UNIT–VII – BRANCH AND BOUND

Branch and Bound: General method, Applications– Travelling salesperson problem, 0/1 Knapsack problem– LC Branch and Bound solution, FIFO Branch and bound solution.

-0-0-0-0-

Introduction:

The term Branch and Bound refers to all state space search methods in which all children of the E–Node are generated before any other live node can become the E–Node.

We have already seen two graph search strategies, BFS and DFS in which the exploration of a new node cannot begin until the node currently being explored is fully explored.

Both of these generalize to Branch and Bound strategies.

In Branch and Bound BFS like space search will be called FIFO search as the list of live nodes is a first in first out list (or queue).

A DFS like state space search will be called LIFO (Last In First Out) search as the list of live nodes is a last in first out list (or Stack).

Ex) let us see how a FIFO branch and bound algorithm would search the state space tree for the 4 queens problem. We have already solved the problem using backtracking as under.

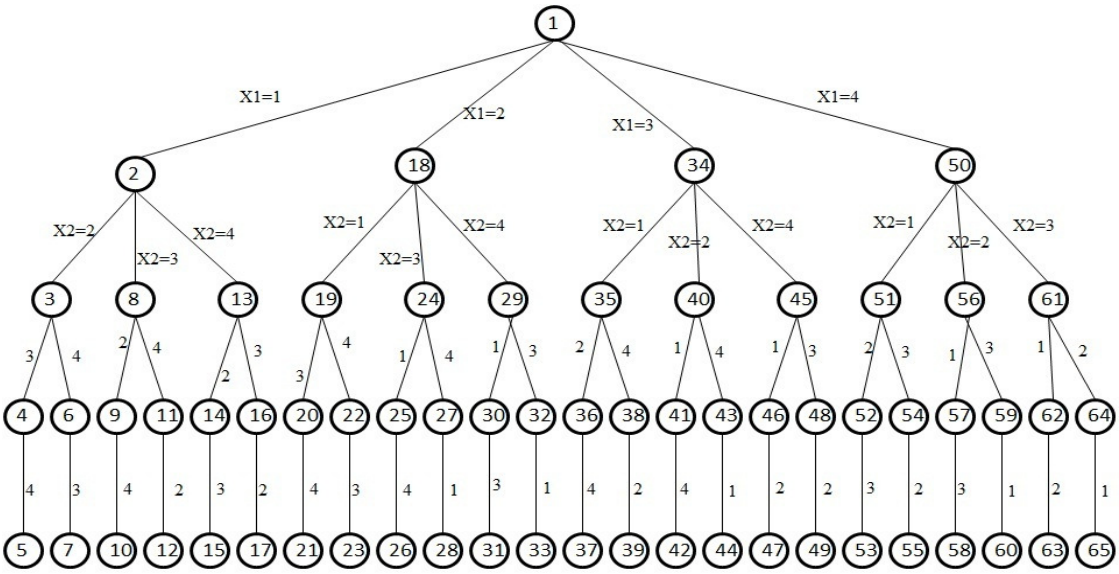


Fig 1) state space tree for 4 queens problem. Nodes are numbered in DFS

Some terminology regarding tree organizations of solution spaces.

Problem state: Each node in the tree defines a problem state.

State space: All paths from the root to other nodes define the state space of the problem.

Solution states: Solution states are those problem states s for which the path from the root to s defines a tuple in the solution space.

Answer states: Answer states are those solution states s for which the path from the root to s defines a tuple that is a member of the set of solutions.

State space Tree: The tree organization of the solution space is referred to as the state space tree.

Live Node: A node which has been generated and all of whose children have not yet been generated is called a live node.

E-Node: The live node whose children are currently being generated is called E-node(node being expanded).

Dead Node: A dead node is generated node which has not to be expanded further or all of whose children have been generated.

