) and tends to be more understandable (or auditable) because once the user understands the first formula in a range, he/she understands all the formulas in a range. The calculations listed for columns E and H in Figure 2 almost beg to be replaced with more generic formulas for implementing the logic behind the EST and LFT calculations.

The remainder of this paper presents a brief overview of array formulas and then shows a new approach to implementing project networks that uses array formulas and circular references to calculate the EST and LFT for each activity. This allows one to implement a project network by entering only five simple formulas regardless of the number of activities in the network. The resulting spreadsheet model is updated automatically if precedence relations change. This model also makes it easy to add or delete project activities.

3 AN ARRAY FORMULA PRIMER

An array formula can perform multiple calculations using a range of cells and then return either a single result or multiple results. Array formulas are created in the same way as other formulas, except one presses [Ctrl]+[Shift]+[Enter] to enter the formula. Array formulas are one of the most powerful and least understood features in Excel. To understand array formulas better, consider the examples in Table 1 that show how various array formulas can be used to perform the same operations as standard Excel functions.

Table 1: Examples of array formulas for several standard Excel functions.

Standard Excel Function	Equivalent Array Formula
=SUMPRODUCT(E2:E11,F2:F11)	=SUM(E2:E11*F2:F11)
=SUMIF(E2:E11,”>10”,F2:F11)	=SUM(IF(E2:E11>10,F2:F11))
=COUNTIF(E2:E11,”>0”)	=SUM(IF(E2:E11>0,1))
=SUMXMY2(E2:E11,F2:F11)	=SUM((E2:E11−F2:F11)^2)
=VARP(E2:E11)	=AVERAGE((E2:E11−AVERAGE(E2:E11))^2)

As the examples in Table 1 illustrate, some array formulas are more obscure than their Excel function counterparts. However, when no standard Excel function exists to carry out a desired operation on a range of cells, array formulas can be quite useful. For example, in Figure 2, suppose you want to find the maximum value in cells F2 through F11 where the corresponding value in cells E2 through E11 is less than 15. Your instincts may tell you to use Excel’s MAXIF( ) function but no such function currently exists! However, we can calculate the desired result using the array formula:

$$= MAX(IF(E2:E11 < 15, F2:F11))$$

The embedded IF( ) function in this formula first re-

turns the values in the range F2 through F11 where the corresponding values in E2 through E11 are less than 15. The MAX( ) function then returns the largest of the values returned by the IF( ) function.

Similarly, suppose you want to find the minimum value in cells F2 through F11 where the corresponding value in cells E2 through E11 is greater than 15. There is no MINIF( ) function in Excel, but we can calculate the desired result using the array formula:

$$= MIN(IF(E2:E11 > 15, F2:F11))$$

Remember that one must press [Ctrl]+[Shift]+[Enter] to enter array formulas. Note that Excel automatically inserts curly brackets (i.e., "{}") around an array formula. One should not attempt to type the curly brackets as part of an array formula. Also note that if one double-clicks (or edits) a cell containing an array formula, one must press [Ctrl]+[Shift]+[Enter] again to ensure the entry continues to operate as an array formula (or press the escape key to instruct Excel to ignore any changes to the cell).

All the array formulas mentioned above return a single value as their result. Array formulas can also be used to return a matrix of values (e.g., see Excel's LINEST( ) function for regression). Additional background information about array formulas may be found at:

- http://www.decisionmodels.com/optspeedj.htm
- http://www.emailoffice.com/excel/arrays-bobumlas.html
- http://www.cpearson.com/excel/array.htm

4 MODELING PROJECT NETWORKS: A NEW APPROACH

The calculation of the EST and LFT values for a project network can be accomplished easily using array formulas as shown in Figure 3. Note that the array formulas used here create circular references in the workbook. A circular reference occurs when the value in a cell depends on the value in another cell that, in turn, is dependent on the value in the original cell. Often, a circular reference in a workbook indicates a formula contains an error and Excel displays a dialog box warning you about this possibility. However, there are occasions when a circular reference is exactly what is needed to accomplish a particular task. This is such an occasion. To tell Excel that we intend to use circular references:

1. Click Tools, Options.
2. Click the Calculation tab.
3. Select the Iteration option.
4. Click OK.

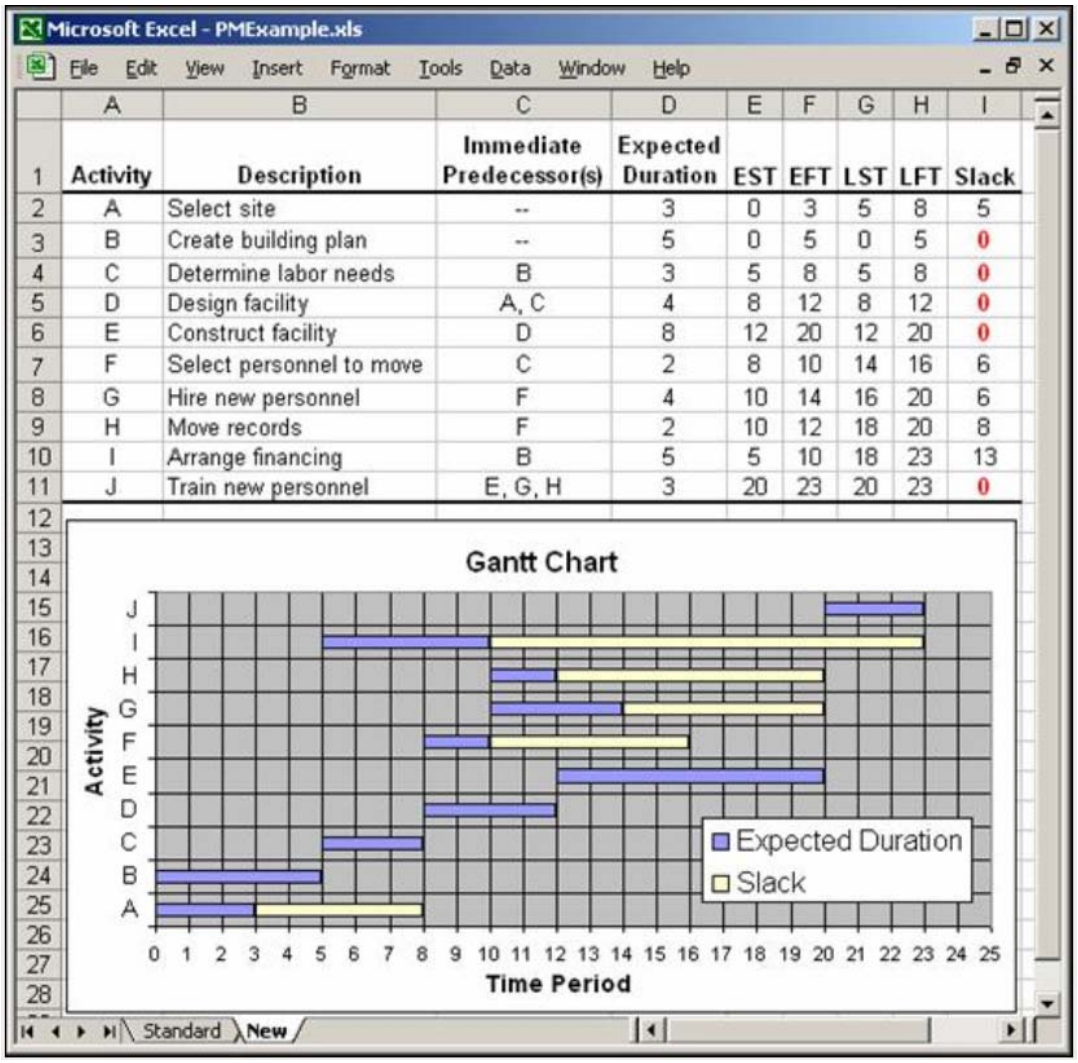
Key Cell Formulas

Cell	Formula	Copy to
E2	=MAX(IF(ISERR(FIND($A$2:$A$11,C2)),0,$F$2:$F$11))	E3:E11
	(Press [Ctrl]+[Shift]+[Enter] to enter.)	
F2	=E2+D2	F3:F11
G2	=H2−D2	G3:G11
H2	=MIN(IF(ISERR(FIND(A2,$C$2:$C$11)),MAX($F$2:$F$11),$G$2:$G$11))	H3:H11
	(Press [Ctrl]+[Shift]+[Enter] to enter.)	
I2	=G2−E2	I3:I11

In Figure 3, each cell in column E must implement the logic to calculate the ESTs. As mentioned earlier, for each activity, this involves determining the maximum EFT for the activities that immediately precede it. This is accomplished by the following array formula in cell E2 (copied to cells E3 through E11):

$$= MAX(IF(ISERR(FIND(\$A\$2:\$A\$11, C2)), 0, \$F\$2:\$F\$11))$$

This array formula is comprised of four nested functions (i.e., FIND( ), ISERR( ), IF( ), and MAX( )) that execute a "loop" or a series of repeated calculations. Table 2 outlines the operation of this array formula in determining the EST for activity D (cell E5 in Figure 3).



	Activity	Description	Immediate Predecessor(s)	Expected Duration	EST	EFT	LST	LFT	Slack
2	A	Select site	--	3	0	3	5	8	5
3	B	Create building plan	--	5	0	5	0	5	0
4	C	Determine labor needs	B	3	5	8	5	8	0
5	D	Design facility	A, C	4	8	12	8	12	0
6	E	Construct facility	D	8	12	20	12	20	0
7	F	Select personnel to move	C	2	8	10	14	16	6
8	G	Hire new personnel	F	4	10	14	16	20	6
9	H	Move records	F	2	10	12	18	20	8
10	I	Arrange financing	B	5	5	10	18	23	13
11	J	Train new personnel	E, G, H	3	20	23	20	23	0

