Visualizing solutions in network analysis with visual basic for applications

Larry J. LeBlanc∗

(received October 2003; accepted November 2003)

Network models have been very useful for analyzing problems involving telecommunications, supply chains, vehicle routing, and other areas. In this paper I discuss the importance of visualization for network optimization. Reasons include explaining the model to non-technical managers and obtaining better solutions for the network model. Better solutions can be a direct result of visualization (when human intuition identifies a solution from the network picture) or an indirect result (when human intuition identifies improved procedures in algorithmic design, which then result in superior solutions). I discuss the importance of Visual Basic for Applications (VBA) for drawing networks directly within Microsoft Excel. I show how to use VBA efficiently to draw these networks using various Excel drawing objects.

Keywords: Visual interactive optimization; Network optimization; Network graphing; Supply chains; Visual Basic for Applications

1. Introduction

Network models are essential for many firms with telecommunication systems, supply chains, vehicle delivery systems, etc. These network models are the basis for analyzing optimal facility locations, flows, and capacities for communications, freight shipments, and vehicle routings. Most large firms have extensive telecommunication requirements as an inherent part of their business. For example, a bank has ongoing remote database queries regarding account-balances, withdrawals and deposits to record, and updates to remote databases such as new customer information and new addresses for existing ones. An insurance company faces similar queries regarding quotes for new customers which require remote data base access. E-mail, file transfers, web-page access, video conferencing and image transfers are other examples. These firms need communication networks with sufficient link capacities (bandwidth) to support their communication requirements. An ongoing important management problem is to find the topology and capacity of the network, i.e., to choose capacity for each possible link, and to find the best routes for data traffic over this network.

A firm’s telecommunications facilities are usually modeled by a net- work whose nodes represent server sites and locations where traffic originates and terminates [13-14, 27-28, 30-33, 41, 46-47]. These include the firm’s own offices and Internet and other access facilities (both dial-up and dedicated access) for vendors and customer sites. Network links represent either physical connections between pairs of locations, or logical connections, i.e., a transmission path. Manufacturers have also realized the importance of efficient coordination of their supply chains [5-6, 18-19, 25-26, 34, 45, 48]. This includes forward supply chains (manufacturer to central warehouses to local warehouses to retail stores) and backward supply chains (materials suppliers to the various tiers of vendors to the manufacturer). Supply chain network models use nodes to represent locations (warehouses, vendors and retail stores), and links to represent the flows of materials and information. Similarly, railroad networks are used to determine cost effective routing for locomotives and rail cars, and urban road networks are used to determine efficient routes for vehicles and to predict traffic flows [4, 8-9, 11-12, 17, 29, 35, 38-39, 43-44].

2. Network Optimization

There has been enormous research effort addressed at developing efficient algorithms for solving algebraic optimization models for improved design of networks and for routing within a given network. An algebraic optimization model is a quantitative representation of the problem using numbers, variables, equations and inequalities to form the objective function and constraints. 0-1 decision variables are often used to model the selection of links and/or nodes to include in the network and (in cases where a single route must be selected) the route to use. Algebraic models are very helpful, but they are abstract and difficult to explain to non-technical managers. In this case, visualizing the network model is helpful for explanation to managers.

Furthermore, it is well known that in many cases large models cannot be solved to exact optimality. In this case, visualization of the network can be valuable directly in the solution process. For example, the telecommunications network shown at the top of Figure 1 would benefit from adding links (5,6), (6,7), and (7,8), which are currently un-chosen, and simultaneously deleting the much longer link (3,9) which is proposed and chosen. However, deleting link (3,9) and adding any one of the three shorter links individually would not be feasible, since the network would then be un-connected. A pair-wise swap algorithm would fail to realize this; a human observer, however, could guide the algorithm by directing that all three
shorter three links be included and link (3,9) be deleted. In cases such as this, human intuition is superior to a computer in combinatorial issues. Obviously, the opposite is true in most other cases involving a combinatorial nature.

The transportation network shown at the bottom of Figure 1 gives a similar example of adding link (6,9) instead of links (5,8), (8,11) and (11,14).