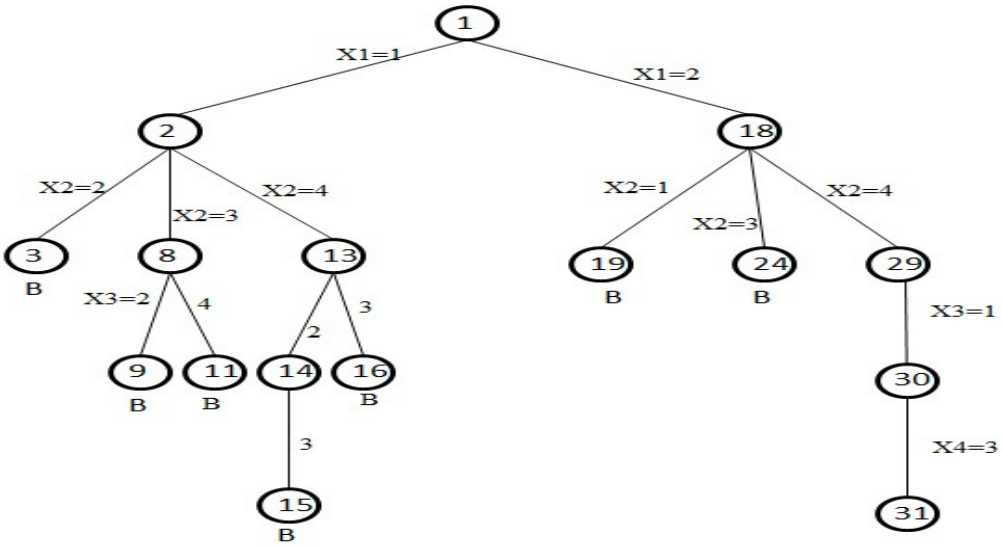


Fig 2) Portion of the tree of fig (1) that is generated during Backtracking.

To solve the above problem using branch and bound method Initially there is only one live node, i.e. node 1. This represents the case in which no queen has been placed on the chess board. This node becomes the E-node.

It is expanded and its children nodes 2,18,34 and 50 are generated. These nodes represent a chess board with queen 1 in row1 and columns 1,2 ,3 and 4 respectively. The only live nodes now are 2,18,34 and 50.

If the nodes are generated in this order then the next E-Node is node 2. It is expanded and nodes 3,8, and 13 are generated.

Node 3 is immediately killed using bounding function. Nodes 8 and 13 are added to the queue of live nodes.

Node 18 becomes next E-Node. Nodes 19,24 & 29 are generated. Nodes 19 and 24 are killed as a result of the bounding function. Node 29 is added to the queue of live nodes.

Now the E-node is 34.

Nodes that are killed as a result of the bounding functions have a "B" under them. Numbers inside the nodes correspond to the numbers in fig (1) the state space tree or permutation tree. Numbers outside the nodes give the order in which the nodes are generated by FIFO branch and bound.

At the time the answer node, i.e. node 31 is reached, the only live nodes remaining are nodes 38 and 54.

A comparison of figures 2 and 3 indicates that back tracking is a superior search method for this problem.