Figure 3: Implementing the project network using array formulas.

To evaluate the array formula for cell E5, Excel starts with the innermost function and substitutes each of the values in cells A2 through A11 as the first argument (denoted by x in Table 2) in the FIND( ) function. In general, the function FIND(x, y) attempts to find the text string denoted by x within the text string denoted by y and, if found, returns the starting position of x within y; otherwise, it returns the Excel error code "#VALUE!". Thus, in Table 2 the FIND( ) function returns the value 1 when cell A2 (containing "A") is compared to cell C5 (containing "A, C"). Similarly, the FIND( ) function returns the value 4 when cell A4 (containing "C") is compared to cell C5 (containing "A, C"). In every other case, the FIND( ) function returns the error code "#VALUE!" because none of the other activities in cells A2 through A11 appear in cell C5 as immediate predecessors of activity D.

The result of each FIND( ) function is then evaluated by the ISERR( ) function which returns a Boolean value of TRUE if its argument evaluates to an error code or, otherwise, returns the value FALSE. The IF( ) function then uses this Boolean result to determine whether to return a value of zero or the corresponding value in the range F2 through F11. For instance, because the value in cell A2 ("A") is found in cell C5 as an immediate predecessor of activity D, the ISERR( ) function returns the value FALSE and the IF( ) function returns the corresponding EFT value from the range F2 through F11 (i.e., the value 3 from cell F2).

