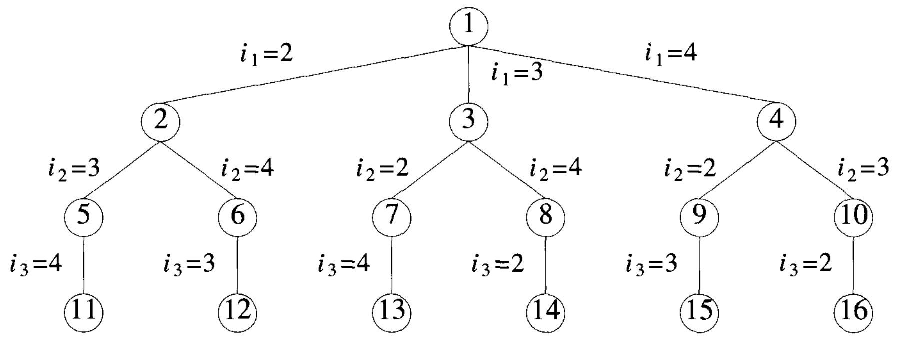


Figure 8.10 State space tree for the traveling salesperson problem with $n = 4$ and $i_0 = i_4 = 1$

A simple $\hat{c}(\cdot)$ such that $\hat{c}(A) \leq c(A)$ for all A is obtained by defining $\hat{c}(A)$ to be the length of the path defined at node A. For example, the path defined at node 6 of Figure 8.10 is $i_0, i_1, i_2 = 1, 2, 4$. It consists of the edges $\langle 1, 2\rangle$ and $\langle 2, 4\rangle$. A better $\hat{c}(\cdot)$ can be obtained by using the reduced cost matrix corresponding to G. A row (column) is said to be *reduced* iff it contains at least one zero and all remaining entries are non-negative. A matrix is *reduced* iff every row and column is reduced. As an example of how to reduce the cost matrix of a given graph G, consider the matrix of Figure 8.11(a). This corresponds to a graph with five vertices. Since every tour on this graph includes exactly one edge $\langle i, j\rangle$ with $i = k$, $1 \leq k \leq 5$, and exactly one edge $\langle i, j\rangle$ with $j = k$, $1 \leq k \leq 5$, subtracting a constant t from every entry in one column or one row of the cost matrix reduces the length of every tour by exactly t. A minimum-cost tour remains a minimum-cost tour following this subtraction operation. If t is chosen to be the minimum entry in row i (column j), then subtracting it from all entries in row i (column j) introduces a zero into row i (column j). Repeating this as often as needed, the cost matrix can be reduced. The total amount subtracted from the columns and rows is a lower bound on the length of a minimum-cost tour and can be used as the $\hat{c}$ value for the root of the state space tree. Subtracting 10, 2, 2, 3, 4, 1, and 3 from rows 1, 2, 3, 4, and 5 and columns 1 and 3 respectively of the matrix of Figure 8.11(a) yields the reduced matrix of Figure 8.11(b). The total amount subtracted is 25. Hence, all tours in the original graph have a length at least 25.

We can associate a reduced cost matrix with every node in the traveling salesperson state space tree. Let A be the reduced cost matrix for node R. Let S be a child of R such that the tree edge (R, S) corresponds to including edge $\langle i, j\rangle$ in the tour. If S is not a leaf, then the reduced cost matrix for S may be obtained as follows: (1) Change all entries in row i and column j of A to ∞. This prevents the use of any more edges leaving vertex i or entering vertex j. (2) Set $A(j, 1)$ to ∞. This prevents the use of edge $\langle j, 1\rangle$. (3) Reduce all rows and columns in the resulting matrix except for rows and columns containing only ∞. Let the resulting matrix be B. Steps (1) and (2) are valid as no tour in the subtree s can contain edges of the type $\langle i, k\rangle$ or $\langle k, j\rangle$ or $\langle j, 1\rangle$ (except for edge $\langle i, j\rangle$). If r is the total amount subtracted in step (3) then $\hat{c}(S) = \hat{c}(R) + A(i, j) + r$. For leaf nodes, $\hat{c}(\cdot) = c()$ is easily computed as each leaf defines a unique tour. For the upper bound function u, we can use $u(R) = \infty$ for all nodes R.

