CASE STUDY

twelve

Critical Path Method

OVERVIEW

CS12.1 Application Overview and Model Development
CS12.2 Worksheets
CS12.3 User Interface
CS12.4 Procedures
CS12.5 Re-solve Options
CS12.6 Summary
CS12.7 Extensions
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.

### 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.
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 \( \text{If, Then} \) statement calculates the early start times based on the activities found in the precedence matrix. [Here \( \text{CurrentAct} \) is the activity listed in a cell \((i, j)\).]

\[
\text{If EarlyTime}(j) < (\text{EarlyTime}(i) + \text{ActDuration}(\text{CurrentAct})) \text{ Then} \\
\quad \text{EarlyTime}(j) = (\text{EarlyTime}(i) + \text{ActDuration}(\text{CurrentAct})) \\
\text{End If}
\]

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 \( \text{If, Then} \) statement calculates the late start times based on the activities found in the precedence matrix. [Here \( \text{CurrentAct} \) is again the activity listed in a cell \((i, j)\).]

\[
\text{If LateTime}(i) = 0 \text{ or } \text{LateTime}(i) > (\text{LateTime}(j) - \text{ActDuration}(\text{CurrentAct})) \text{ Then} \\
\quad \text{LateTime}(i) = (\text{LateTime}(j) - \text{ActDuration}(\text{CurrentAct})) \\
\text{End If}
\]

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:

\[
\text{FreeFloat}(\text{CurrentAct}) = \text{EarlyTime}(j) - \text{EarlyTime}(i) - \text{ActDuration}(\text{CurrentAct}) \\
\text{TotalFloat}(\text{CurrentAct}) = \text{LateTime}(j) - \text{EarlyTime}(i) - \text{ActDuration}(\text{CurrentAct})
\]

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.

\[
\text{If TotalFloat}(i) = 0 \text{ Then} \\
\quad \text{ProjDur} = \text{ProjDur} + \text{ActDuration}(i) \\
\quad \text{ProjCost} = \text{ProjCost} + \text{NormCost}(i) \\
\text{End If}
\]

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.

![Network Diagram]

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.

<table>
<thead>
<tr>
<th></th>
<th>s</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>t</th>
</tr>
</thead>
<tbody>
<tr>
<td>s</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>t</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

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.
  - $B(i) \leq \text{ActDur}(i) - \text{CrashDur}(i)$
- The total project time must be less than or equal to the desired project time.
  - $u(t) - u(s) \leq P$

- Start time of end $-$ Start time of beginning $\leq$ 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 \(i\).

\[
\text{start time}(j) \geq \text{start time}(i) + \text{ActDur}(i) - \text{CrashTime}(i)
\]

**Objective Function:**

Minimize total project cost = \(\text{SUMPRODUCT}(\text{decision variables, crash slopes})\)

\[
(\text{crash slope } (i) = \text{ABS}((\text{CrashCost}(i) - \text{NormCost}(i)) / (\text{ActDur}(i) - \text{CrashDur}(i)))
\]

**Model Formulation:**

Minimize \(\sum_i B_i S_i\), \(S_i = \text{crash slopes or cost per time unit crashed}\)

Subject to:

- \(B_i \leq A_i - C_i\), \(A_i = \text{activity time and } C_i = \text{crash time}\)
- \(u_t - u_s \leq P\)
- \(u_j \geq u_i + A_i - B_i\)
- \(B_i \geq 0\)
- \(u_i \geq 0\)

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

**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

**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.

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.

![Figure CS12.4 (a)](image1.png)

Figure CS12.4 (a)

![Figure CS12.4 (b)](image2.png)

Figure CS12.4 (b)
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.

Figure CS12.5 (a)

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).]
CASE STUDY 12  ■  Critical Path Method

Figure CS12.6 (a)

Figure CS12.6 (b)

Figure CS12.6  The output sheets.

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

![Network Form](image)

Figure CS12.9  The network input form.