Network visualization is also helpful to supply chain managers when choosing vendors and mode of transportation. For example, Kia Motors sources parts for its Sorento Sport Utility Vehicle from locations as diverse as Wales, Mexico, Malaysia, Thailand, and the U.S. [37] transporting components through the Suez Canal or around the Straits of Malacca. Furthermore, components move through numerous countries before final assembly in South Korea. Visualization (Figure 2) by plotting various manufacturing plants (nodes) and connections makes it much easier and faster to understand changes to this complex supply chain to react to contingencies and ensure a smooth manufacturing operation. Visualization
also makes it much easier to look for redundancy in the supply chain and inefficient transportation strategies.

Figure 2 Example Supply Chain—Chosen and Un-Chosen Transport Links

Visualization can also be very helpful indirectly in the solution process by assisting in algorithmic design. For example, viewing the tentative network at each iteration of a proposed algorithm can help the management scientist realize what different steps are needed by the algorithm. I discuss this further in section 5.

Other authors have discussed the importance of graphical user interfaces (GUIs), including the importance of visualization when designing networks. Pirkul et al. [42] discuss the importance of humans complementing numerical algorithms for the P-median problem. This is a type of network design involving facility location and assignment on a network. They discuss VisOpt, a visual aid developed originally in Turbo Pascal and later in Visual C++ for the P-median problem. Ogryczak et al. [40] describe an interactive optimization system for transshipment problems with linear cost costs for link flows and fixed costs for new nodes. Users can view the solution to the model using text-based graphics, i.e., typed ASCII letters and symbols such as hyphens to denote the locations of nodes and links. In classical early work, Glover et al. [15-16] discuss the importance of using the pictorial nature of networks. Dean et al. [10] also describe an interactive graphics system, written in the C programming language, for network
modeling and optimization. Steiger et al. [49] describe a graphical interface for network modeling, which involves a simultaneous display of the model overview and network details for help in model validation. More recently, Kim et al. [23] develop an algorithm for developing trees as an aid to understanding navigational paths in a proxy server.

There is a significant advantage of using customized software developed by the user for his/her specific purpose. This allows the user to highlight certain network links or nodes with various colors depicting different customer types, etc. Furthermore, developing software in VBA within Excel, with its greater user-friendliness for many analysts than the C programming language, has obvious advantages. Thus, rather than use software for the specific types of problems referred to in the previous paragraph or developed in C, VBA/Excel offers distinct advantages. In the next sections I give an overview of VBA within Microsoft Excel and discuss its application for efficient drawing of networks.

3. Overview of VBA

Microsoft’s Visual Basic® for Applications (VBA) is a programming language which is very helpful for management science applications within Microsoft Excel (and Access, PowerPoint and Word). Because VBA’s development environment is integrated into Excel, programmers can easily work with data, viewing intermediate results directly in Excel.

VBA for Excel combines the original BASIC programming language [1-3, 7, 20-22, 36] with new features inherent to Excel, such as those referring to the values and formats for Excel ranges, drawing objects such as lines and textboxes, fields and buttons within custom dialog boxes, cell comments, Excel charts, etc. Because VBA is an object-oriented programming language, users can easily manipulate these features within Excel. Furthermore, because VBA is integrated into Excel, all of Excel’s functions can be used directly within VBA. For example, rather than use a loop to find the average, maximum, and minimum values of a range of data, VBA can invoke Excel’s AVERAGE, MAX, and MIN worksheet functions. In cases where one must determine which interval contains a given number, VBA can use the very powerful Excel worksheet functions HLOOKUP or VLOOKUP. An example of this is looking up a given shipment quantity in a shipping rate table with values for 1 – 10 kilos, 10 – 50 kilos, 50 – 100 kilos, etc. In contrast to VBA, Microsoft’s Visual Basic® (VB) is a stand-alone application that lacks the efficiency of being able to work directly within Excel. Similarly C++ and FORTRAN lack these advantages of VBA.
VBA’s fundamental control logic, shown in Figure 3, consists of Subroutine definition and Ending statements, Dimension statements to declare variable types, statements for assigning values to variables, For...Next loops, If Then Else logic, GoTo statements, and statement labels.