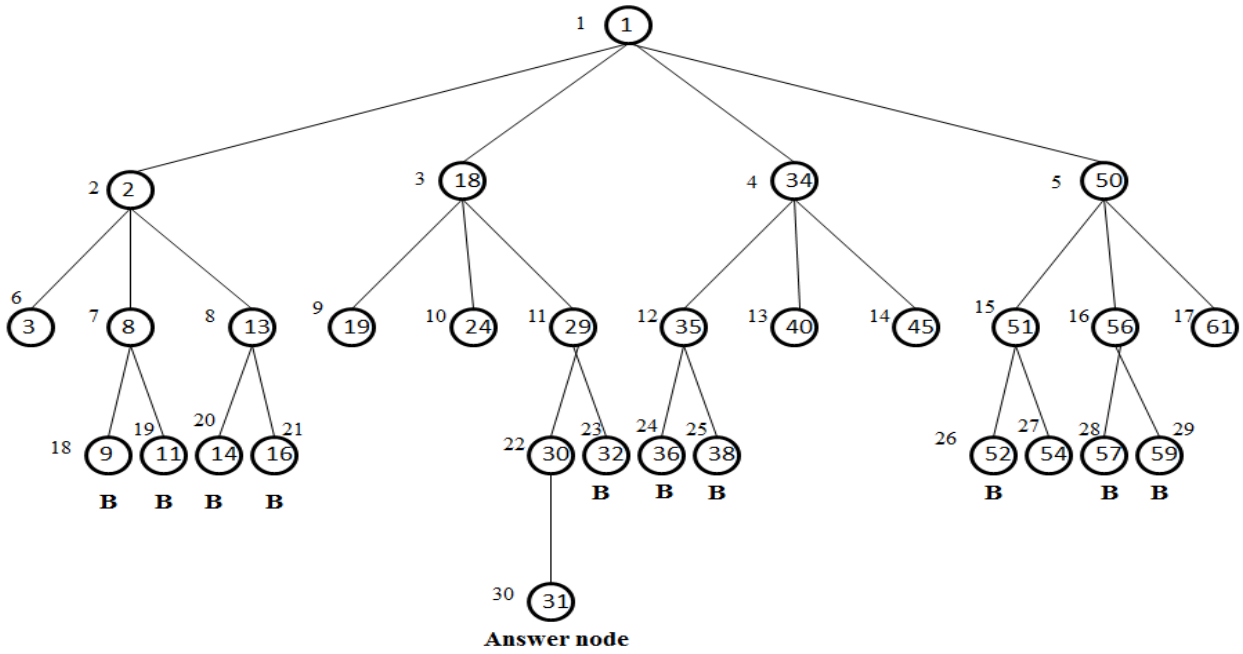


Fig 3) Portion of the tree in fig 1) that is generated by a FIFO Branch and Bound search.

General Methods in branch and Bound
FIFO branch and bound search
LIFO branch and bound search
LC (Least Cost) branch and bound search / LCBB search

FIFO branch and bound search

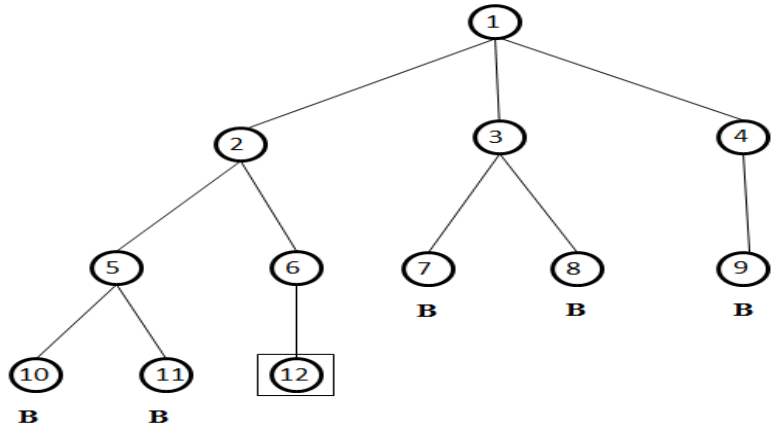


Fig 4) State space tree

Assume that the node 12 is an answer node (solution).
In FIFO search first we start with node 1. i.e. E-Node is 1.
Next we generate the children of node 1. We place all the live nodes in queue

2	3	4		

Now we delete an element from queue i.e. node 2, next and node 2 becomes E-node and generates its children and places those live nodes in queue.

3	4	5	6	

Next, delete an element from queue and considering it as E-node. i.e. E-node =3 and its children are generated. 7,8 are children of 3 and these live nodes are killed by bounding functions. So we will not include in queue.

4	5	6		

Again delete an element from Q. considering it as E-node, generate the children of node 4. Node 9 is generated and killed by bounding function.

5	6			

Again delete an element from Q. generate children of node 5, i.e. nodes 10 and 11 are generated and killed by bounding function, last node in the queue is 6.

6				

The child of node 6 is 12 and it satisfies the conditions of the problem which is the answer node, so search terminates.

