CASE STUDY

twelve

Critical Path Method

case study

OVERVIEW

CS12.1 *Application Overview and Model Development*

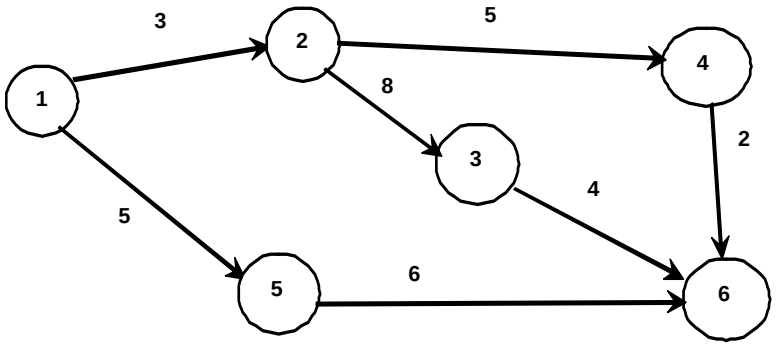
The critical path method is a quantitative technique to manage projects. This technique models a project as a network of several activities. Each activity has an associated activity time that denotes the length of the time needed to complete the activity. Activities also have precedence relationships between them; in other words, an activity can be started only when other activities are completed. The critical path method (CPM) seeks to determine which path, or set, of activities is critical to completing the project.

Related to the critical path method, the time-cost tradeoff problem entails crashing, or reducing, the activity times to a crash time at a given crash cost. Depending on the structure of the activities network, there may be several different ways in which the activities can be crashed in order to reduce the overall project time at a minimal cost. As activities are crashed and the project time is reduced, the total project cost is increased, which creates the time-cost tradeoff. The time-cost tradeoff problem is to determine for a desired overall project time which activities should be crashed in order to minimize the total crash cost, or total project cost.

In this application, the user can either find the critical path of the project or create a graph of the time-cost tradeoff from crashing activities. For the critical path method option, the user creates a project network, provides the activity times and costs, provides a precedence matrix of the events, and then views the critical path of the project with the total project completion time. For the time-cost tradeoff option, the user creates a project network, provides the activity times and costs along with the crash times and crash costs, provides a precedence matrix of the activities, and then views the time-cost tradeoff as the project is crashed for various project times.

CS12.1.1 Model Definition and Assumptions

For the CPM option, the user provides information about the activities. Here, we assume that the project network is represented by nodes as events and arcs as activities that connect those events. For example, in the network pictured below, there are six events and seven activities. The numbers written above each activity and each arc are the activity times.



To create the precedence matrix for this network representation, we create a matrix with the number of rows and number of columns equal to the number of events. Then, if any event in row *i* precedes an event in column *j*, then the activity that connects these two events is written in cell *(i, j)*. The table below is the precedence matrix corresponding to the network above.

	1	2	3	4	5	6
1		1			4	
2			3	2		
3						6
4						5
5						7
6						

The CPM then computes the early start time (the earliest time the activity can start) and the late start time (the latest time the activity can start) for each activity. Early start times are calculated by analyzing the network activities from the beginning to the end of the project. This analysis is performed using the precedence matrix; the matrix is scanned over each row i = 1 to the number of events and over each column j = 1 to the number of events. The following *If, Then* statement calculates the early start times based on the activities found in the precedence matrix. [Here *CurrentAct* is the activity listed in a cell *(i, j)*.]

```
If EarlyTime(j) < (EarlyTime(i) + ActDuration(CurrentAct)) Then
   EarlyTime(j) = (EarlyTime(i) + ActDuration(CurrentAct))
End If
```

The application calculates the late times by analyzing the network activities from the end back to the beginning of the project. This analysis is performed using the precedence matrix; the matrix is scanned over each row i = number of events backwards to 1 and over each column j = number of events backwards to 1. The following *If, Then* statement calculates the late start times based on the activities found in the precedence matrix. [Here *CurrentAct* is again the activity listed in a cell *(i, j)*.]

```
If LateTime(i) = 0 or LateTime(i) > (LateTime(j) - ActDuration(CurrentAct)) Then
   LateTime(i) = (LateTime(j) - ActDuration(CurrentAct))
End If
```

From these times, the free float and total float for each activity are computed using the precedence matrix. If there is an activity in a cell *(i, j)*, then the following calculations are performed:

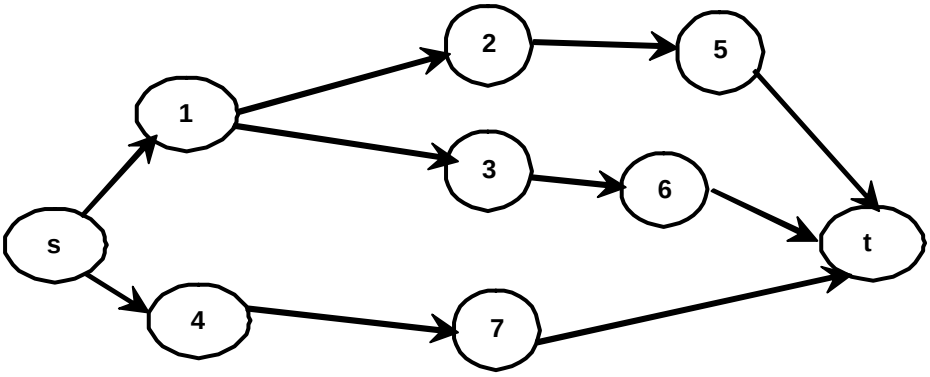
FreeFloat(CurrentAct) = EarlyTime(j) - EarlyTime(i) - ActDuration(CurrentAct)
TotalFloat(CurrentAct) = LateTime(j) - EarlyTime(i) - ActDuration(CurrentAct)

The CPM can now determine the critical path of the project. Any activity with a total float time of zero is on the critical path. The total project time and the total project cost are updated based on these critical activities.

```
If TotalFloat(i) = 0 Then
   ProjDur = ProjDur + ActDuration(i)
   ProjCost = ProjCost + NormCost(i)
End If
```

For the time-cost tradeoff option, the user provides the project network and activity information including the crash times and crash costs for each activity. Here, we assume

that the project network is represented by nodes as activities, and arcs represent the ordering of those activities. For example, in the network pictured below, there are seven activities. The nodes *s* and *t* are dummy sources and sink nodes respectively; they represent the beginning and end of the project.



To create the precedence matrix for this network representation, we develop a matrix with the number of rows and columns equal to the number of activities + 2. (The additional two activities are for the dummy *s* and *t* nodes.) Then, if any activity in row *i* precedes an activity in column *j*, the number 1 is written in cell *(i, j)*. The table below is the precedence matrix that corresponds to the network above.

	s	1	2	3	4	5	6	7	t
s		1			1				
1			1	1					
2						1			
3							1		
4								1	
5									1
6									1
7									1
t									

The application then solves a linear programming model to find the best selection of activities that should be crashed to minimize the total project cost for a desired project time. We solve this problem iteratively for various desired project times. The linear programming problem is prepared in a hidden worksheet and solved by the Solver. The parts of the model and the formulation for this problem are below. (Figure CS12.1 presents the hidden worksheet with the prepared model.)

Decision Variables:

Start time per activity, u(i)

Crash time per activity, B(i)

Constraints:

- Crash times should be less than or equal to the difference between the activity time and the crash time provided in the activity table.