4. VBA for Drawing Networks

Networks can be drawn very efficiently using VBA in Excel by using standard Excel generated drawing objects. The network nodes can be represented using geometric shapes such as rectangles, ovals, or textboxes, using embedded text to identify the node. Links can be represented using straight lines. The logic for drawing a network, either the final result or an intermediate solution from a network design algorithm, has three fundamental steps:

i. Initialize For each node, read the X and Y coordinates, traffic demand, type of node, etc.

ii. Draw Nodes Add a drawing object such as a textbox or circle representing each node at the corresponding coordinates.

iii. Draw Links For each link (i, j), draw a line from the tail node i to the head node j.

When drawing the nodes, the symbol representing node i can contain the number i or the name of node i for identification purposes. Different symbols or colors or border widths can be used to represent different types of nodes, such as supply vs. demand vs. transshipment points in a supply chain network and company offices vs. client sites in a telecommunications network. When plotting links, different line thicknesses or styles (solid, dashed, dotted) can be used to represent the amount of flow on the link or the type of link (high speed vs. low speed).

Figure 4 shows a completed network created using VBA code with textboxes to represent nodes. Using this figure as a reference, I show how to build a network with VBA for Excel using drawing objects. Although this example has only 20 nodes for simplicity, I have used the code to draw much larger networks. I discuss this in Appendix B.

---

1 A “Delete” step can be inserted between “Initialize” and “Draw Nodes” to delete any existing network drawing before a new network is drawn. The details of this operation are given in Appendix A.
Define the beginning of a new subroutine

Sub Demo()

Dimension the array “NodeFlow” with 2,000 integer entries

Dim NodeFlow(2000) As Integer

Assign the value of the VBA variable “Nodes” to equal the value in cell B3

Nodes = ActiveSheet.Range("B3").Value

Inform the Excel user with a message (dialog) box

If Nodes > 2000 Then
    MsgBox ("There are too many nodes to process")
    GoTo MyLabel
Else
    Jump to statement labeled “MyLabel”
End If

Multiple line If statement needs “End If”

If Nodes > 1500 Then MsgBox ("Long processing due to number of nodes")

Single line If statement has no “End If” statement

For i = 1 To Nodes
    Begin a loop for i = 1, 2, 3, …, Nodes
Next i

MyLabel: MsgBox ("Aborting Execution")

Define the end of the current subroutine

End Sub

Figure 3 Common VBA Statements
4.1 Initialization

The first step is to save the coordinates of each node. These will be used to identify the locations, in points, of the upper left corners of the text-boxes that represent the nodes. In Excel, there are 72 points per inch (approximately 28 points per centimeter) for drawing objects when they are printed. Figure 5A shows that the origin of the coordinates is at the upper left corner of the first cell in the Excel window. Thus, to move a node lower on the screen, its Y-axis coordinate is increased. This is in contrast with conventional axes (Figure 5B) in which decreasing the Y-axis coordinate lowers the node.
In Figure 4, the Excel range C2:G21 contains the node names, coordinates, and the other nodes that each node is connected to. In the VBA code shown below, these coordinates are stored in the array variables XCoord() and YCoord(). These arrays are initially Dimensioned as single precision with unknown size (a “dynamic array”). In the next step, the actual number of nodes to be drawn is determined, and only then is the size of these arrays specified. The VBA variable “Nodes” represents the number of nodes and therefore textboxes in the network. This variable, which in this case equals 20, is initialized in VBA from cell A2. After determining this number of nodes, the sizes of the array variables are ReDimensioned to allow storing data for each node. In Figure 4, columns D – G contain the node names, coordinates, and connected nodes, respectively. A For…Next loop is used to input this data for each node in the Excel range, storing them in the appropriate arrays.

'Dimension these as arrays of unknown size. "Single" means single precision:

Dim XCoord() As Single, YCoord() As Single, ToNode() As Single
Dim BoxWidth() As Single, BoxHeight() As Single, NodeName() As String

'Read the number of nodes from A2 in Excel; re-dimension each array to this size:

Nodes = ActiveSheet.Range("A2").Value
ReDim XCoord(Nodes), YCoord(Nodes), ToNode(Nodes), BoxHeight(Nodes)
ReDim BoxWidth(Nodes), NodeName(Nodes)

'Read the node names, coordinates, and connections to other nodes from D1...:

For i = 1 To Nodes
    NodeName(i) = ActiveSheet.Range("D1").Offset(i, 0).Value
    XCoord(i) = ActiveSheet.Range("E1").Offset(i, 0).Value
    YCoord(i) = ActiveSheet.Range("F1").Offset(i, 0).Value
    ToNode(i) = ActiveSheet.Range("G1").Offset(i, 0).Value
Next i

4.2 Draw Nodes Using TextBoxes

Using the stored X and Y coordinate values, each node’s textbox can be created and located on the Excel worksheet. This is done using the following VBA commands embedded in a For...Next Loop.
Place textboxes representing nodes. Upper left coordinates are (XCoord, YCoord):

For i = 1 To Nodes

   Textbox width and height = any number. They will be reset using “AutoSize.”

   ActiveSheet.TextBoxes.Add(XCoord(i), YCoord(i), 1, 1).Select
   Selection.Characters.Text = NodeName(i) 'Put the name of node i in the ith textbox
   Selection.HorizontalAlignment = xlCenter     'Center the name in the textbox
   Selection.AutoSize = True                               'Size the textbox to fit the text
   BoxHeight(i) = Selection.Height                     'Store box height and width for later
   BoxWidth(i) = Selection.Width                       'centering of links

Next i

The “Add” method requires the X-axis and Y-axis coordinates, as well as the width and height dimensions of the textbox. The width and height of the textbox can be set to any number (1 in the code above), since they will be automatically adjusted later to the number of characters contained in it using the “AutoSize” method.

4.3 Draw Links Using Lines

Adding lines for each link also requires the X- and Y-axis coordinates for both the beginning and ending of the line. This is also accomplished using the “Add” method. Because the lines are the last objects added they will appear on top of each textbox. For cosmetic reasons, it is desirable to place them behind the textboxes, as shown in Figure 6. These lines can be placed behind each textbox by using the ZOrder method with the attribute, “msoSendToBack.”

Place the lines representing the links:

   For i = 1 To Nodes
       j = ToNode(i)                               'The line points from node i to node j
       If j = 0 Then GoTo SkipNode

       'Plot line from CENTER of boxes, so offset by half the height and width:
       ActiveSheet.Lines.Add(XCoord(i) + BoxWidth(i)/2, YCoord(i) + BoxHeight(i)/2, _
                                XCoord(j) + BoxWidth(j)/2, YCoord(j) + BoxHeight(j)/2)_
                                .Select

       Selection.ShapeRange.Line.Weight = 2               'Make the line thick
       Selection.ShapeRange.ZOrder msoSendToBack  'Put the line behind the textboxes

   SkipNode: Next i
The execution of these commands produces the network in Figure 4, which is a visual representation of the node coordinates and connecting links shown in the worksheet range C2:G21. The completed VBA code is summarized in Appendix C.

Most programmers object to excessive use of “go to” statements, and some programmers object to any use at all of this statement. In my opinion, however, occasional use of these statements is perfectly acceptable. For readers wishing to avoid these statements, the line:

```
If j = 0 Then GoTo SkipNode
```

can be replaced with:

```
If j <> 0 Then
    'Plot line from CENTER of boxes, so offset by half the height and width:
    ActiveSheet.Lines.Add (etc.—continue with logic above)
End If
```

5. Visualization for Use in Algorithm Design

I illustrate the importance of visualization with an example of designing an algorithm for optimizing telecommunication virtual private networks (VPNs). These are used for secure communication of voice, data, digital images, and video, etc. among various corporate offices, remote employees, key vendors, and customers. In [41], my co-authors and I developed an optimization model for the problem of VPN design using a combination of 1) leased lines and 2) switched multi-megabit data services (SMDS). With SMDS, data packets are inserted into the network without specifying a path just like letters get dropped off at a post office. We showed that substantial cost savings are possible by exploiting the difference between the two services with respect to their required network topologies, available link band-widths, and tariff structures.