<table>
<thead>
<tr>
<th>Summary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Main options form</td>
<td>Where the user chooses between the CPM and time-cost tradeoff options.</td>
</tr>
<tr>
<td>Network form</td>
<td>Where the user enters the number of activities and events in the network.</td>
</tr>
<tr>
<td>Navigational buttons</td>
<td>“End,” “View Network,” “See Example,” “Return to Network Sheet,” “Return to Activity Table,” “Return to Precedence Matrix.”</td>
</tr>
<tr>
<td>Functional buttons</td>
<td>“Continue” and “Solve.”</td>
</tr>
</tbody>
</table>

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.
**CASE STUDY 12  ■ Critical Path Method**

**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 options 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.

```
Option Explicit
Option Base 1

Public nEvents As Integer, nActivities As Integer, i As Integer, j As Integer,
Uby As Object, UstartCell As Range, UendCell As Range, UstartAsDouble,
GcAs Range, DcAs Double, VcAs Double, VrAs Double,
StAs Range, DsAs Double, UendAs Double, VtAs Double,
VsAs Double, VrAs Double, LmiAs Double, LmAs Double,
Lmax As Double, Lmin As Double, EsAs Double, EiAs Double,
RcAs Range, RAs Integer, EjAs Double, EjcAs Double

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 CS12.11**  The Demo procedure.

```vbnet
Sub Demo()
   “This sub runs the demo using data from example sheet
   Call ClearPrevious
   Default = True
   frmMain.Show

   Application.ScreenUpdating = False
   If SolveType = 1 Then
      Worksheets("Example").Range("E:Extended").Copy
      ActiveSheet.Paste Destination:=Worksheets("Network Sheet").Range("B1")
      Range("B6:C6").Copy "CrashStart"
      Range("D:CrashStart").Offset(20, 1).Clear
      Worksheets("Example").Range("F:SampleTable").Columns(3).Range("F:SampleTable").Columns(4).Copy
      ActiveSheet.Paste Destination:=WtStartCell
      Worksheets("Example").Range("G:Extended").Copy
      ActiveSheet.Paste Destination:=WtStartCell
      Else
      Worksheets("Example").Range("E:Extended").Copy
      ActiveSheet.Paste Destination:=Worksheets("Network Sheet").Range("B1")
      Range("C:CopyOut").Copy
      Range("C:CopyOut").Range("C:CopyOut").Offset(20, 1).PasteSpecial xlPasteFormats
      Worksheets("Example").Range("F:SampleTable").Copy
      ActiveSheet.Paste Destination:=WtStartCell
      Worksheets("Example").Range("G:Extended").Copy
      ActiveSheet.Paste Destination:=WtStartCell
   End If
End Sub
```

"The demo has been created. Demo values will be filled in on all following sheets as well." & vbCrLf & "Please press the Continue button on the next to sheets, followed by the Solve button."
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.

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 CreateTable() 'This sub is called by pressing Continue on Network Sheet  
If Default = False Then 
    frmNetwork.Show 'number of events and activities firm 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 
        Application.UseThemeColor = True  
        Range("StartCell", Range("StartCell").Offset(1, i)).Value = i  
    Next i  
    If SolveCM Then 'only show crash columns if solving crash problem  
        Range.Range("CrashStart"), Range("CrashStart").Offset(1, 1).Value = "Crash Duration"  
        Range("CrashStart").Offset(1, 1).Value = "Crash Cost"  
        Range("CopyCol1").Copy  
        Range.Range("CrashStart"), Range("CrashStart").Offset(1, 1).PasteSpecial xlPasteFormats  
    End If  
  
    Worksheets("Precedence Matrix Sheet").Activate 'creates Precedence Matrix  
    If SolveCM Then  
        Range(RStartCell, RStartCell.Offset(nActivities, nEvents)).Interior.ColorIndex = 2 
        Range(RStartCell, RStartCell.Offset(nActivities, nEvents)).Borders(alsInsideHorizontal).Weight = xlThin 
        Range(RStartCell, RStartCell.Offset(nActivities, nEvents)).Borders(alsInsideVertical).Weight = xlThin 
        For i = 1 To nEvents 
            RStartCell.Offset(i, 1).Value = 1 
            RStartCell.Offset(i, nActivities).Value = 1 
        Next i  
        Range(RStartCell.Offset(1, 1), RStartCell.Offset(1, nActivities)).Interior.ColorIndex = 56 'upper-triangular matrix  
        Next i  
    Else  
        Range(RStartCell, RStartCell.Offset(nActivities + 1, nActivities + 2)).Interior.ColorIndex = 2 
        Range(RStartCell, RStartCell.Offset(nActivities + 1, nActivities + 2)).Borders(alsInsideHorizontal).Weight = xlThin 
        Range(RStartCell, RStartCell.Offset(nActivities + 1, nActivities + 2)).Borders(alsInsideVertical).Weight = xlThin 
        For i = 1 To nActivities 
            RStartCell.Offset(i, 1).Value = i 
            RStartCell.Offset(i, nActivities + 1).Value = i 
        Next i  
        Range(RStartCell.Offset(1, 0), RStartCell.Offset(nActivities, 0)).Interior.ColorIndex = 56 'lower-triangular matrix  
        Next i  
        RStartCell.Offset(0, 0).Value = "A"  
        RStartCell.Offset(1, 1).Value = "A"  
        RStartCell.Offset(2, 2).Value = "B"  
        RStartCell.Offset(3, 3).Value = "B" 
        Range(RStartCell.Offset(nActivities + 1, nActivities + 1), RStartCell.Offset(nActivities + 1, nActivities + 1)).Interior.ColorIndex = 9 
        Range(RStartCell.Offset(nActivities + 2, nActivities + 2), RStartCell.Offset(nActivities + 2, nActivities + 2)).Interior.ColorIndex = 9  
    End If  
End If  

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

Figure CS12.13 The CreateTable procedure.
If the time-cost tradeoff problem is solved, then the application calls the \textit{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.
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.
CASE STUDY 12  ■  Critical Path Method

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 CM option
Application.ScreenUpdating = False

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

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

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

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

Worksheets("Report Sheet").Activate
Worksheets("Report Sheet").Visible = True
Range("All").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.
Figure CS12.19 The CrashLP procedure.
Figure CS12.20  The `ClearPrevious` procedure.

Figure CS12.21 presents the navigational procedures.