  $B(i) \leq ActDur(i) - CrashDur(i)$

- The total project time must be less than or equal to the desired project time.
- Start time of end – Start time of beginning <= desired project time

  $u(t) - u(s) \leq P$

- Using the precedence matrix for every cell (i, j) with a value of 1, the start time of activity j should be greater than or equal to the start time of activity i plus the activity time for i minus the crash time for i.

  start time(j) >= start time(i) + ActDur(i) – CrashTime (i)
  $u(j) \geq u(i) + ActDur(i) - B(i)$

Objective Function:
Minimize total project cost = SUMPRODUCT(decision variables, crash slopes)
(crash slope (i) = ABS((CrashCost(i) – NormCost(i)) / (ActDur(i) - CrashDur(i))))

Model Formulation:
Minimize $\sum_i B_i S_i$, S_i = crash slopes or cost per time unit crashed
Subject to:
$B_i \leq A_i - C_i$, A_i = *activity time and* C_i = *crash time*
$u_t - u_s \leq P$
$u_j \geq u_i + A_i - B_i$
$B_i \geq 0$
$u_i \geq 0$

ObjFunc =SUMPRODUCT(DecVar, Slopes)

Crashing LP Model

End

Decision Variables | Constraints | Objective Function

Activity	Start Time per Activity	Crash Time per Activity	<=	(ai - bi)	u(t) - u(s)	<=	Project Duration	u(j)	>=	u(i) + (ai - Bi)	Min	Project Cost
s	0	0		0	7		7	0		0	0	$290.00
1	0	2		2				0		0	70	
2	4	1		1				4		4	50	
3	3	0		0				3		4	0	
4	0	0		0				5		5	0	
5	5	2		2				6		6	40	
6	6	0		0				2		2	0	
7	2	1		3				7		7	20	
t	7							7		7		
								7		7		

Figure CS12.1 The hidden optimization sheet.

The time-cost tradeoff graph shows the project cost achieved from the crashing model for each iterative project time. In this graph, the total project cost includes the sum of the normal activity costs. The resulting graph should be a piece-wise linear convex function.

For more details on the critical path method or time-cost tradeoff problem, please see *Introduction to Operations Research* by Winston.

CS12.1.2 Input

The input for the CPM option is the following:

- Project network with nodes = events and arcs = activities
- Activity times and costs
- Precedence matrix of the events

The input for the time-cost tradeoff option is the following:

- Project network with nodes = activities
- Activity times and costs along with the crash times and crash costs
- Precedence matrix of the activities

CS12.1.3 Output

The output for the CPM option is the following:

- Critical path of the project
- Total project completion time
- Event early start times and late start times
- Activity free float and total float times

The output for the time-cost tradeoff option is the following:

- Time-cost tradeoff graph for various project times and costs

CS12.2 *Worksheets*

This application requires seven worksheets: the welcome sheet, the network sheet, the activity table sheet, the precedence matrix sheet, the CPM output sheet, the time-cost tradeoff output sheet, and the hidden crash LP model sheet. (Note: we also have created an example sheet, which stores demo data and is available for the user to view if extra help is needed.) The welcome sheet contains the title, the description of the application, and the "Run Demo" and "Start" buttons. (See Figure CS12.2.) "Run Demo" and "Start" both display an option form and take the user to the network sheet.

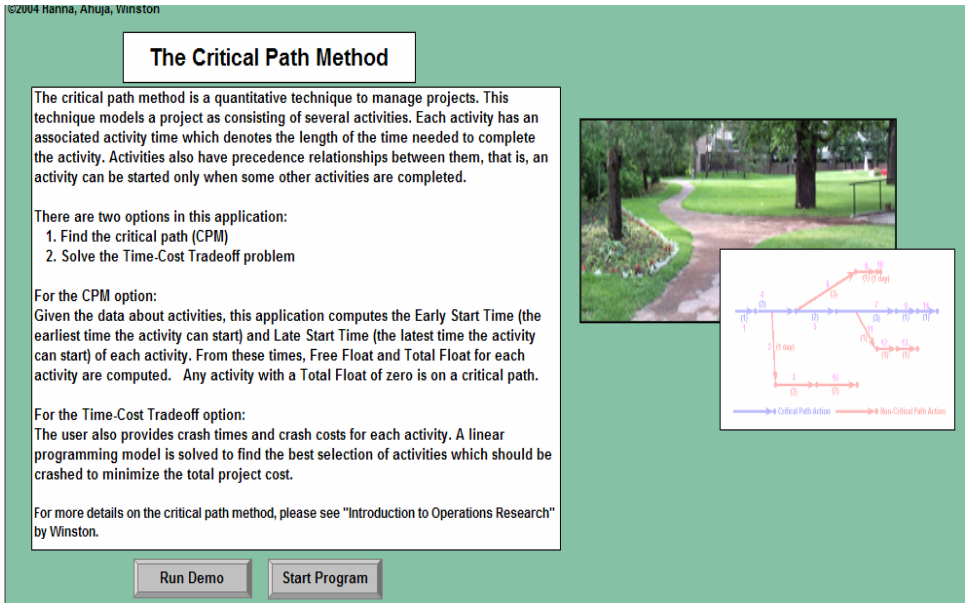


Figure CS12.2 The welcome sheet.

On the network sheet, the user creates the project network with the sample drawing objects provided (the circle with the text box, the arrow with the text box, and the dashed arrow). (See Figure CS12.3.) For the CPM option, the user creates the network such that each node = event and each arc = activity. [See Figure CS12.3 (a).] The user also needs

to name the arcs in order to view the resulting critical path. Instructions on how to name the arcs are provided in a comment box of the cell with the text “Help?” next to the buttons on the sheet. For the time-cost tradeoff option, the user creates the network such that each node = activity. [See Figure CS12.3 (b).] The buttons on the sheet are: “End,” which takes the user back to the welcome sheet; “See Example,” which takes the user to an example sheet; and “Continue,” which takes the user to the activity table sheet.

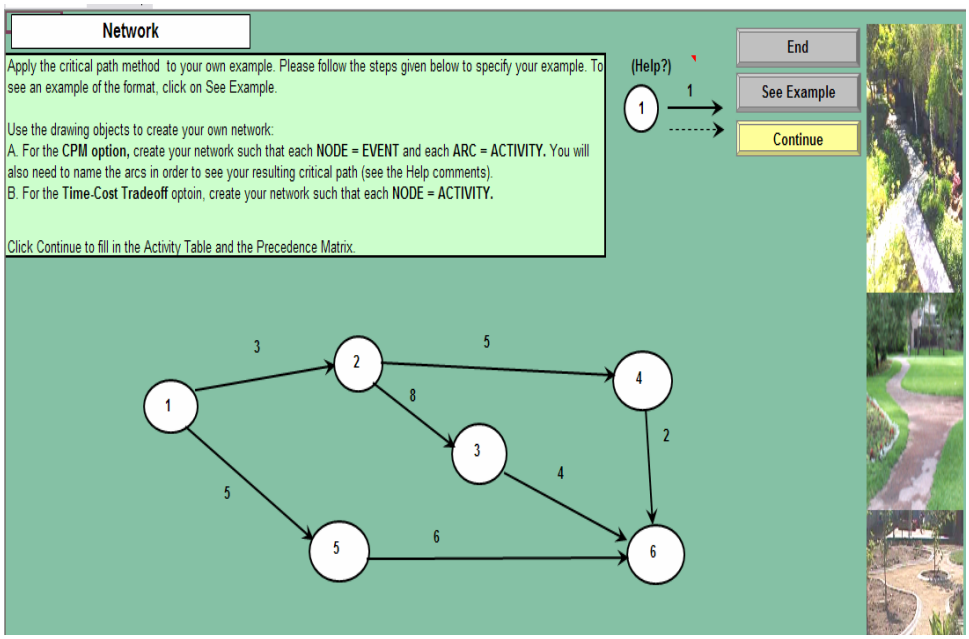


Figure CS12.3 (a)