LIFO branch and bound search

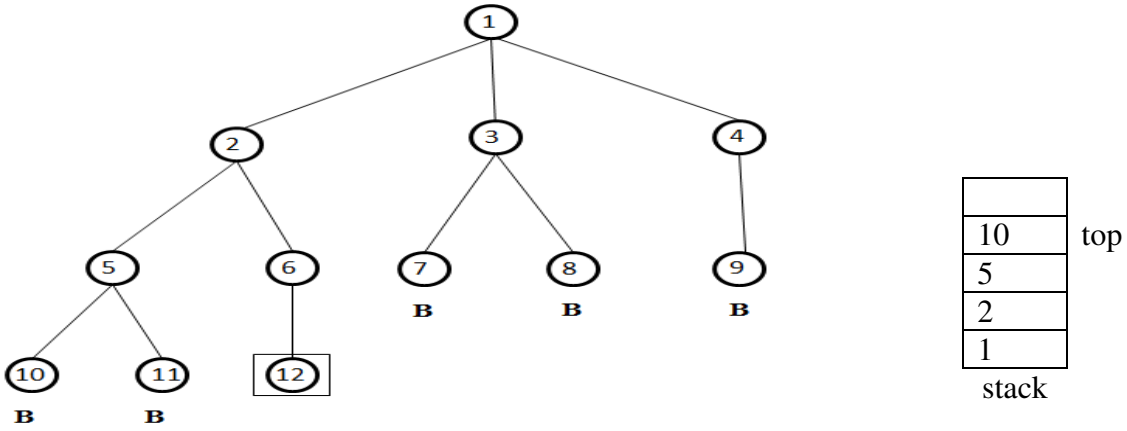


Fig 5) State space tree

For LIFO branch and bound we use the data structure stack. Initially stack is empty.
The order is 1,2,5,10,11,6,12

LC (Least Cost) branch and bound search / LCBB search

In this we use a ranking function or cost function, which is denoted by $\hat{c}(x)$. We generate the children of the E-node, among these live nodes, we select a node which has minimum cost. By using the ranking function we will calculate the cost of each node.

Note that the ranking function $\hat{c}(x)$ or cost function which depends on the problem.

Initially we take the node 1 as E-node.

Generate children of node 1, the children are 2,3,and 4. By using the ranking function we calculate the cost of 2,3 and 4 nodes as $\hat{c}(2)=2$, $\hat{c}(3)=3$, $\hat{c}(4)=4$ .Now we select a node which has minimum cost i.e. node 2. For node 2 the children are 5 and 6. $\hat{c}(5)=4$, $\hat{c}(6)=1$ we select node 6 as cost is less. Generate children of 6 i.e. 12 and 13. We select node 12 since its cost is 1 i.e. minimum. More over node 12 is the answer node. So we terminate search process.

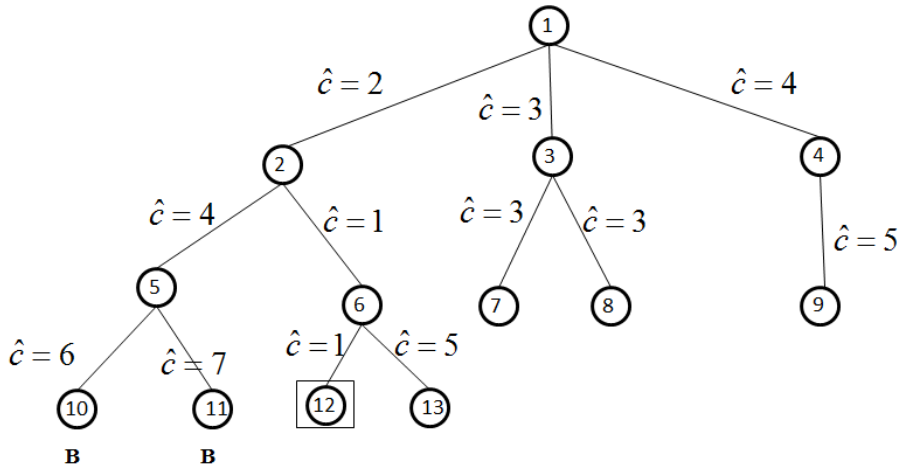


Fig 6) State space tree

Applications

Travelling salesperson problem :: LCBB Solution

0/1 Knapsack problem :: LCBB solution

0/1 Knapsack problem :: FIFOBB solution

Travelling salesperson problem

If there are n cities and cost of travelling from one city to another city is given. A sales person has to start from any one of the city and has to visit all the cities exactly once and has to return to the starting place with shortest distance or minimum cost.