```vba
Sub ClearPrevious()
    'class previous data
    Application.ScreenUpdating = False
    SolveCRM = False
    TradeDone = False
    MinSlope = 999999
    NumCrashed = 0
    Iter = 1
    DSStop = False
    Worksheets("Activity Sheet").Activate
    Range(StartCell).Offset(1, 0, "B9").ClearContents
    Worksheets("Precedence Matrix Sheet").Activate
    Range(StartCell).Offset(0, 0, "AA100").ClearContents
    .Interior.ColorIndex = 0
    .Borders[allinsidehorizontal].LineStyle = mMemo
    .Borders[allinsidevertical].LineStyle = mMemo
End With
    Worksheets("Report Sheet").Activate
    Range(StartCell).Offset(0, 0, "1100").ClearContents
    Range("ProDCost").Offset(1, 0).ClearContents
    Range("ProDOut").Offset(4, 0).ClearContents
    Worksheets("TradeOff").Activate
    Range(StartCell).Offset(1, 0).Range("TradeStart").Offset(0, 2).End(xlDown).ClearContents
    .Font.Strikethrough = False
    .Font.ColorIndex = 1
End With
    Worksheets("Network Sheet").Activate
    Range("A18:134").Graph, Delete
    ActiveSheet, Paste Destination:=Worksheets("Deleted").Range("B3")
    Worksheets("Deleted").Cells, Delete
    ActiveSheet.Shapes("RetAct").Visible = False
    ActiveSheet.Shapes("PastTree").Visible = False
    ActiveSheet.Shapes("SetReport").Visible = False
    Worksheets("Network Sheet"), Range("H2").ClearContents
    Worksheets("Activity Sheet"), Range("FN").ClearContents
    Worksheets("Precedence Matrix Sheet"), Range("C2") .ClearContents
End Sub
```

Figure CS12.21  The navigational procedures.
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?