Class Contribution:
Excel Add-In for Solving the Shortest Path Tree and the Optimum Tree Problems

By Robert Mahfoud
December 4th 2001
**Introduction:**

This project is the class participation for the class ORI391Q Integer Programming. It consists of a MS Excel add-in that extends the Jensen add-ins to solve two additional classes of problems, namely the shortest path tree problem, and the optimum tree problem with undirected edges. It was designed following the model of the Jensen add-ins package, and has the same look-and-feel. In summary, the shortest path tree problem is the problem of finding the minimum cost (length) path between an origin node and all the other nodes in a digraph. On the other hand, the optimum tree with undirected edges problem is the problem of finding a tree that spans all the nodes in an undirected graph, with optimum (minimum or maximum) length. The two variations of this problem are the minimum spanning tree (when minimizing) and the maximum weight tree (when maximizing) problems.

**General Description:**

The add-in can be started as any other Excel add-in. Once it is started, two items are added under the OR_MM drop-down menu. These two menu items are labeled "Shortest Path" and "Optimum Tree". By clicking on any of these 2 items, a new problem can be created in a new worksheet.

**Shortest Path Tree:**

The problem worksheet looks very much like a Network problem of the Math Programming add-in, however the number of input parameters per arc are limited to the start and termination node, in addition to the arc cost. While for the nodes, no balance data is needed, however, it is of interest to know the shortest path from the origin to a particular node, as well as the path length. This information will be shown next to the each node after the problem is solved, and the shortest path tree is built. Note that the implementation used is that of Djikstra's algorithm. The dialog that is used to create the problem requests the following input from the user:

- Name of the problem (which will be the name of the new)
• Number of nodes
• Number of arcs
• The origin node (the shortest path will be calculated from this node to all other nodes in the network)
• Whether to generate a random problem or not

Once the problem is generated, the user can go on and fill the appropriate data that define the network. Each arc in the network must have its start and end node defined, and must also have a nonnegative cost (required by Djikstra's algorithm). The network must be consistent, in that the node numbers included as origins and destinations must actually be in the network.

For more details on what kind of exceptions that might occur due to inconsistent definition of the network (and other user input), and what is the outcome of the problem in each case, please refer to the technical description of the project below.

In case the user wants to change the number of nodes and/or arcs in the network, he can click on the "Change" button, and the same dialog box will appear again, giving him access to change all the parameters (less the problem name). Note that in this "Change" instance of the shortest path dialog, the "Make Random Problem" check box is turned off by default, so the user will not inadvertently delete any network data he has entered if he presses OK.

Once the user has completely defined the network in terms of connectivity and arc costs, he can proceed and solve the problem to build the shortest path tree. The shortest path tree is represented through the "In Tree?" column, as follows: if an arc is included in the shortest path tree, the "In Tree?" column has the value of 1, otherwise, it has the value of 0. Regarding the path to each node, it is shown in the list next to each node. The first value is the cost from the origin to this particular node. The next value is the "predecessor" node, i.e. the node that precedes this node in the tree, while the remaining list of nodes is the whole path, node by node, from the origin to the destination.
**Optimum Tree with Undirected Edges:**

The second problem type that can be solved with this add-in is the optimum tree with undirected edges. The input data to this problem is very similar to that of the shortest path problem. The dialog that is used to create the problem requests the following input from the user:

- Name of the problem (which will be the name of the new)
- Number of nodes
- Number of arcs
- The optimization direction (min or max, which correspond to the minimum spanning tree and the maximum weight tree problems respectively)
- Whether to generate a random problem or not

Once the skeleton worksheet is generated, the user can go on and input the tree connectivity and cost data in the appropriate fields. The user can also use the "Change" button to modify the structure of the problem, and the same considerations for the name of the problem and the random generation in the shortest path dialog apply here as well.

Once the data input is complete, the user can press the "Solve" button to get the optimum tree. The resulting tree is represented in the same way as the shortest path, with the help of a "In Tree?" column next to each edge which takes the value of 1 is the edge in the tree, and 0 otherwise.

**Technical Description:**

This add-in (named ip_tree.xla) was programmed, as much as possible, to have the same functionality as the other OR_MM add-ins. Some remarks about the implementation will be mentioned below. Some other details are scattered in the code itself as remarks. Note that the code of the add-in is protected with the password "ppp" as for the other add-ins.
**Algorithms:**

The shortest path algorithm implemented here is that of Djikstra, which was adopted from the book Modeling Transport, 2nd Edition, by Ortuzar and Willumsen (1994). This algorithm is a dynamic programming approach to building the shortest path tree. The implementation is found in the module `modSP`, in the `ip_tree` project.

On the other hand, the algorithm used for the optimum tree problem is a greedy algorithm and was adopted from the book Integer Programming, by Laurence Wolsey, described on pages 44 and 45. The implementation is found in the module `modOPTTREE`.

**Exception Handling:**

Some of the exceptions that might due to user input are handled (most of the time following the same model of the other add-ins) so that the program will not crash in an unfriendly way for the user. Some examples of these cases are described below.

- If the user inputs a problem name that is the same as the name of an existing worksheet, or if the name contains characters that cannot exist in range name (like " , . ; ' etc…), he will be prompted that the name is invalid.
- If the user inputs a non-numeric value in a fields that expect numbers in the dialog box, an error message will appear, and the program will not crash
- Errors occurring while reading the input data on the worksheet (probably because a character string was found where a number was expected) are also handled.
- Any errors that might occur in the algorithms because of inconsistent user input are handled smoothly such that the execution will stop with a friendly user message (for example, if the user input an origin node that is beyond the range of available nodes in the shortest path problem).

These exceptions (and some other) are handled so that the add-in will be more tolerant to user-related errors, and consequently more friendly to use.
**Code Re-use:**

Since the Jensen add-ins were used as a model for writing this add-in, it was necessary to copy some of the code from the original add-ins and re-use it in the new one. Some of the functions copied were left intact, while others were significantly modified to accommodate the need of this add-in.

While reviewing the code, it is useful to note the following remarks:

- Some function names have been changed (even if sometimes the content was left intact) strictly because of personal taste in naming, and because this makes reading/writing the code easier to me.
- Most of the formatting functions (those that are used to create the visual look of the Jensen add-ins) have been re-used.
- Some functions were intended to be more general in the other add-ins, and to handle a greater variety of problems were truncated to produce only the output needed by this add-in (for example the function BuildArcs in the module Net_Mod of the project mp_models).
- Most of the changes in the functions re-used were due to the fact that very little global variables are used in the project (because they make the project less friendly to code), and instead variables are made available to the different functions strictly by argument passing.

It might as well be helpful to know the distribution of the functions in the different modules (in the project ip_tree):

- **modGlobals** contain some global definitions and some general-purpose functions.
- **modControl** contains the user interface functions (button-click procedures)
- **modSP, modOPTTREE** contain the algorithm implementations for the shortest path and the optimum tree problems respectively.
- **modWSSFormat** contains all the functions used to add a worksheet, and format the cells according to each problem.
- **modSolver** contain the function that control solution procedure and outputting of the results.