Formula for cell E5:
=MAX(IF(ISERR(FIND($A$2:$A$11,C5)),0,$F$2:$F$11))

Array Value x	**FIND(x ,C5)**	**ISERR( )**	**IF( )**
A2	1	FALSE	3
A3	#VALUE!	TRUE	0
A4	4	FALSE	8
A5	#VALUE!	TRUE	0
A6	#VALUE!	TRUE	0
A7	#VALUE!	TRUE	0
A8	#VALUE!	TRUE	0
A9	#VALUE!	TRUE	0
A10	#VALUE!	TRUE	0
A11	#VALUE!	TRUE	0
			MAX( ) = 8

Alternatively, because the value in cell A3 (”B”) is not found in cell C5 as an immediate predecessor of activity D, the ISERR( ) function returns the value TRUE and the IF( ) function returns the value zero.

As shown in Table 2, the IF( ) function produces a series of values that are either zero or the EFTs of activities that are immediate predecessors of activity D. The MAX( ) function then returns the largest of these values (i.e., 8) which, of course, represents the EST of activity D. Note that if an activity has no immediate predecessors (e.g., activities A and B) then all the values returned by the IF( ) function will be zero and, as a result, the EST for such an activity will also be zero.

Now consider the array formulas entered in column H of Figure 3 to implement the logic to calculate the LFTs. For each activity, this involves determining the minimum LST for the activities that immediately follow (or succeed) it. This is accomplished by the following formula in cell H2 (copied to cells H3 through H11):

$$= MIN(IF(ISERR(FIND(A2,\$C\$2:\$C\$11)),MAX(\$F\$2:\$F\$11),\$G\$2:\$G\$11))$$

This array formula is also comprised of several nested functions that execute a ”loop” or a series of repeated calculations. Table 3 outlines the operation of this array formula in determining the LST for activity C (cell H4 in Figure 3).

To evaluate the array formula for cell H4, Excel starts with the innermost function and substitutes each of the values in cells C2 through C11 as the second argument (denoted by y in Table 3) in the FIND( ) function. Thus, in Table 3, the FIND( ) function returns the value 4 when cell A4 (containing ”C”) is compared to cells C5 (containing ”A, C”) and the value 1 when cell A4 is compared to C7 (containing ”C”). In every other case, the FIND( ) function returns the error code ”#VALUE!” since the activity in cell A4 is not listed as an immediate predecessor in any of the other cells in the range C2 through C11.

The result of each FIND( ) function is then evaluated by the ISERR( ) function which again simply returns a Boolean value of TRUE or FALSE. The IF( ) function then uses this Boolean result to determine whether to return the maximum EFT from column F (representing the latest completion time of the project) or the corresponding value in the range G2 through G11. For instance, because the value in cell A4 (”C”) is found in cell C5 as an immediate predecessor of activity D, the ISERR( ) function returns the value FALSE and the IF( ) function returns the corresponding LST value from the range G2 through G11 (i.e., the value 8 from cell G5). Alternatively, because the value in cell A4 (”C”) is not found in cell C6 as an immediate predecessor of activity E, the ISERR( ) function returns the value TRUE and the IF( ) function returns the value 23 (i.e., MAX(F2:F11)).

Table 2: Summary of the LFT calculation for activity C (cell H4).

Formula for cell H4:
=MIN(IF(ISERR(FIND(A4,$C$2:$C$11)),MAX($F$2:$F$11),$G$2:$G$11))

Array Value y	**FIND(A4, y)**	**ISERR( )**	**IF( )**
C2	#VALUE!	TRUE	23
C3	#VALUE!	TRUE	23
C4	#VALUE!	TRUE	23
C5	4	FALSE	8
C6	#VALUE!	TRUE	23
C7	1	FALSE	14
C8	#VALUE!	TRUE	23
C9	#VALUE!	TRUE	23
C10	#VALUE!	TRUE	23
C11	#VALUE!	TRUE	23
			MIN( ) = 8

As shown in Table 3, the IF( ) function produces a series of values that are either the maximum EFT for the project or the LSTs of activities that immediately follow (or succeed) activity C. The MIN( ) function then returns the smallest of these values which, of course, represents the LFT of activity C. Note that if an activity has no immediate successors (e.g., activities I and J) then all the values returned by the IF( ) function will be the maximum EFT of the project and, as a result, the LFT for such an activity will also be the maximum EFT of the project.