In the optimization model, node 0 represents the SMDS service, and
nodes 1, 2, ..., n each represent a company or partner office site (called a
network node). The communication demand at each network node is a given
constant. The model has variables denoted by Flow$_{i0}$ and Flow$_{ij}$. These
respectively denote the one-way flow inserted into the SMDS system at each
node i and the two-way flow on leased lines between nodes i and j. The
model is denoted by Problem P:

\[
\text{MINIMIZE } \sum_{\text{network nodes } i} \text{SMDSCost}(\text{Flow}_{i0}) + \sum_{\text{network nodes } i, j} \text{LeaseLineCost}_{ij}(\text{Flow}_{ij})
\]

The first term in the objective function is the cost of SMDS services
from each node i. Of course, this cost depends on the total number of
packets or bytes inserted at node i, Flow$_{i0}$. The second term is the cost of
leased lines between each pair of nodes i and j, which also depends on the
flow that the line must accommodate. Constraints (which I do not show here)
use 0-1 variables to require that each node must be connected either directly
to the SMDS node or to another network node. Additional constraints are
conservation of flow at the network nodes. This problem is known to be NP-
Complete, and so it cannot normally be solved to optimality.

In [41], we developed a dedicated simulated annealing (SA) technique
for solving Problem P. SA is a generic heuristic algorithm for solving dis-
crete optimization models [24, 32]. The SA algorithm makes random
changes to a small number of variables in a current solution to obtain a
neighbor solution. If the neighbor has improved objective function value, it
becomes the new current solution for the next iteration. Even if the neighbor
has worse objective function value, it becomes the new current solution with
a certain probability. The reason for occasionally accepting worse solutions
is that the chance of getting stuck at a local optimum is decreased. If SA
begins to converge to a non-optimal solution, subsequent random moves can
lead to another solution from which SA can converge to the optimal solution.

We developed the five annealing types shown in Figure 8. At each SA
iteration, one of these annealing types is randomly chosen, so each annealing
type is typically used many times for each problem. However, when design-
ing the SA algorithm’s moves shown in Figure 8, we did not have the benefit
diagram for visualization. If we had been able to visualize the intermed-
iate solutions obtained by SA, our algorithmic development time would have been much faster. For example, Figure 7A shows a possible final solution obtained by SA using only the first three annealing types shown in Figure 8. From Figure 7B, nodes 8 – 11, it is clear that annealing type 4 would result in an improved solution.

Figure 9 shows the excellent solution that we were able to obtain for a different telecommunication problem—access network design [43]. We attribute the success of our algorithmic development for the model for this problem to the use of visualization of the network at the intermediate iterations of the simulated annealing algorithm.

Figure 7 Network Solutions from Simulated Annealing

<table>
<thead>
<tr>
<th>Type</th>
<th>Before Annealing</th>
<th>After Annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1" alt="Before Annealing 1" /></td>
<td><img src="image2" alt="After Annealing 1" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image3" alt="Before Annealing 2" /></td>
<td><img src="image4" alt="After Annealing 2" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image5" alt="Before Annealing 3" /></td>
<td><img src="image6" alt="After Annealing 3" /></td>
</tr>
<tr>
<td>4</td>
<td><img src="image7" alt="Before Annealing 4" /></td>
<td><img src="image8" alt="After Annealing 4" /></td>
</tr>
<tr>
<td>5</td>
<td><img src="image9" alt="Before Annealing 5" /></td>
<td><img src="image10" alt="After Annealing 5" /></td>
</tr>
</tbody>
</table>

Figure 8 Annealing Types for Access Network Design Problem

<table>
<thead>
<tr>
<th>Type</th>
<th>Before Annealing</th>
<th>After Annealing</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1" alt="Before Annealing 4" /></td>
<td><img src="image2" alt="After Annealing 4" /></td>
</tr>
<tr>
<td>2</td>
<td><img src="image3" alt="Before Annealing 2" /></td>
<td><img src="image4" alt="After Annealing 2" /></td>
</tr>
<tr>
<td>3</td>
<td><img src="image5" alt="Before Annealing 3" /></td>
<td><img src="image6" alt="After Annealing 3" /></td>
</tr>
</tbody>
</table>
6. Conclusions

Network models are useful for many different application areas, yet they can be difficult to solve to exact optimality. For this reason, visualization is a very valuable tool for obtaining solutions and for explaining the model to non-technical managers. Visual Basic for Applications within Microsoft Excel has features which make it ideal software for drawing such networks for visualization of the process. I have shown how to use these features efficiently for such purposes.