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix}$$

(a) Cost matrix

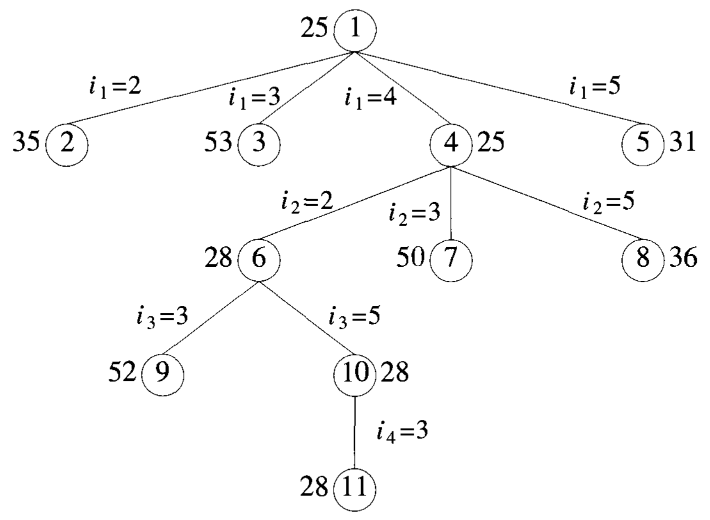
$$\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

(b) Reduced cost matrix
L = 25

Figure 8.11 An example

Let us now trace the progress of the LCBB algorithm on the problem instance of Figure 8.11(a). We use $\hat{c}$ and u as above. The initial reduced matrix is that of Figure 8.11(b) and *upper* $= \infty$. The portion of the state space tree that gets generated is shown in Figure 8.12. Starting with the root node as the E-node, nodes 2, 3, 4, and 5 are generated (in that order). The reduced matrices corresponding to these nodes are shown in Figure 8.13. The matrix of Figure 8.13(b) is obtained from that of 8.11(b) by (1) setting all entries in row 1 and column 3 to ∞, (2) setting the element at position (3, 1) to ∞, and (3) reducing column 1 by subtracting by 11. The $\hat{c}$ for node 3 is therefore 25 + 17 (the cost of edge $\langle 1, 3\rangle$ in the reduced matrix) + 11 = 53. The matrices and $\hat{c}$ value for nodes 2, 4, and 5 are obtained similarly. The value of *upper* is unchanged and node 4 becomes the next E-node. Its children 6, 7, and 8 are generated. The live nodes at this time are nodes 2, 3, 5, 6, 7, and 8. Node 6 has least $\hat{c}$ value and becomes the next E-node. Nodes 9 and 10 are generated. Node 10 is the next E-node. The solution node, node 11, is generated. The tour length for this node is $\hat{c}(11) = 28$ and *upper* is updated to 28. For the next E-node, node 5, $\hat{c}(5) = 31 > upper$. Hence, LCBB terminates with 1, 4, 2, 5, 3, 1 as the shortest length tour.

An exercise examines the implementation considerations for the LCBB algorithm. A different LCBB algorithm can be arrived at by considering



Numbers outside the node are $\hat{c}$ values

Figure 8.12 State space tree generated by procedure LCBB

a different tree organization for the solution space. This organization is reached by regarding a tour as a collection of n edges. If $G = (V, E)$ has e edges, then every tour contains exactly n of the e edges. However, for each $i, 1 \leq i \leq n$, there is exactly one edge of the form $\langle i, j \rangle$ and one of the form $\langle k, i \rangle$ in every tour. A possible organization for the state space is a binary tree in which a left branch represents the inclusion of a particular edge while the right branch represents the exclusion of that edge. Figure 8.14(b) and (c) represents the first two levels of two possible state space trees for the three vertex graph of Figure 8.14(a). As is true of all problems, many state space trees are possible for a given problem formulation. Different trees differ in the order in which decisions are made. Thus, in Figure 8.14(c) we first decide the fate of edge $\langle 1, 2 \rangle$. Rather than use a static state space tree, we now consider a dynamic state space tree (see Section 7.1). This is also a binary tree. However, the order in which edges are considered depends on the particular problem instance being solved. We compute $\hat{c}$ in the same way as we did using the earlier state space tree formulation.