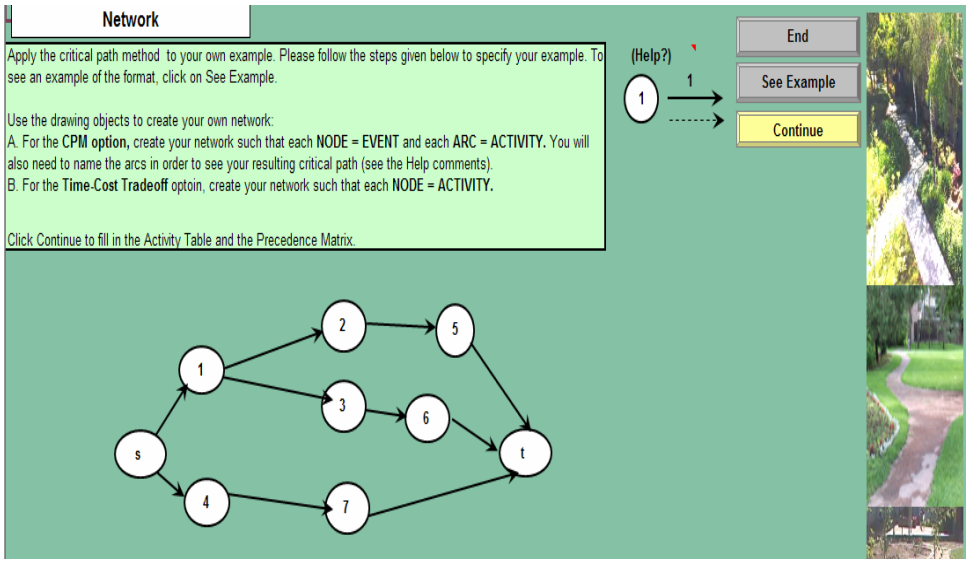
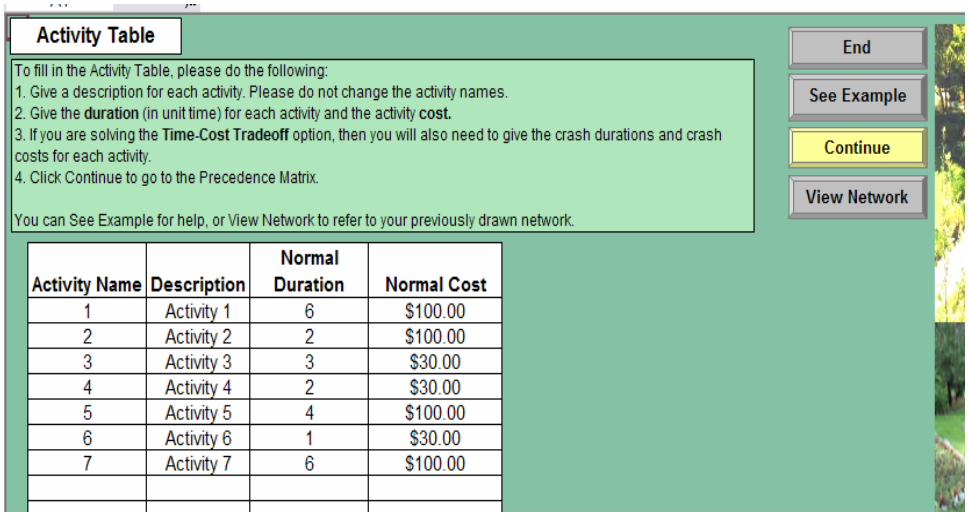


Figure CS12.3 (b)
Figure CS12.3 The network sheet.

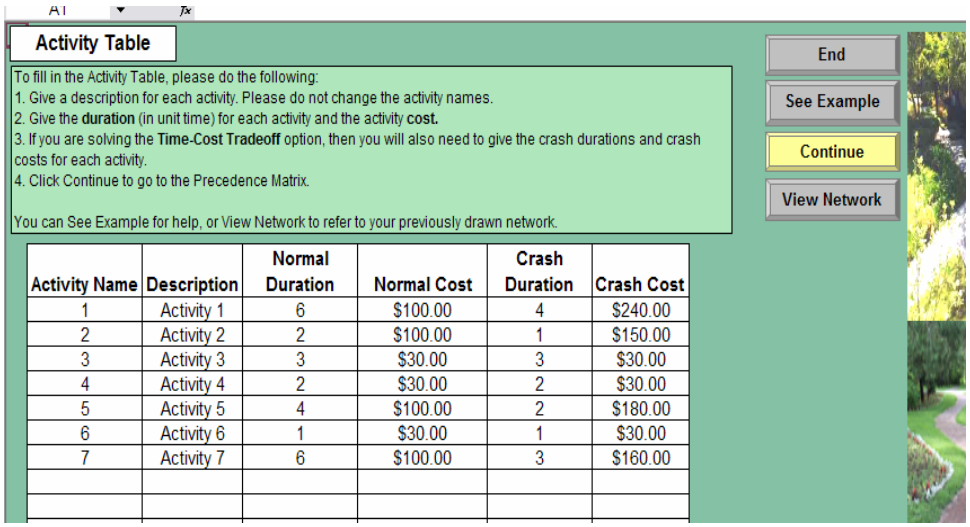
On the activity table sheet, the user provides information about each activity in the project. (See Figure CS12.4.) An activity number is automatically entered in the table; the user then describes for each activity and inputs the normal duration, or time, of the activity as well as the normal cost of the activity. [See Figure CS12.4 (a).] For the time-cost tradeoff

option, the user also provides the crash duration and crash cost for each activity. [See Figure CS12.4 (b).] The same buttons found on the network sheet are available here; this time the "Continue" button takes the user to the precedence matrix sheet. There is also a "View Network" button, which takes the user back to the network sheet; this may be useful if the user needs to check the project network to find the corresponding activities times.



Activity Name	Description	Normal Duration	Normal Cost
1	Activity 1	6	$100.00
2	Activity 2	2	$100.00
3	Activity 3	3	$30.00
4	Activity 4	2	$30.00
5	Activity 5	4	$100.00
6	Activity 6	1	$30.00
7	Activity 7	6	$100.00

Figure CS12.4 (a)

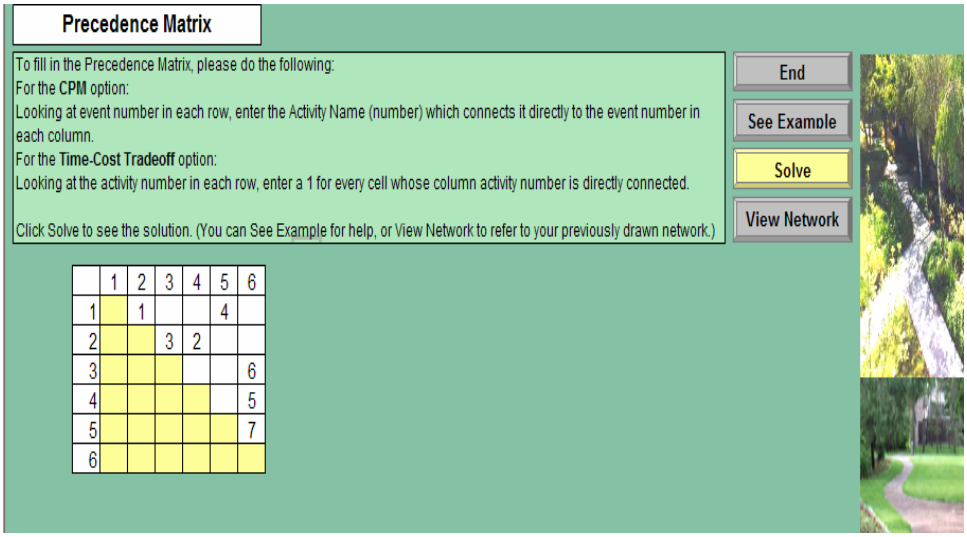


Activity Name	Description	Normal Duration	Normal Cost	Crash Duration	Crash Cost
1	Activity 1	6	$100.00	4	$240.00
2	Activity 2	2	$100.00	1	$150.00
3	Activity 3	3	$30.00	3	$30.00
4	Activity 4	2	$30.00	2	$30.00
5	Activity 5	4	$100.00	2	$180.00
6	Activity 6	1	$30.00	1	$30.00
7	Activity 7	6	$100.00	3	$160.00

Figure CS12.4 (b)
Figure CS12.4 The activity table sheet.