The final results in Figure 3 are, of course, identical to those calculated in Figure 2. However, if we need to change any of the predecessor relations in column C, the spreadsheet in Figure 3 automatically updates the results to reflect these changes no manual changes are required to the formulas.

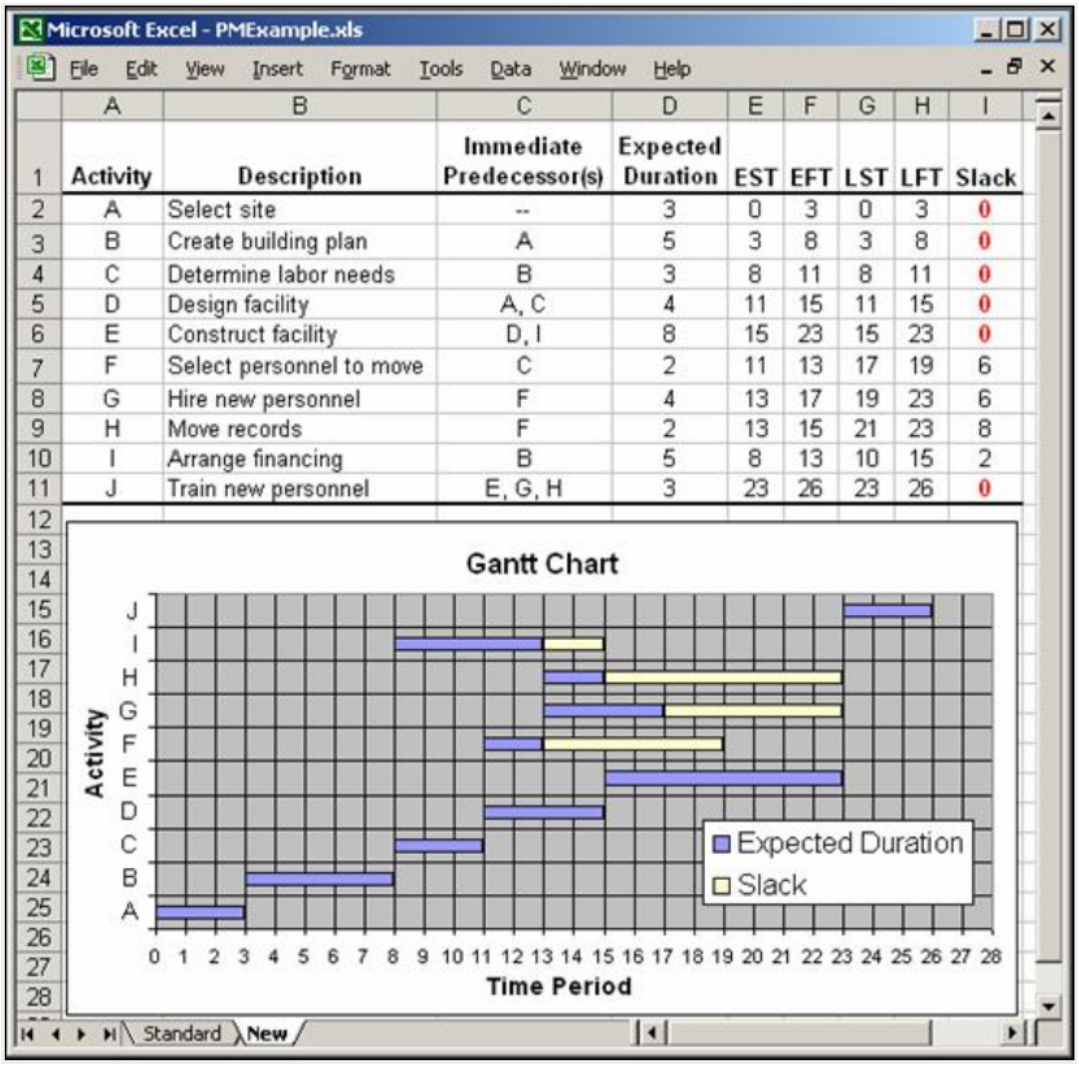
For example, suppose we want to make activity A (select site) a predecessor of activity B (create building plan), and activity I (arrange financing) a predecessor of activity E (construct facility). Upon making the necessary changes to the predecessors in column C, the spreadsheet automatically adjusts for these changes and presents the solution shown in Figure 4. The technique illustrated here also makes the addition or deletion of project activities a trivial task.