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & 2 & 0 \\ 0 & \infty & \infty & 0 & 2 \\ 15 & \infty & 12 & \infty & 0 \\ 11 & \infty & 0 & 12 & \infty \end{bmatrix}$$

(a) Path 1,2; node 2

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & 2 & 0 \\ \infty & 3 & \infty & 0 & 2 \\ 4 & 3 & \infty & \infty & 0 \\ 0 & 0 & \infty & 12 & \infty \end{bmatrix}$$

(b) Path 1,3; node 3

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & \infty & 0 \\ 0 & 3 & \infty & \infty & 2 \\ \infty & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & \infty & \infty \end{bmatrix}$$

(c) Path 1,4; node 4

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & 9 & 0 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 12 & 0 & 9 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix}$$

(d) Path 1,5; node 5

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 11 & \infty & 0 \\ 0 & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & 0 & \infty & \infty \end{bmatrix}$$

(e) Path 1,4,2; node 6

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & \infty & \infty & 0 \\ \infty & 1 & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & 0 & \infty & \infty & \infty \end{bmatrix}$$

(f) Path 1,4,3; node 7

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 1 & \infty & 0 & \infty & \infty \\ 0 & 3 & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & 0 & 0 & \infty & \infty \end{bmatrix}$$

(g) Path 1,4,5; node 8

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 0 \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \end{bmatrix}$$

(h) Path 1,4,2,3; node 9

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix}$$

(i) Path 1,4,2,5; node 10

Figure 8.13 Reduced cost matrices corresponding to nodes in Figure 8.12

As an example of how LCBB would work on the dynamic binary tree formulation, consider the cost matrix of Figure 8.11(a). Since a total of 25

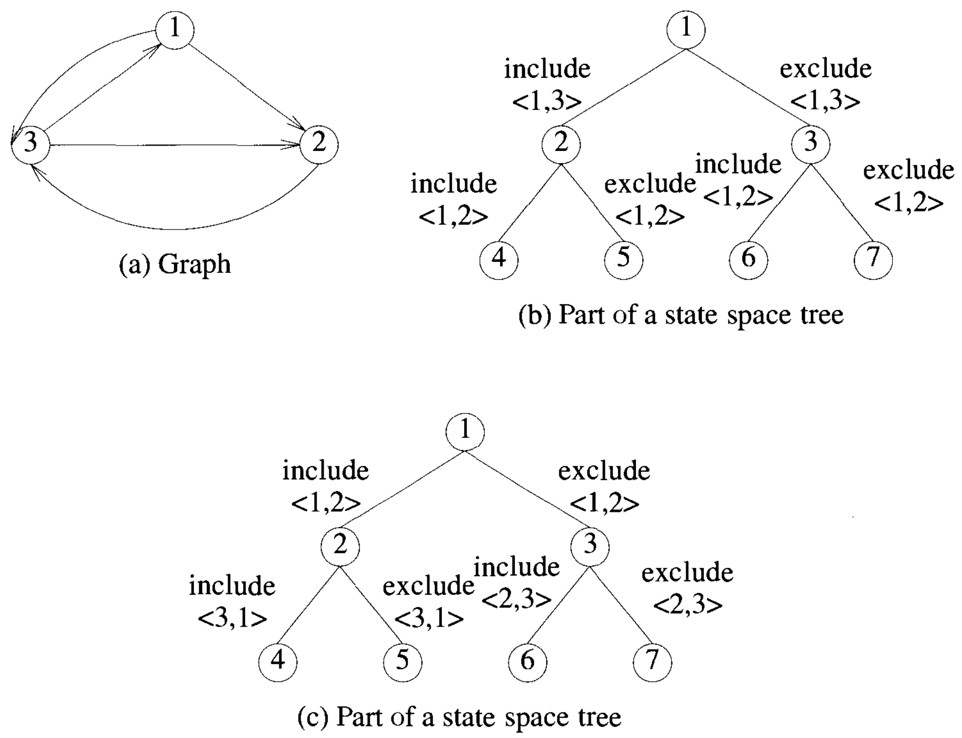


Figure 8.14 An example

needs to be subtracted form the rows and columns of this matrix to obtain the reduced matrix of Figure 8.11(b), all tours have a length at least 25. This fact is represented by the root of the state space tree of Figure 8.15. Now, we must decide which edge to use to partition the solution space into two subsets. If edge $\langle i, j \rangle$ is used, then the left subtree of the root represents all tours including edge $\langle i, j \rangle$ and the right subtree represents all tours that do not include edge $\langle i, j \rangle$. If an optimal tour is included in the left subtree, then only $n - 1$ edges remain to be selected. If all optimal tours lie in the right subtree, then we have still to select n edges. Since the left subtree selects fewer edges, it should be easier to find an optimal solution in it than to find one in the right subtree. Consequently, we would like to choose as the partitioning edge an edge $\langle i, j \rangle$ that has the highest probability of being