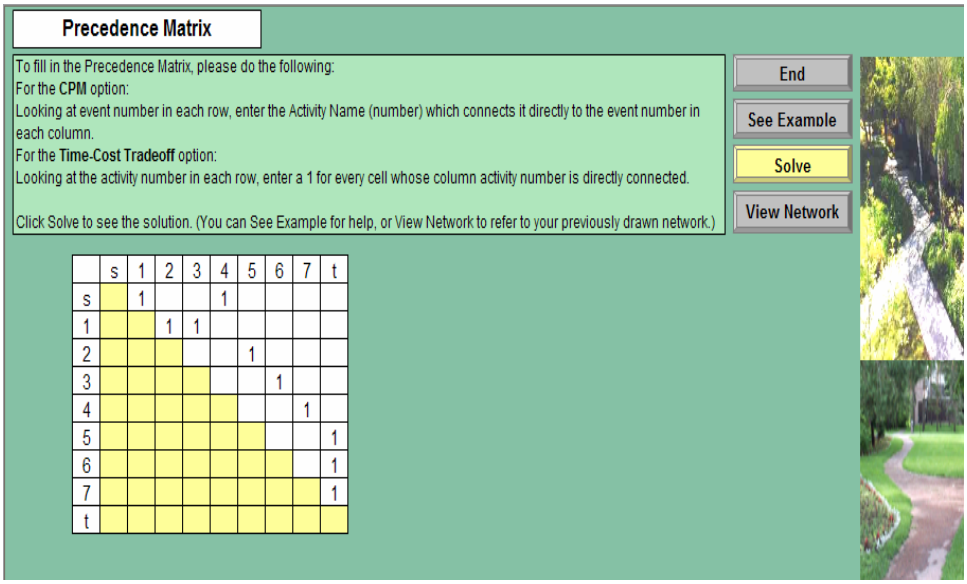
On the precedence matrix sheet, the user completes the project's precedence matrix. (See Figure 48.5.) Depending on which option the user has selected to solve, the corresponding precedence matrix format is automatically created. In other words, for the CPM option, a precedence matrix is created with the number of rows and columns = number of events. [See Figure CS12.5(a).] Here, cell values equal the activity number in the precedence relationship between the row and column events. For the time-cost tradeoff option, a precedence matrix is created with the number of rows and columns = number of activities plus two. [See Figure CS12.5 (b).] Here, cell values equal 1 if precedence exists between the row and column activities. The "End," "See Example," and

"View Network" buttons are also on this sheet. The user can also press the "Solve" button to solve the problem. He or she is then taken to the corresponding output sheet.



	1	2	3	4	5	6
1		1			4	
2			3	2		
3						6
4						5
5						7
6						

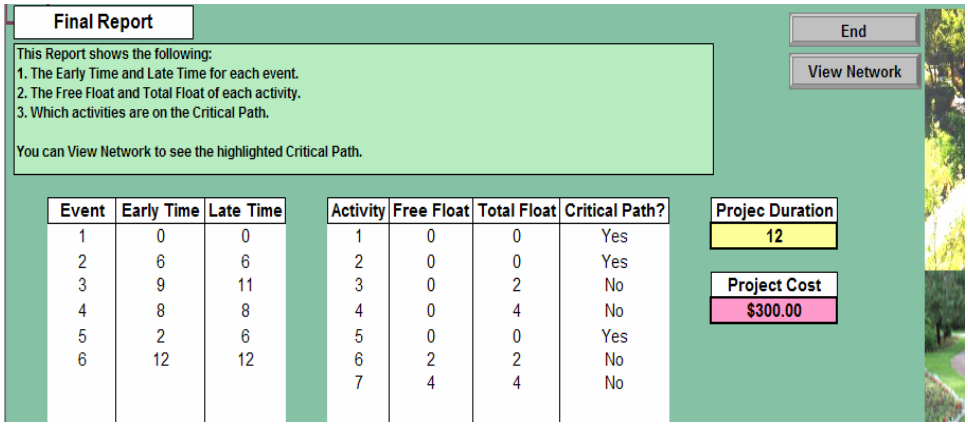
Figure CS12.5 (a)



	s	1	2	3	4	5	6	7	t
s		1			1				
1			1	1					
2						1			
3							1		
4								1	
5									1
6									1
7									1
t									

Figure CS12.5 (b)
Figure CS12.5 The precedence matrix sheet.

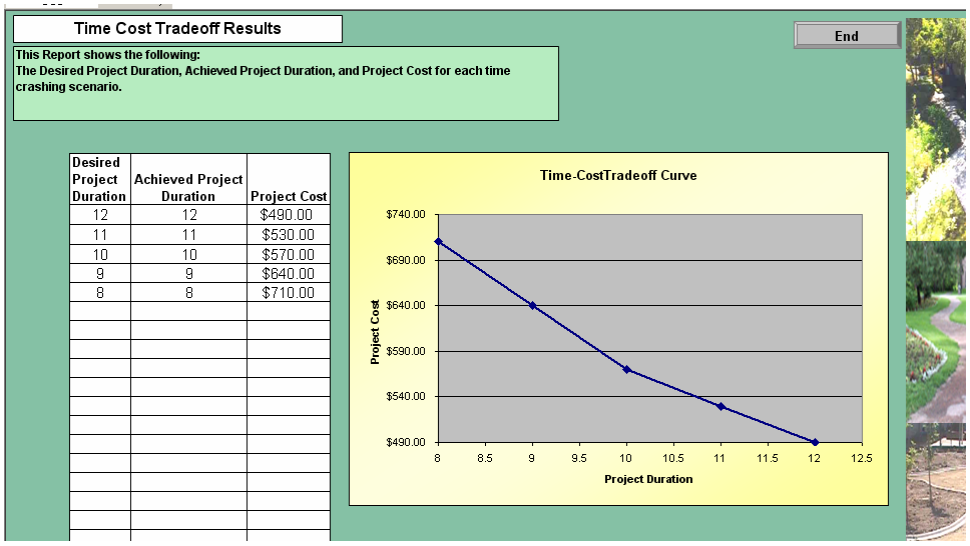
The output sheets show the solution to the selected problem. (See Figure CS12.6.) For the CPM option, the report presents the early time and late time for each event, the free float and total float time of each activity, and which activities are on the critical path. [See Figure CS12.6 (a).] The total project duration and total project cost are also provided. The user can press the "View Network" button to return to the network sheet and see the critical path highlighted on the network he or she drew. (See Figure CS12.7.) For the time-cost tradeoff option, the report presents the time-cost tradeoff graph, the corresponding desired project durations, the achieved project durations, and the project costs for each time-crashing iteration. [See Figure CS12.6 (b).]



Event	Early Time	Late Time
1	0	0
2	6	6
3	9	11
4	8	8
5	2	6
6	12	12

Activity	Free Float	Total Float	Critical Path?
1	0	0	Yes
2	0	0	Yes
3	0	2	No
4	0	4	No
5	0	0	Yes
6	2	2	No
7	4	4	No

Figure CS12.6 (a)



Desired Project Duration	Achieved Project Duration	Project Cost
12	12	$490.00
11	11	$530.00
10	10	$570.00
9	9	$640.00
8	8	$710.00

Figure CS12.6 (b)

Figure CS12.6 The output sheets.

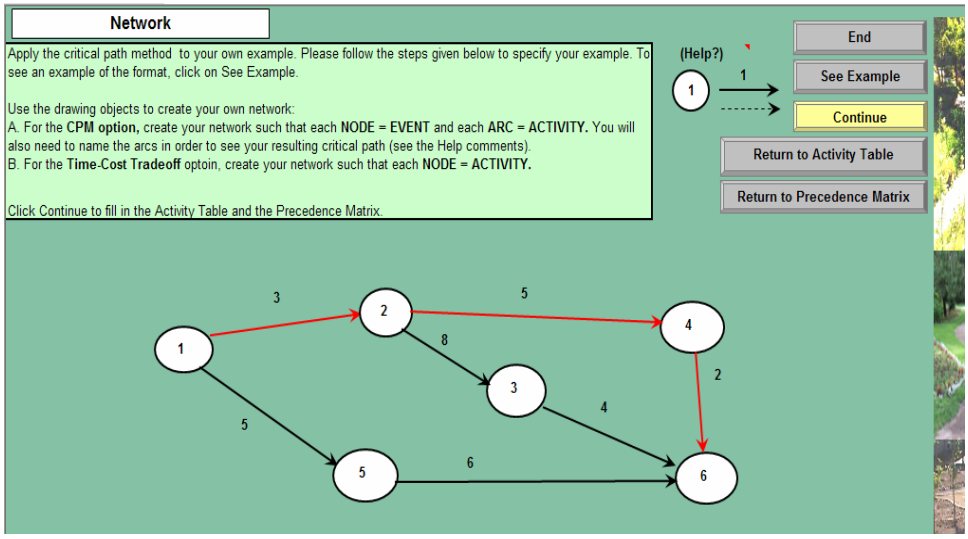


Figure CS12.7 The updated network sheet for the CPM option.

