Branch-and-bound algorithm for the traveling salesman problem

The traveling salesman problem is discussed in Section 8.7 of the textbook. The branch-and-bound algorithm described in that section is slightly incomplete, so here is a careful description of an improved version of the algorithm.

The problem

The *traveling salesman problem* (TSP) is as follows: Given a list of cities and a table of distances from each city to the others, find a shortest circuit that visits each city exactly once and returns to the starting city.

Ingredients of the algorithm

The branch-and-bound algorithm for the traveling salesman problem uses a branch-and-bound tree, like the branch-and-bound algorithms for the knapsack problem and for solving integer programs.

- The node at the top of the tree is called the *root*. All edges (arrows) in the tree point downward. If an edge points from a node P to a node C, then P is called the *parent* of C, and C is called a *child* of P.
- Every node in the branch-and-bound tree has
  - a *node number*;
  - a *label*, representing the decision made at that node either to take or not to take a specific link from one city to another (the labels of the terminal nodes of the tree represent two links);
  - a *bound*, giving a lower limit on the possible lengths of circuits below that node in the tree;
  - an *incoming matrix*; and
  - an *opportunity matrix*.

  (Exceptions: The root node of the branch-and-bound tree does not need a label, and the terminal [leaf] nodes of the tree do not have incoming or opportunity matrices.)
- Every matrix (both incoming matrices and opportunity matrices) has an associated "L-value," which could be called the "deduction total." As the algorithm progresses, distances in the matrix are decreased, and the L-value keeps track of the total amount we have subtracted from distances since the original distance matrix.

Top-level outline of the algorithm

1. Draw and *initialize* the root node (see below for details).
2. Repeat the following step until a solution (i.e., a complete circuit, represented by a terminal node) has been found **and** no unexplored non-terminal node has a smaller bound than the length of the best solution found:
   - Choose an unexplored non-terminal node with the smallest bound, and *process* it (see page 2 for details about this step).
3. When a solution has been found **and** no unexplored non-terminal node has a smaller bound than the length of the best solution found, then the best solution found is optimal.

Initialization of the root node

- The *node number* of the root node is 0 (or 1 if you like; it doesn't really matter where you start counting).
- The root node doesn't need a *label*, but if you want to give it one you can label it "All circuits" as the textbook does, or use the label "$\varnothing$" because the node doesn't represent a decision, or you can label it "Root," or whatever.
- The *bound* of the root node is 0.
- The *incoming matrix* of the root node is the original matrix of distances, with M's along the diagonal; the L-value of this matrix is $L = 0$.

Steps to process a node

1. Compute the *opportunity matrix* for the node. Beginning with the node's incoming matrix:
   (a) Set $L =$ the incoming matrix's L-value.
   (b) Ensure that every row and every column has exactly one M. [The matrix might contain N's too, from step 6(b)—these don't count as M's.] If necessary, replace one entry of the matrix with an M to meet this condition. [The M's enforce acyclicity until a full circuit has been completed.]
   (c) Ensure that every column contains at least one zero. If necessary, subtract the smallest number in a column from every number in that column, and add the number that was subtracted to L.
   (d) Ensure that every row contains at least one zero. If necessary, subtract the smallest number in a row from every number in that row, and add the number that was subtracted to L.

   The resulting matrix is this node's opportunity matrix, and the L-value of this opportunity matrix is the value that L has after possibly being increased in steps (c) and (d) above.

2. [First special case.] Is the opportunity matrix 2×2? If so:
   (a) A 2×2 opportunity matrix will always look like either $\left[\begin{smallmatrix} M & 0 \\ 0 & M \end{smallmatrix}\right]$ or $\left[\begin{smallmatrix} 0 & M \\ M & 0 \end{smallmatrix}\right]$.
   (b) Create a single child node, and assign it the next available node number.
      – The *label* of the child node consists of the two links that correspond to the zeroes in the 2×2 opportunity matrix. (For example, if one zero is in the row for city 3 and the column for city 1, and the other zero is in the row for city 4 and the column for city 5, then the label of the child node is "$3 \to 1$, $4 \to 5$.")
      – The *bound* of the child node is the L-value of the 2×2 opportunity matrix.
      – The child node has no *incoming matrix*.
   (c) The child node is a terminal node that represents a solution (i.e., a complete circuit) given by the links specified by the labels of the nodes along the path from the root node to the child node, and the length of this circuit is equal to the bound of the child node.
   (d) <u>*Check*</u> that the length of this circuit equals the bound of the child node by adding up the distances from the original distance matrix. (If not, something is wrong. Check the arithmetic in your work.)
   (e) You are done processing this node. Do not continue to the following steps. Go back to the top-level outline on page 1.

3. [Second special case.] Check to see whether the opportunity matrix has one or more zeroes that are the only number in their row or the only number in their column. [This can happen if the opportunity matrix contains N's from step 6(b).] If so:
   (a) Choose any such zero in the opportunity matrix. Because this zero is the only number in its row (or in its column), it represents a link that *must* be taken.
   (b) Create a single child node, and assign it the next available node number.
      – The *label* of the child node is the link that corresponds to the zero chosen in step (a). (For example, if that zero is in the row for city 2 and the column for city 3, then the label of the child node is "$2 \to 3$.")
      – The *bound* of the child node is the L-value of the opportunity matrix of the node being processed.
      – The *incoming matrix* of the child node is formed from the opportunity matrix of the node being processed by deleting the row and column of the chosen zero, and the L-value of this incoming matrix is the L-value of the opportunity matrix of the node being processed.
   (c) You are done processing this node. Do not continue to the following steps. Go back to the top-level outline on page 1.

4. Compute *regrets*. Every zero in the opportunity matrix has a corresponding regret, which is the sum of the smallest other number in that row (possibly zero) and the smallest other number in that column (possibly zero).

5. Choose the largest regret, and call it $R_{\max}$.
   (a) If there is a tie: Choose the one that corresponds to the smaller distance in the *original* distance matrix.
   (b) If there is still a tie: Choose arbitrarily.

6. Create two child nodes, and assign them the next available node numbers.
   (a) Right child:
      – The *label* of the right child is the link that corresponds to the regret $R_{\max}$. (For example, if $R_{\max}$ came from the row for city 4 and the column for city 1, then the label of the right child is "$4 \to 1$.")
      – The *bound* of the right child is the L-value of the opportunity matrix of the node being processed.
      – The *incoming matrix* of the right child is formed from the opportunity matrix of the node being processed by deleting the row and column of the regret $R_{\max}$, and the L-value of this incoming matrix equals the L-value of the opportunity matrix of the node being processed.
   (b) Left child:
      – The *label* of the left child is the "negation" of the label of the right child. (For example, if $R_{\max}$ came from the row for city 4 and the column for city 1, then the label of the left child is "$4 \not\to 1$," meaning that the link from 4 to 1 will *not* be used in the circuit. The textbook uses the notation "$\sim 4 \to 1$" instead.)
      – The *bound* of the left child is the L-value of the opportunity matrix of the node being processed, plus $R_{\max}$.
      – The *incoming matrix* of the left child is formed from the opportunity matrix of the node being processed by replacing the zero corresponding to $R_{\max}$ with an N (not an M!), and the L-value of this incoming matrix equals the L-value of the opportunity matrix of the node being processed. [The N enforces the decision not to take that link.]

   Note that the bound of the left child does *not* equal the L-value of its incoming matrix!

7. You are done processing this node. Go back to the top-level outline on page 1.