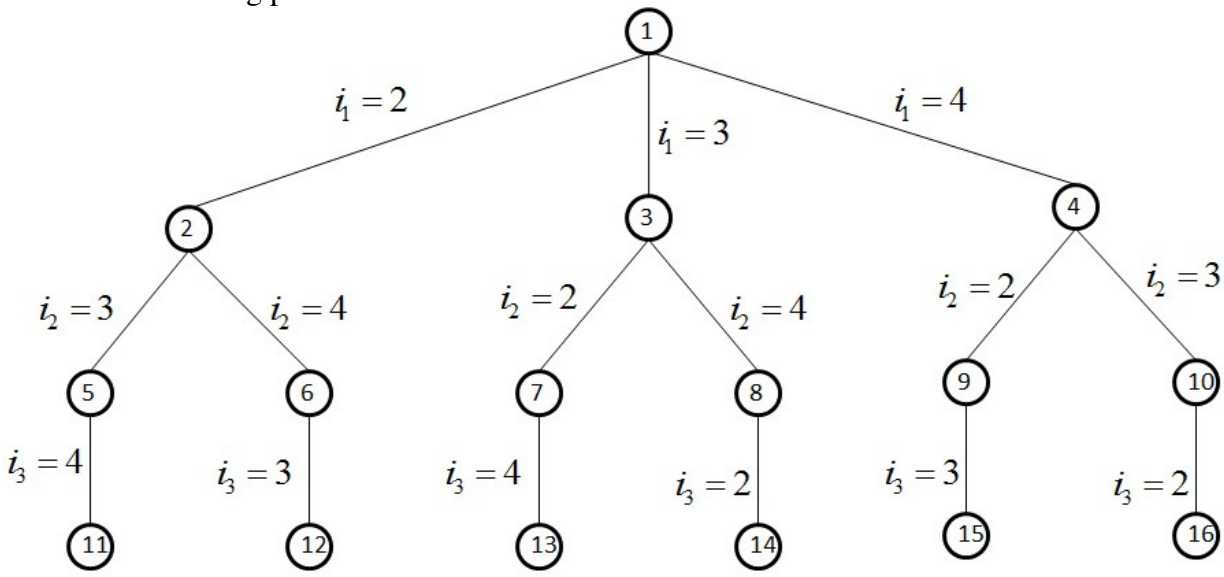


Figure: State space Tree for the travelling salesperson problem with n=4 and $i_0 = i_4 = 1$ i.e. the graph contains 4 nodes and start and end nodes are node 1.

Initially the sales person starts from node 1. Which is E-Node.
As there are 4 nodes the sales person starts at node 1 and reaches node 1 through nodes 2, 3 and 4.
From node1 there is a possibility of travelling to node 2 or node 3 or node 4. So it generates 3 live nodes namely node 2, node 3 and node 4.

Now one of nodes 2 or 3 or 4 becomes E-node depending on the cost. For each node a cost value will be calculated. Depending on the cost the next E-node is selected. The process is continued.

Let G=(V,E) be a directed graph defining an instance of the travelling sales person problem.
Let C_{ij} be the cost of the edge (i,j) and cij=∞ if (i,j)$\notin$ E(G) and let $|V| = n$.
Assume that every tour starts and ends at vertex 1.

To use least cost branch and bound to search the travelling sales person state space tree, we must define a cost function c(x) and two other functions $\hat{c}(x)$ *and* $\hat{u}(x)$ such that $\hat{c}(x) \leq c(x) \leq \hat{u}(x)$.

LCBB Travelling Salesperson problem: The cost matrix of a graph consisting of 5 vertices is given under. Each entry in the cost matrix is considered as the cost to travel from one node to another node. We need to find the least cost path from node 1 to node 1 through all other nodes such that the remaining nodes will be visited only once using Least Cost Branch and Bound Method.

Cost matrix:

$$\begin{array}{c} \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array}\begin{array}{c} \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \end{array}$$

Solution

A row is said to be reduced iff it contains at least one 0 and all remaining entries are non-negative.

A column is said to be reduced iff it contains at least one 0 and all remaining entries are non-negative.

A matrix is reduced iff every row and column is reduced.

Row Reduction :

$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \begin{matrix} 10 \\ 2 \\ 2 \\ 3 \\ 4 \end{matrix} \Rightarrow \begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 2 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix}$$

Column Reduction:

$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 2 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 2 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix}$$

$$\begin{matrix} 1 & - & 3 & - & - \end{matrix}$$

Total amount subtracted = r + c = 21 + 4 = 25

$$\hat{c}(1) = 25$$

25 will be the minimum cost to travel from node 1 as source and destination and nodes 2,3,4,5 as intermediate nodes. So node 1 is root node and it is E-node. As there is a possibility of selecting path in either of nodes 2 or 3 or 4 or 5 it generates node 2, node 3 , node 4 and node 5 as live nodes.