Summary

Welcome sheet	Contains the application description, the "Run Demo" button, and the "Start" button.
Network sheet	Where the user creates the project network. For the CPM option, nodes = events and arcs = activities; for the time-cost tradeoff option, nodes = activities.
Activity table sheet	Where the user provides activity information. For the CPM option, the activity times and costs; for the time-cost tradeoff option, the crash times and crash costs also.
Precedence matrix sheet	Where the user completes the precedence matrix. For the CPM option, the number of rows and columns = number of events, cell values = activity number in precedence relationship; for the time-cost tradeoff option, the number of rows and columns = number of activities plus two, cell values = 1 if precedence exits.
Output sheets	For the CPM option, the critical path is provided along with the total project time and cost and the event and activity times. For the time-cost tradeoff option, the tradeoff graph is provided with the corresponding project times and costs for each crashing iteration.
Hidden crashing LP sheet	Contains the model formulation for the crashing LP. The Solver finds the crashing times that minimize the project cost for the desired project time in each iteration.
Hidden example sheet	Contains an example of the CPM and time-cost tradeoff networks, the activity tables, and the precedence matrices.

CS12.3 *User Interface*

For this application's user interface, we use navigational and functional buttons and two user forms. When the user presses the "Run Demo" or "Start" button on the welcome sheet, the main options form appears. (See Figure CS12.8.) On this form, the user selects whether to solve the CPM problem or the time-cost tradeoff problem. This form requires one frame and two option buttons.

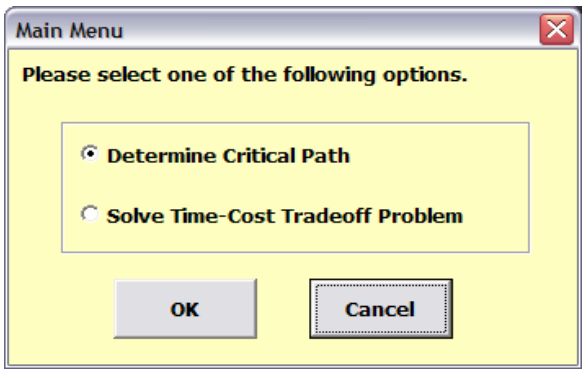


Figure CS12.8 The main menu form.

After the user creates the project network on the network sheet and presses the "Continue" button, the following form appears before the activity table sheet. (See Figure CS12.9.) This form, the network form, prompts the user for the number of events and activities represented in the project network. This form requires two text boxes.

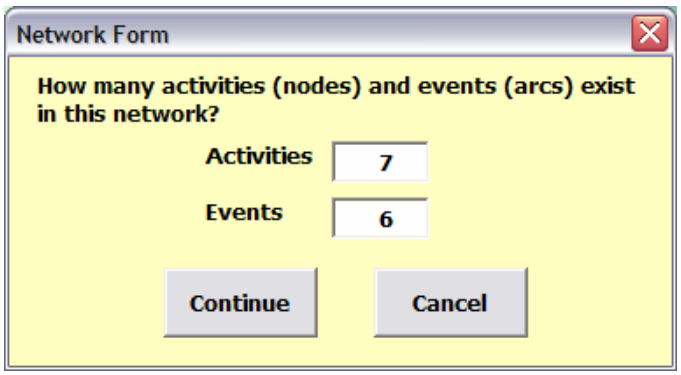


Figure CS12.9 The network input form.

Summary

Main options form	Where the user chooses between the CPM and time-cost tradeoff options.
Network form	Where the user enters the number of activities and events in the network.
Navigational buttons	"End," "View Network," "See Example," "Return to Network Sheet," "Return to Activity Table," "Return to Precedence Matrix."
Functional buttons	"Continue" and "Solve."

CS12.4 *Procedures*

We will now outline the procedures for this application beginning with the variable declarations and the initial sub procedures. (See Figure CS12.10.) The *Main* procedure, which is called from the "Start" button, calls the *ClearPrevious* procedure to clear previous values from all the sheets. It then displays the main options form and takes the user to the network sheet.

```
Option Explicit
Option Base 1

Public NEvents As Integer, NActivities As Integer, i As Integer, j As Integer, _
Obj As Object, PStartCell As Range, AStartCell As Range, RStartCell As Range, _
Graph As Range, Default As Boolean, ws As Worksheet, ProjDurUser As Double, _
SolveCPM As Boolean, Iter As Integer, TradeDone As Boolean, _
NumCrashed As Integer, PDStop As Boolean, PreCount As Integer

Public FreeFloat() As Double, TotalFloat() As Double, _
EarlyTime() As Double, LateTime() As Double, ActDuration() As Double, CurrentAct As Integer, _
CrashDuration() As Double, NormCost() As Double, CrashCost() As Double, ProjDur As Double, _
CrashSlope() As Double, MinSlope As Double, ProjCost As Double, _
MinSlopeIndex As Integer, ReductionTime() As Double, SumNormCost As Double, SolverResult As Integer

Sub Main()     'This sub is called when the user clicks the Start button
    Call ClearPrevious
    Default = False
    frmMain.Show

    Worksheets("Network Sheet").Visible = True
    Worksheets("Welcome").Visible = False
    Range("A1").Select
End Sub
```

Figure CS12.10 The variable declarations and the *Main* procedure.

The *Demo* procedure is called from the “Run Demo” button on the welcome sheet. (See Figure CS12.11.) This procedure also begins by calling the *ClearPrevious* procedure and showing the main optoins form. Then, depending on which problem the user has selected to solve, the corresponding example data is copied from the example sheet to the network sheet, the activity table sheet, and the precedence matrix sheet. The user is then taken to the network sheet. A message box informs the user that the project network and other values have already been created for the demo.

```
Sub Demo()      'This sub runs the demo using data from example sheet
    Call ClearPrevious         'in case demo is run multiple times
    Default = True
    frmMain.Show

    'copy example data to all sheets
    Application.ScreenUpdating = False
    If SolveCPM Then
        Worksheets("Example").Range("ExNetworkA").Copy
        ActiveSheet.Paste Destination:=Worksheets("Network Sheet").Range("B15")

        Range(Range("CrashStart"), Range("CrashStart").Offset(20, 1)).Clear
        Worksheets("Example").Range(Range("ExActTable").Columns(1), Range("ExActTable").Columns(4)).Copy
        ActiveSheet.Paste Destination:=AStartCell

        Worksheets("Example").Range("ExPrecMatA").Copy
        ActiveSheet.Paste Destination:=PStartCell
    Else
        Worksheets("Example").Range("ExNetworkB").Copy
        ActiveSheet.Paste Destination:=Worksheets("Network Sheet").Range("B15")

        Range("CopyCol").Copy
        Range(Range("CrashStart"), Range("CrashStart").Offset(20, 1)).PasteSpecial xlPasteFormats
        Worksheets("Example").Range("ExActTable").Copy
        ActiveSheet.Paste Destination:=AStartCell

        Worksheets("Example").Range("ExPrecMatB").Copy
        ActiveSheet.Paste Destination:=PStartCell
    End If
    Application.CutCopyMode = False

    Worksheets("Network Sheet").Visible = True
    Worksheets("Welcome").Visible = False
    Application.ScreenUpdating = True
    MsgBox "The demo network has already been created. Demo values will be filled on all following sheets as well." & _
        vbCrLf & "Please press the Continue button on the next to sheets, followed by the Solve button."
End Sub
```

Figure CS12.11 The *Demo* procedure.