5 AN IMPORTANT IMPLEMENTATION ISSUE

The technique presented above relies on the (case-sensitive) FIND( ) function to identify immediate predecessor and successor activities when computing ESTs and LFTs for each activity. Recall that the function FIND(x, y) attempts to find the text string denoted by x within the text string denoted by y. As a result, it is critically important to use activity labels that are unique and do not appear as substrings within other activity labels.

For example, the 26 letters of the English alphabet may be used to uniquely identify up to 26 activities in a project. However, using the strings ”A1” and ”A11” as activity labels would not work in the proposed technique because the FIND( ) function would locate ”A1” within ”A11” (i.e., FIND(”A1”,”A11”)=1) erroneously identifying activity A11 as a predecessor or successor of activity A1. Fortunately, use of the strings ”A01” and ”A11” as activity labels easily remedies this situation.

Similarly, if one wishes to use numbers rather than letters to identify activities, using the numbers 1, 2, 3, ..., 9 as activity labels would create matching problems within activity labels 11, 12, 13, ..., 19 (among



	Activity	Description	Immediate Predecessor(s)	Expected Duration	EST	EFT	LST	LFT	Slack
2	A	Select site	--	3	0	3	0	3	0
3	B	Create building plan	A	5	3	8	3	8	0
4	C	Determine labor needs	B	3	8	11	8	11	0
5	D	Design facility	A, C	4	11	15	11	15	0
6	E	Construct facility	D, I	8	15	23	15	23	0
7	F	Select personnel to move	C	2	11	13	17	19	6
8	G	Hire new personnel	F	4	13	17	19	23	6
9	H	Move records	F	2	13	15	21	23	8
10	I	Arrange financing	B	5	8	13	10	15	2
11	J	Train new personnel	E, G, H	3	23	26	23	26	0

Figure 4: Solution with revised predecessor relations.

others). However, this can be avoided easily by applying Excel's "Text" format to cells containing activity labels and immediate predecessors (i.e., columns A and C in Figure 3) and using two-digit numbers for all activity labels (e.g., 01, 02, 03, ..., 09, 10, 11, 12, 13, ..., 99). If more than 100 numeric activity labels are needed, three−digit numbers (formatted as text) should be used.

6 CREATING GANTT CHARTS IN EXCEL

The Gantt charts shown in Figures 3 and 4 are actually stacked horizontal bar charts that plot each activity's EST, expected activity duration, and slack. However, the bars for the EST for each activity are formatted so that they do not appear on the graph creating the illusion that only the expected activity durations and slacks are being graphed. To create a Gantt chart like the ones shown here:

1. Click cell A1. While pressing the left mouse button on cell A1, press and hold down the Ctrl key and select the ranges A1:A11, D1:E11, and I1:I11.
2. Click the Insert menu.
3. Click Chart.
4. Select the Bar chart type and the Stacked Bar chart subtype.