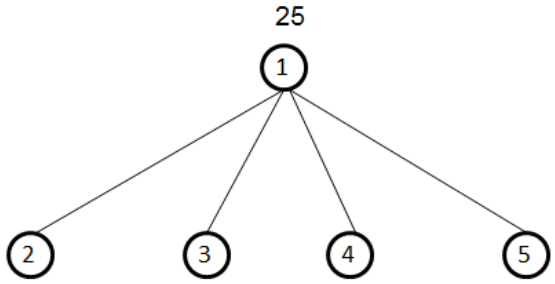


Figure ( ) Part of a state space tree

Now find minimum cost of nodes 2,3,4 and 5.i.e. $\hat{c}(2),\ \hat{c}(3),\ \hat{c}(4)\ and\ \hat{c}(5)$

Consider the path (1,2) node 2: The reduced cost matrix may be obtained as follows.
1) Change all entries in row i and column j of A to ∞. This prevents any more edges leaving vertex i or entering vertex j.
2) Set A(j,1) to ∞. This prevents the use of edge <j,1>.
3) Reduce all rows and columns in the resulting matrix except for rows and columns containing only ∞.

Change all entries of first row and second column of reduced matrix to ∞.
Assign A[2,1]= ∞.
Reduce row and reduce column
We get the following reduced cost matrix:

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

The reduced cost matrix after applying row reduction and column reduction i.e. $\Rightarrow r=0$

$$\hat{c}(2)=\hat{c}(1)+A(1,2)+r=25+10+0=35$$

Consider the path (1,3) node 3: Change all entries of first row and third column of reduced matrix to ∞ and set A(3,1) to ∞.

$$A=\begin{bmatrix}\infty & 10 & 17 & 0 & 1\\ 12 & \infty & 11 & 2 & 0\\ 0 & 2 & \infty & 0 & 2\\ 15 & 3 & 12 & \infty & 0\\ 11 & 0 & 0 & 12 & \infty\end{bmatrix}\Rightarrow\begin{bmatrix}\infty & \infty & \infty & \infty & \infty\\ 12 & \infty & \infty & 2 & 0\\ \infty & 3 & \infty & 0 & 2\\ 15 & 3 & \infty & \infty & 0\\ 11 & 0 & \infty & 12 & \infty\end{bmatrix}\Rightarrow\begin{bmatrix}\infty & \infty & \infty & \infty & \infty\\ 1 & \infty & \infty & 2 & 0\\ \infty & 3 & \infty & 0 & 2\\ 4 & 3 & \infty & \infty & 0\\ 0 & 0 & \infty & 12 & \infty\end{bmatrix}$$

11

Applying row reduction and column reduction. $\Rightarrow r=11+0=11$

$$\hat{c}(3)=\hat{c}(1)+A(1,3)+r=25+17+11=53$$

Consider the path (1,4) node 4 : Changing all entries of first row and fourth column to ∞and set A(4,1) to ∞.

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

Applying row reduction and column reduction, $\Rightarrow r=0$

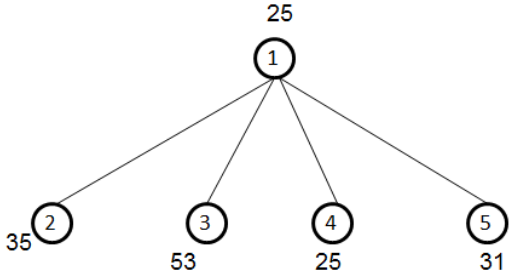
$$\hat{c}(4) = \hat{c}(1) + A(1,4) + r = 25+0+0=25$$

Consider the path (1,5) node 5: Changing all entries of first row and fifth column to ∞ and set A(5,1) to ∞. Reduce row and columns.

$$A=\begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 2 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & 2 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 15 & 3 & 12 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix} \begin{matrix} 2 \\ \\ \\ 3 \\ \\ 5 \end{matrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ 10 & \infty & 9 & 0 & \infty \\ 0 & 3 & \infty & 0 & \infty \\ 12 & 0 & 9 & \infty & \infty \\ \infty & 0 & 0 & 12 & \infty \end{bmatrix}$$

Applying row reduction and column reduction, $\Rightarrow r=5$

$$\hat{c}(5) = \hat{c}(1) + A(1,5) + r = 25+1+5=31$$



$$\hat{c}(2)=35,\ \hat{c}(3)=53,\ \hat{c}(4)=25 \text{ and } \hat{c}(5)=31$$

Since the minimum cost is 25, select node 4.

The matrix obtained for path (1,4) is considered as reduced cost matrix.

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

Now node 4 becomes E-node. Its children 6,7, and 8 are generated. The cost values are calculated for nodes 6,7 and 8.