The procedures for the main options form are illustrated in Figure CS12.12. A Boolean value records the user's selection of either the CPM problem or the time-cost tradeoff problem.

```
Private Sub cmdCancel_Click()
    Worksheets("Welcome").Visible = True
    Unload Me
    End
End Sub

Private Sub cmdOK_Click()
    Dim UserInput As Variant
    If optCPM Then
        'run standard CPM program
        SolveCPM = True
    ElseIf optTradeOff Then
        'solve tradeoff with crashing
        SolveCPM = False
    End If

    Unload Me
End Sub
```

Figure CS12.12 The main menu form procedures.

When the user presses the "Continue" button on the network sheet, the *CreateTables* procedure is called. (See Figure CS12.13.) If the demo is not being run, then the network form asks the user for the number of events and activities in the network. Using these parameters, the activity table is created with the number of rows equal to the number of activities. The precedence matrix is also created. For the CPM option, the precedence matrix's number of rows and number of columns equal the number of events; for the time-cost tradeoff option, the precedence matrix's number of rows and number of columns equal the number of activities plus two. The application also formats the precedence matrix to create an upper-triangle matrix. In other words, the user should only enter values in the upper right-hand part of the matrix. If the demo is being performed, neither the activity table nor the precedence matrix is created since each has already been copied from the example sheet. At the end of this procedure, the user views the activity table sheet.

The procedures for the network input form are presented in Figure CS12.14. These procedures simply record the number of activities and number of events. The procedure also performs some error checking to ensure that there are at least two activities in the network.

When the user presses the "Continue" button on the activity table sheet, the *ActivityList* procedure is called. (See Figure CS12.15.) In this procedure, arrays store the activity times and costs that the user has entered into the table. If the user has selected the time-cost tradeoff option, then the crash times and costs are also recorded and the crash slopes are calculated. The user then views the precedence matrix sheet.

The "Solve" button on the precedence matrix sheet is assigned to the *Solve* procedure. (See Figure CS12.16.) If the user is solving the CPM problem, then the *CPM* procedure is called to determine the critical path. Then, the *CreateReport* procedure is called to display the CPM results on the output sheet.

```
Sub CreateTables()     'This sub is called by pressing Continue on Network Sheet
    If Default = False Then
        frmNetwork.Show    'number of events and activities from user
    Else
        NActivities = 7
        NEvents = 6
    End If

    If Default = False Then     'tables do not need to be created when running demo since example data is copied
      Application.ScreenUpdating = False
      Worksheets("Activity Sheet").Activate     'creates Activity Table
      For i = 1 To NActivities
        AStartCell.Offset(i, 0).Value = i
      Next i
      If SolveCPM Then  'only show crash columns if solving crash problem
        Range(Range("CrashStart"), Range("CrashStart").Offset(20, 1)).Clear
      Else
        Range("CrashStart").Value = "Crash Duration"
        Range("CrashStart").Offset(0, 1).Value = "Crash Cost"
        Range("CopyCol").Copy
        Range(Range("CrashStart"), Range("CrashStart").Offset(20, 1)).PasteSpecial xlPasteFormats
      End If

      Worksheets("Precedence Matrix Sheet").Activate    'creates Precedence Matrix
        If SolveCPM Then
          Range(PStartCell, PStartCell.Offset(NEvents, NEvents)).Interior.ColorIndex = 2
          Range(PStartCell, PStartCell.Offset(NEvents, NEvents)).Borders(xlInsideHorizontal).Weight = xlThin
          Range(PStartCell, PStartCell.Offset(NEvents, NEvents)).Borders(xlInsideVertical).Weight = xlThin
            For i = 1 To NEvents
              PStartCell.Offset(i, 0).Value = i
              PStartCell.Offset(0, i).Value = i
              Range(PStartCell.Offset(i, 1), PStartCell.Offset(i, i)).Interior.ColorIndex = 36 'upper-triangular matrix
            Next i
        Else
          Range(PStartCell, PStartCell.Offset(NActivities + 2, NActivities + 2)).Interior.ColorIndex = 2
          Range(PStartCell, PStartCell.Offset(NActivities + 2, NActivities + 2)).Borders(xlInsideHorizontal).Weight = xlThin
          Range(PStartCell, PStartCell.Offset(NActivities + 2, NActivities + 2)).Borders(xlInsideVertical).Weight = xlThin
            For i = 1 To NActivities
              PStartCell.Offset(i + 1, 0).Value = i
              PStartCell.Offset(0, i + 1).Value = i
              Range(PStartCell.Offset(i, 1), PStartCell.Offset(i, i)).Interior.ColorIndex = 36  'upper-triangular matrix
            Next i
          PStartCell.Offset(1, 0).Value = "s"
          PStartCell.Offset(0, 1).Value = "s"
          PStartCell.Offset(NActivities + 2, 0).Value = "t"
          PStartCell.Offset(0, NActivities + 2).Value = "t"
          'upper-triangular matrix
          Range(PStartCell.Offset(NActivities + 1, 1), PStartCell.Offset(NActivities + 1, NActivities + 1)).Interior.ColorIndex = 3
          Range(PStartCell.Offset(NActivities + 2, 1), PStartCell.Offset(NActivities + 2, NActivities + 2)).Interior.ColorIndex = 3
        End If
    End If

    Worksheets("Network Sheet").Activate
    ActiveSheet.Shapes("RetAct").Visible = True     'makes Return to Activity Table button visible
    Worksheets("Activity Sheet").Visible = True
    Worksheets("Network Sheet").Visible = False
    Range("A1").Select
    Application.ScreenUpdating = True
End Sub
```

Figure CS12.13 The *CreateTable* procedure.

```
Sub cmdCancel_Click()
    Worksheets("Welcome").Visible = True
    Worksheets("Network Sheet").Visible = False
    Unload Me
    End
End Sub
Sub cmdContinue_Click()
    If txtActivities.Value < 2 Or IsNumeric(txtActivities.Value) = False Then
        MsgBox "You must have at least 2 activities in your network."
        Exit Sub
    End If

    NActivities = txtActivities
    NEvents = txtEvents

    Unload Me
End Sub
```

Figure CS12.14 The network input form procedures.

```
Sub ActivityList()     'This sub is called by pressing Continue on Activity List Sheet
    Worksheets("Activity Sheet").Activate
    ReDim ActDuration(NActivities), CrashDuration(NActivities), NormCost(NActivities), CrashCost(NActivities), _
          CrashSlope(NActivities), ReductionTime(NActivities), _
          FreeFloat(NActivities), TotalFloat(NActivities), EarlyTime(NEvents), LateTime(NEvents)

    SumNormCost = 0
    For i = 1 To NActivities
        ActDuration(i) = AStartCell.Offset(i, 2).Value      'set ActDuration array to duration values from table
        NormCost(i) = AStartCell.Offset(i, 3).Value
        SumNormCost = SumNormCost + NormCost(i)
        If SolveCPM = False Then
            CrashDuration(i) = AStartCell.Offset(i, 4).Value
            CrashCost(i) = AStartCell.Offset(i, 5).Value
            If (ActDuration(i) - CrashDuration(i)) = 0 Then
                CrashSlope(i) = 0
            Else
                CrashSlope(i) = Abs((CrashCost(i) - NormCost(i)) / (ActDuration(i) - CrashDuration(i)))
            End If
        End If
    Next i

    Worksheets("Network Sheet").Activate
    ActiveSheet.Shapes("RetPrec").Visible = True        'makes Return to Precedence Matrix button visible

    Worksheets("Precedence Matrix Sheet").Visible = True
    Worksheets("Activity Sheet").Visible = False
    Range("A1").Select
End Sub
```

Figure CS12.15 The *ActivityList* procedure.