A. Starting Cluster Solution (Switch Connections Suppressed for Clarity)
B. Final Solution

Figure 9 Starting and Final Solution from Simulated Annealing Algorithm for Typical Telecommunication Access Design Problem
References


[16]. ______. D. Klingman, C. McMillan. 1977. The NETFORM concept: A


Appendix A—Details of “Delete” Operation

Deleting the drawing an existing network is best accomplished using the “Name” property of each “Shape” object in the Excel worksheet. The system-generated names for buttons, textboxes and straight lines begin with “Button”, “Text” and “Line,” respectively. Using a For...Each loop to scroll through all shapes in the worksheet allows you to decide whether or not to delete each one. This is done using an If...Then conditional statement as seen in the code below. The VBA variable “MyShape” can be any user defined index for the loop. The function “Left(MyShape.Name, 6),” which returns the first 6 characters of the text, is used to identify whether or not the object is a button for running the VBA code. This will prevent the button for running the VBA code from being deleted along with all the other shapes.

For Each MyShape In ActiveSheet.Shapes
    If Left(MyShape.Name, 6) <> “Button” Then MyShape.Delete
Next MyShape

<> means not equal.

Appendix B—Solution Times for Different Size Networks

I tested the VBA for drawing random networks with number of nodes ranging from 500 to 5,000. Elapsed times on a 1.8 GHz PC are shown in the figure below. The “detail” times shown include times for centering text in the textboxes and auto-sizing the textboxes, as discussed in section 4.2. The times “Time 1” omit these steps and are approximately half of the detailed times.
Appendix C—Complete Draw-Network VBA Routine

Sub DrawNetwork()

'Dimension these as arrays of unknown size. "Single" means single precision:
Dim XCoord() As Single, YCoord() As Single, ToNode() As Single
Dim BoxWidth() As Single, BoxHeight() As Single, NodeName() As String

'Read the number of nodes from A2 in Excel; re-dimension each array to this size:
Nodes = ActiveSheet.Range("A2").Value
ReDim XCoord(Nodes), YCoord(Nodes), ToNode(Nodes), BoxHeight(Nodes)
ReDim BoxWidth(Nodes), NodeName(Nodes)

'Read the node names, coordinates, and connections to other nodes from D1...:
For i = 1 To Nodes
    NodeName(i) = ActiveSheet.Range("D1").Offset(i, 0).Value
    XCoord(i) = ActiveSheet.Range("E1").Offset(i, 0).Value
    YCoord(i) = ActiveSheet.Range("F1").Offset(i, 0).Value
    ToNode(i) = ActiveSheet.Range("G1").Offset(i, 0).Value
Next i

'Delete any existing network (textboxes & lines) but not the VBA button:
For Each MyShape In ActiveSheet.Shapes
    If Left(MyShape.Name, 6) <> "Button" Then MyShape.Delete
Next MyShape

    "Offset" by i rows and no columns from cell G1

    "SkipNode" is the label shown below

    Text box width and height = any number (reset later using "AutoSize.")

'Place textboxes representing nodes at coordinates (XCoord, YCoord):
For i = 1 To Nodes
    ActiveSheet.TextBoxes.Add(XCoord(i), YCoord(i), 1, 1).Select
    Selection.Characters.Text = NodeName(i) 'Put node i name in ith textbox
    Selection.HorizontalAlignment = xlCenter 'Center the text in textbox
    Selection.AutoSize = True 'Size the textbox to fit the text
    BoxHeight(i) = Selection.Height 'Store box height and width for later centering of links
    BoxWidth(i) = Selection.Width
Next i

'Place the lines representing the links:
For i = 1 To Nodes
    j = ToNode(i) 'The line points from node i to node j
    If j = 0 Then GoTo SkipNode
Next i

542
'Plot line from CENTER of boxes, so offset by half the height and width:
ActiveSheet.Lines.Add(XCoord(i) + BoxWidth(i)/2, YCoord(i) + BoxHeight(i)/2,_,
                     XCoord(j)+ BoxWidth(j)/2, YCoord(j)+ BoxHeight(j)/2) _ Select
Selection.ShapeRange.Line.Weight = 2        'Make the line thick
Selection.ShapeRange.ZOrder msoSendToBack    'Put line behind textbox
SkipNode: Next i

End Sub