5. Click Next.

The Excel Chart Wizard then prompts you to make a number of selections concerning the type of chart you want and how it should be formatted. When finished with the Chart Wizard, you need to change the order in which the bars are displayed on the graph. To do this:

1. Double-click on any of the bars in the chart.
2. Click the Series Order tab.
3. Click EST.
4. Click Move Up (so the Series order is: EST, Expected Duration, Slack).
5. Click OK.

Finally, double-click any of the bars representing the EST of any activity, select the Patterns tab, and indicate that you want no border or area patterns. The basic Gantt chart is now complete. You can continue to customize the chart by clicking any element and specifying options in the resulting dialog boxes.

7 OTHER PM ISSUES

Seal (2001) observes that some authors avoid the difficulties illustrated in Figure 2 by formulating the project network as a linear programming (LP) problem and then solving this LP through the Solver module available in spreadsheets (Plane, 1994; Hesse, 1997; Ragsdale, 2001). The LP approach provides accurate results, scales well to problems of realistic size, and facilitates the discussion of project crashing commonly found in introductory treatments of PM. Unfortunately, the MAX( ) and MIN( ) functions in the models shown in Figure 2 and 3 create nonlinear and nonsmooth relations that lead to local optimal solutions if one attempts to use these models as a basis for project crashing with Solver's nonlinear or evolutionary algorithms. Thus, the LP approach to crashing projects is still the most reliable approach to analyzing cost-time tradeoffs in a project.

Many introductory treatments of PM also address the issue of variable activity durations and show how to address the uncertainties in the critical path and project completion time using simulation (Ragsdale, 1989). The new approach to implementing project networks presented here is very amenable to simulation. If one simply replaces the expected activity durations shown in column D of Figure 3 with appropriate random number generators, standard spreadsheet simulation techniques can be applied to the model (Ragsdale, 2001).

8 CONCLUSIONS

This paper presents a new, efficient approach for implementing project networks in spreadsheets that overcomes many of the weaknesses of previous techniques. The new approach uses array formulas and circular references to compute the EST and LFT of each activity in a project. By entering and copying five simple formulas, this new technique determines the critical activities in a network regardless of the number of activities in the project. Additionally, this new approach automatically adjusts for changes in precedence relations among the activities and easily accommodates the addition or deletion of activities.

Early reports on the use of this technique in the classroom are favorable. The array formulas and (intentional) use of circular references associated with this technique are likely to be challenging, new experiences for most students. Some of the difficulty associated with students understanding the array formulas presented here may be mitigated by introducing easier array formulas earlier in the course (e.g., using an array formula SUM( ) as an alternative to SUMPRODUCT() as shown in Table 1).

9 REFERENCES

Conway, D.G., and C.T. Ragsdale, (1997), "Modeling Optimization Problems in the Unstructured World of Spreadsheets," *Omega: The International Journal of Management Science,* Vol. 25, No. 3, pp. 313-322.

Hesse, R., (1997), *Managerial Spreadsheet Modeling and Analysis,* IRWIN.

Hillier, F.S., M.S. Hillier and G.J. Lieberman, (2000), *Introduction to Management Science,* Irwin-McGraw-Hill, Boston, MA.

Moore, J.H., L.R. Weatherford, G.D. Eppen, F.J. Gould, and C.P. Schmidt, (2001), *Introductory Management Science: Decision Modeling with Spreadsheets,* Prentice-Hall, NY.

Plane, D.R., (1994), *Management Science: A Spreadsheet Approach,* Boyd and Fraser.

Ragsdale, C.T., (1989), "The Current State of Network Simulation in Project Management Theory and Practice," *Omega: The International Journal of Management Science,* Vol. 17, No. 1, pp. 21-25.

Ragsdale, C.T., (2001), *Spreadsheet Modeling and Decision Analysis: A Practical Introduction to Management Science,* South-Western College Publishing.

Seal, K.C., (2001), "A Generalized PERT/CPM Implementation in A Spreadsheet," *INFORMS Transactions On Education,* Vol. 2, No 1., pp. 16-26.