If the time-cost tradeoff problem is solved, then the application calls the *CrashLP* procedure to prepare the LP model on the hidden crash LP sheet. Then, a “Do, While” loop solves this model for different project times. The first time the model is solved, the desired project time is set very high to ensure feasibility. The resulting achieved project time then acts the first project time value in the performed iterations; each iteration decreases the desired project time by one time unit. For every loop, the desired project time and the resulting project cost are recorded on the output sheet. When the loop is finished (when the model has no feasible solution for the desired project time), the time-cost tradeoff graph is created from these recorded values.

```
Sub Solve() 'called from Solve button
    Application.ScreenUpdating = False

    If SolveCPM Then
        Call CPM
        Call CreateReport

    Else
        Call CrashLP    'prepare solver model
        Iter = 1
        ProjDurUser = 100000
        Range("ProjDur").Value = ProjDurUser
        Do
            Range("DecVar").ClearContents   'clear dec var or initialize to normal duration times?
            Range("StartTimes").ClearContents 'clear other dec var
            SolverResult = SolverSolve(UserFinish:=True)     'run Solver
            If Iter = 1 Then
                ProjDurUser = Range("Con2").Value
                Worksheets("TradeOff").ChartObjects(1).Activate
                ActiveChart.Axes(xlValue).MinimumScale = SumNormCost
                Worksheets("Optimize").Activate
            End If

            If SolverResult = 5 Then    'infeasible
                Exit Do
            Else
                'record achieved and desired proj duration and proj cost
                Worksheets("TradeOff").Range("TradeStart").Offset(Iter, 0).Value = ProjDurUser
                Worksheets("TradeOff").Range("TradeStart").Offset(Iter, 1).Value = (Range("ProjDur").Offset(0, -2).Value)
                Worksheets("TradeOff").Range("TradeStart").Offset(Iter, 2).Value = Range("ObjFunc").Value + SumNormCost

                Iter = Iter + 1
                ProjDurUser = ProjDurUser - 1   'update trial proj duration
                Range("ProjDur").Value = ProjDurUser    'record to optimization sheet
            End If
        Loop While ProjDurUser > 0

        Application.Union(Range(Range("TradeStart").Offset(1, 0), Range("TradeStart").Offset(1, 0).End(xlDown)), _
                    Range(Range("TradeStart").Offset(1, 2), Range("TradeStart").Offset(1, 2).End(xlDown))).Name = "TradeGraph"
        Worksheets("TradeOff").ChartObjects("TradeChart").Activate
        ActiveChart.SetSourceData Source:=Range("TradeGraph"), PlotBy:=xlColumns
        ActiveChart.Axes(xlCategory).MinimumScale = ProjDurUser + 1
        Worksheets("Optimize").Visible = False
        Worksheets("TradeOff").Visible = True
        Range("A1").Select
    End If

    Worksheets("Precedence Matrix Sheet").Visible = False
    Application.ScreenUpdating = True
End Sub
```

Figure CS12.16 The *Solve* procedure.

For the CPM option, the *CPM* procedure finds the critical path. (See Figure CS12.17.) In this procedure, the early times and late times are calculated for each event by scanning the precedence matrix, as described in Section CS22.1.1. Then, the application calculates the free float and total float times for each activity, again by using the precedence matrix. The total float times are then reviewed to determine which activities are on the critical path (those with a total float time of zero). The total project duration and project cost are also calculated.

```
Sub CPM()     'Find the critical path
    Worksheets("Precedence Matrix Sheet").Activate
    For i = 1 To NEvents        'initializing arrays
        EarlyTime(i) = 0
        LateTime(i) = 0
    Next i

    For i = 1 To NEvents        'find Early Time of each event
        For j = 1 To NEvents
            If IsEmpty(PStartCell.Offset(i, j)) = False Then
                CurrentAct = PStartCell.Offset(i, j).Value
                If EarlyTime(j) < (EarlyTime(i) + ActDuration(CurrentAct)) Then
                    EarlyTime(j) = (EarlyTime(i) + ActDuration(CurrentAct))
                End If
            End If
        Next j
    Next i
    LateTime(NEvents) = EarlyTime(NEvents)
    i = NEvents

     For i = NEvents To 1 Step -1       'find Late Time of each event
        For j = NEvents To 1 Step -1
            If IsEmpty(PStartCell.Offset(i, j)) = False Then
                CurrentAct = PStartCell.Offset(i, j).Value
                If LateTime(i) = 0 Or LateTime(i) > (LateTime(j) - ActDuration(CurrentAct)) Then
                    LateTime(i) = (LateTime(j) - ActDuration(CurrentAct))
                End If
            End If
        Next j
     Next i

    For i = 1 To NEvents        'find Free Float and Total Float of each activity
     For j = 1 To NEvents
        If IsEmpty(PStartCell.Offset(i, j)) = False Then
            CurrentAct = PStartCell.Offset(i, j).Value
            FreeFloat(CurrentAct) = EarlyTime(j) - EarlyTime(i) - ActDuration(CurrentAct)
            TotalFloat(CurrentAct) = LateTime(j) - EarlyTime(i) - ActDuration(CurrentAct)
        End If
     Next j
    Next i

    ProjDur = 0
    ProjCost = 0
    For i = 1 To NActivities
        If TotalFloat(i) = 0 Then       'activity on critical path - use to calc project duration and cost
            ProjDur = ProjDur + ActDuration(i)
            ProjCost = ProjCost + NormCost(i)
        End If
    Next i
End Sub
```

Figure CS12.17 The *CPM* procedure.

The *CreateReport* procedure outputs the values found in the *CPM* procedure to the output sheet. (See Figure CS12.18.) The early and late times for each event are recorded in an events table, and the free float and total float times for each activity are recorded in an activities table. For each activity on the critical path, the corresponding arc object on the project network (which the user created on the network sheet) is colored red. The total project duration and project cost are also recorded. Then, the user views the output sheet.

```
Sub CreateReport()  'create output sheet for CPM option
    Application.ScreenUpdating = False

    For i = 1 To NEvents                        'record early and late times
        RStartCell.Offset(i, 0).Value = i
        RStartCell.Offset(i, 1).Value = EarlyTime(i)
        RStartCell.Offset(i, 2).Value = LateTime(i)
    Next i

    For i = 1 To NActivities                'record free float and total float
        RStartCell.Offset(i, 4).Value = i
        RStartCell.Offset(i, 5).Value = FreeFloat(i)
        RStartCell.Offset(i, 6).Value = TotalFloat(i)

        If TotalFloat(i) = 0 Then           'record whether or not activity is on critical path
            RStartCell.Offset(i, 7).Value = "Yes"
            Worksheets("Network Sheet").Activate
            With Worksheets("Network Sheet")
                For Each Obj In ActiveSheet.DrawingObjects      'color critical path activities red
                    If Obj.Name = "Activity" & i Then
                     Obj.ShapeRange.Line.ForeColor.SchemeColor = 10
                    End If
                Next Obj
            End With
        Else
            Worksheets("Report Sheet").Activate
            RStartCell.Offset(i, 7).Value = "No"
        End If
    Next i

    Range("ProjDurCost").Offset(1, 0).Value = ProjDur
    Range("ProjDurCost").Offset(4, 0).Value = ProjCost

    Worksheets("Report Sheet").Activate
    Worksheets("Report Sheet").Visible = True
    Range("A1").Select
    Application.ScreenUpdating = True
End Sub
```

Figure CS12.18 The *CreateReport* procedure.

The *CrashLP* procedure loads the crashing LP model on the hidden sheet. (See Figure CS12.19.) The values stored from the activity table and the precedence matrix create the constraints. The decision variable cells are prepared for both the start times and the crash times. The crash slopes are listed for the objective function. After all the required ranges have been named, the Solver model is loaded. The Solver is run in the *Solve* procedure's loop.

Figure CS12.20 illustrates the *ClearPrevious* procedure. This procedure clears the previous values on all the sheets. On the network sheet, the drawing objects that create the project network are deleted. The values in the activity table are cleared, and the entire precedence matrix is removed. Finally, the application clears the values on both of the output sheets.

```
Sub CrashLP()   'use Solver to perform crashing
    Worksheets("Optimize").Visible = True
    Worksheets("Optimize").Activate
    'prepare formula using values from activity table and precedence matrix
    For i = 1 To NActivities
        Range("ActStart").Offset(i + 1, 0).Value = i
        Range("Con1").Offset(i + 1, 0).Value = ActDuration(i) - CrashDuration(i)
        Range("DiStart").Offset(i + 1, 0).Value = CrashSlope(i)
    Next i
    Range("ActStart").Offset(NActivities + 2, 0).Value = "t"
    Range("DiStart").Offset(1, 0).Value = 0
    Range("Con1").Offset(1, 0).Value = 0
    Range("Con2").Formula = "=" & Range("StartTimes").Cells(NActivities + 2).Address
    'Range("ObjFunc").Formula = "=SUMPRODUCT(DecVar, Slopes)" is done in spreadsheet
    'Range("ProjDur").Value = ProjDurUser is done in loop in Solve procedure
    PreCount = 0
    For i = 1 To NActivities + 2
        For j = 1 To NActivities + 2
            If PStartCell.Offset(i, j).Value = 1 Then   'check precedence matrix
                PreCount = PreCount + 1
                Range("Con3").Offset(PreCount, 0).Formula = "=" & Range("StartTimes").Cells(j).Address

                If i = 1 Then
                    Range("Con3").Offset(PreCount, 2).Formula = "=" & Range("StartTimes").Cells(i).Address & "+" & _
                    0 & "-" & Range("DecVar").Cells(i).Address
                Else
                    Range("Con3").Offset(PreCount, 2).Formula = "=" & Range("StartTimes").Cells(i).Address & "+" & _
                    ActDuration(i - 1) & "-" & Range("DecVar").Cells(i).Address
                End If
            End If
        Next j
    Next i

    'name all ranges
    Range(Range("DVStart").Offset(1, 0), Range("DVStart").Offset(NActivities + 1, 0)).Name = "DecVar"
    Range(Range("ActStart").Offset(1, 1), Range("ActStart").Offset(NActivities + 2, 1)).Name = "StartTimes"
    Range(Range("Con1").Offset(1, 0), Range("Con1").Offset(NActivities + 1, 0)).Name = "Constraint1"
    Range(Range("DiStart").Offset(1, 0), Range("DiStart").Offset(NActivities + 1, 0)).Name = "Slopes"
    Range(Range("Con3").Offset(1, 0), Range("Con3").Offset(NActivities + 3, 0)).Name = "Constraint3"
    Range(Range("Con3").Offset(1, 2), Range("Con3").Offset(NActivities + 3, 2)).Name = "Con3RHS"

    'prepare Solver
    SolverReset
    SolverOK SetCell:="ObjFunc", MaxMinVal:=2, ByChange:=Range("DecVar, StartTimes")
    SolverAdd CellRef:="DecVar", Relation:=1, FormulaText:="Constraint1"
    SolverAdd CellRef:="Con2", Relation:=1, FormulaText:="ProjDur"
    SolverAdd CellRef:="Constraint3", Relation:=3, FormulaText:="Con3RHS"
    SolverOptions AssumeLinear:=True, AssumeNonNeg:=True
    'SolverSolve UserFinish:=True is done in loop in Solve procedure
End Sub
```

Figure CS12.19 The *CrashLP* procedure.

```
Sub ClearPrevious()      'clears previous data
    Application.ScreenUpdating = False
    SolveCPM = False
    TradeDone = False
    MinSlope = 999999
    NumCrashed = 0
    Iter = 1
    PDStop = False

    Worksheets("Activity Sheet").Activate
    Set AStartCell = Range("B9")
    Range(AStartCell.Offset(1, 0), "H50").ClearContents

    Worksheets("Precedence Matrix Sheet").Activate
    Set PStartCell = Range("B9")
    With Range(PStartCell.Offset(0, 0), "AA100")
        .ClearContents
        .Interior.ColorIndex = 0
        .Borders(xlInsideHorizontal).LineStyle = xlNone
        .Borders(xlInsideVertical).LineStyle = xlNone
    End With

    Worksheets("Report Sheet").Activate
    Set RStartCell = Range("B9")
    Range(RStartCell.Offset(1, 0), "I100").ClearContents
    Range("ProjDurCost").Offset(1, 0).ClearContents
    Range("ProjDurCost").Offset(4, 0).ClearContents

    Worksheets("TradeOff").Activate
    With Range(Range("TradeStart").Offset(1, 0), Range("TradeStart").Offset(0, 2).End(xlDown))
        .ClearContents
        .Font.Strikethrough = False
        .Font.ColorIndex = 1
    End With

    Worksheets("Network Sheet").Activate
    Set Graph = Range("A13:I35")
    Graph.Cut
    ActiveSheet.Paste Destination:=Worksheets("Deleted").Range("B3")
    Worksheets("Deleted").Cells.Delete

    ActiveSheet.Shapes("RetAct").Visible = False
    ActiveSheet.Shapes("RetPrec").Visible = False
    ActiveSheet.Shapes("RetReport").Visible = False

    Worksheets("Network Sheet").Range("M2").ClearContents
    Worksheets("Activity Sheet").Range("H2").ClearContents
    Worksheets("Precedence Matrix Sheet").Range("L2").ClearContents
    Application.ScreenUpdating = True
End Sub
```

Figure CS12.20 The *ClearPrevious* procedure.

Figure CS12.21 presents the navigational procedures.

```
Sub Cancel()
    Worksheets("Welcome").Visible = True
    ActiveSheet.Visible = False
End Sub

Sub SeeEx()
    Worksheets("Example").Visible = True
    ActiveSheet.Visible = False
End Sub

Sub ViewNet()
    Worksheets("Network Sheet").Visible = True
    ActiveSheet.Visible = False
End Sub

Sub ReturnActivity()
    Worksheets("Activity Sheet").Visible = True
    ActiveSheet.Visible = False
End Sub

Sub ReturnPrecedence()
    Worksheets("Precedence Matrix Sheet").Visible = True
    ActiveSheet.Visible = False
End Sub

Sub ReturnReport()
    If SolveCPM Then
        Worksheets("Report Sheet").Visible = True
    Else
        Worksheets("TradeOff").Visible = True
    End If
    ActiveSheet.Visible = False
End Sub
```

Figure CS12.21 The navigational procedures.


Summary

Main	Initializes the application and shows the user the main option form.
Demo	Initializes the application, shows the user the main option form, and copies the demo values from the example sheet.
ClearPrevious	Clears the previous values from all the sheets.
Main option form procedures	Record the user's option to solve the CPM or time-cost tradeoff problem.
CreateTables	Shows the network input form and creates the activity table and precedence matrix based on those parameters.
Network input form procedures	Record the number of events and activities in the user's project network.
ActivityList	Records the values from the activity table.
Solve	Solves the CPM or time-cost tradeoff problem.
CPM	Finds the critical path by calculating the event and activity times.
CreateReport	Displays the CPM results to the output sheet.
CrashLP	Prepares the crashing LP model on the hidden sheet.
Navigational procedures	Apply to all the navigational buttons.

CS12.5 *Re-solve Options*

The user can re-solve this application by returning to the welcome sheet and solving the model again. The user can choose to solve a different model or the same model with a different project network or different activity times and costs.

CS12.6 *Summary*

- In this application, the user can either find the critical path of the project or create a graph of the time-cost tradeoff from crashing activities.
- This application requires seven worksheets: the welcome sheet, the network sheet, the activity table sheet, the precedence matrix sheet, the CPM output sheet, the time-cost tradeoff output sheet, and the hidden crash LP model sheet. (There is also a hidden example sheet.)
- For this application's user interface, we use navigational and functional buttons and two user forms.
- Several procedures in this application record the information about the user's project network and solve either the CPM or time-cost tradeoff problem.
- The user can re-solve this application by returning to the welcome sheet and solving the model again.

CS12.7 *Extensions*

- Make the network form dynamic so that if the time-cost tradeoff problem is being solved, the form only prompts the user for the number of activities. In other

words, the label and text box for the number of events should disappear or be grayed-out if the time-cost tradeoff option was selected in the first options form.

- Create different re-solve options:
  - Allow the user to preserve the project network but change the values on the activity table.
  - Allow the user to keep the same network but solve a different problem. For example, he or she may first solve the CPM problem and then want to solve the time-cost tradeoff problem. What new values does the user need to provide?
  - Can the user take the results of the time-cost tradeoff problem and find the critical path for a particular set of activity times? How would this be done?
- Extend this application to include a third option to solve the PERT problem. What features will be the same? What features will need to be added?