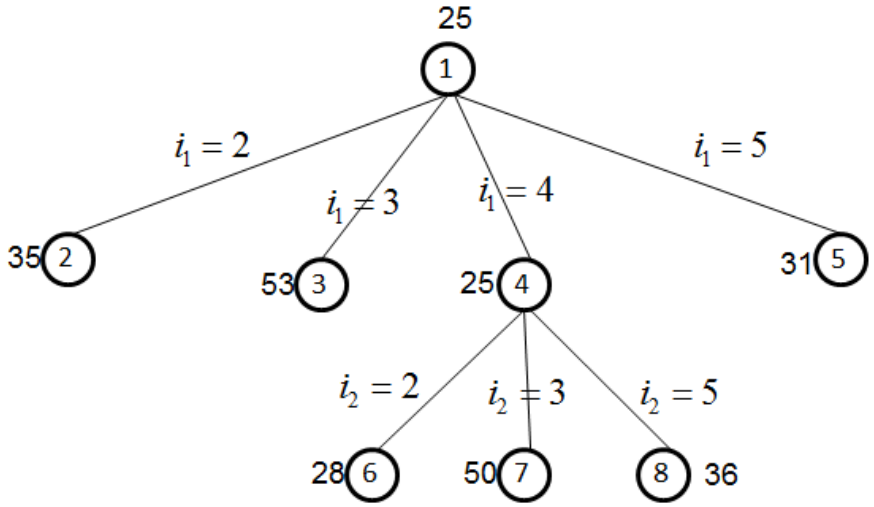


Figure ( ) Part of a state space tree

Consider the path(1,4,2) node 6: Change all entries of fourth row and second column of reduced matrix A to ∞ and set A(2,1) to ∞.

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

Apply row reduction and column reduction $\Rightarrow r=0$

$$\hat{c}(2)=\hat{c}(4)+A(4,2)+r=25+3+0=28$$

Consider the path(1,4,3) node 7: Change all entries of fourth row and third column of reduced matrix A to ∞ and set A(3,1) to ∞

$$A=\begin{bmatrix}\infty&\infty&\infty&\infty&\infty\\12&\infty&11&\infty&0\\0&3&\infty&\infty&2\\\infty&3&12&\infty&0\\11&0&0&\infty&\infty\end{bmatrix}\Rightarrow\begin{bmatrix}\infty&\infty&\infty&\infty&\infty\\12&\infty&\infty&\infty&0\\\infty&3&\infty&\infty&2\\\infty&\infty&\infty&\infty&\infty\\11&\infty&0&\infty&\infty\end{bmatrix}\begin{matrix}\\\\2\\\\\\11\end{matrix}\Rightarrow\begin{bmatrix}\infty&\infty&\infty&\infty&\infty\\1&\infty&\infty&\infty&0\\\infty&1&\infty&\infty&0\\\infty&\infty&\infty&\infty&\infty\\0&0&\infty&\infty&\infty\end{bmatrix}$$

Apply row reduction and column reduction $\Rightarrow r=2+11=13$

$$\hat{c}(3)=\hat{c}(4)+A(4,3)+r=25+12+13=50$$

Consider the path(1,4,5) node 8: Change all entries of fourth row and fifth column of reduced matrix A to ∞ and set A(5,1) to ∞

$$A=\begin{bmatrix}\infty&\infty&\infty&\infty&\infty\\12&\infty&11&\infty&0\\0&3&\infty&\infty&2\\\infty&3&12&\infty&0\\11&0&0&\infty&\infty\end{bmatrix}\Rightarrow\begin{bmatrix}\infty&\infty&\infty&\infty&\infty\\12&\infty&11&\infty&\infty\\0&3&\infty&\infty&\infty\\\infty&\infty&\infty&\infty&\infty\\\infty&0&0&\infty&\infty\end{bmatrix}\begin{matrix}\\\\11\\\\\\\end{matrix}\Rightarrow\begin{bmatrix}\infty&\infty&\infty&\infty&\infty\\1&\infty&0&\infty&\infty\\0&3&\infty&\infty&\infty\\\infty&\infty&\infty&\infty&\infty\\\infty&0&0&\infty&\infty\end{bmatrix}$$

Apply row reduction and column reduction $\Rightarrow r=11+0=11$

$$\hat{c}(5)=\hat{c}(4)+A(4,5)+r=25+0+11=36$$

Since the minimum cost is 28, select node 2.

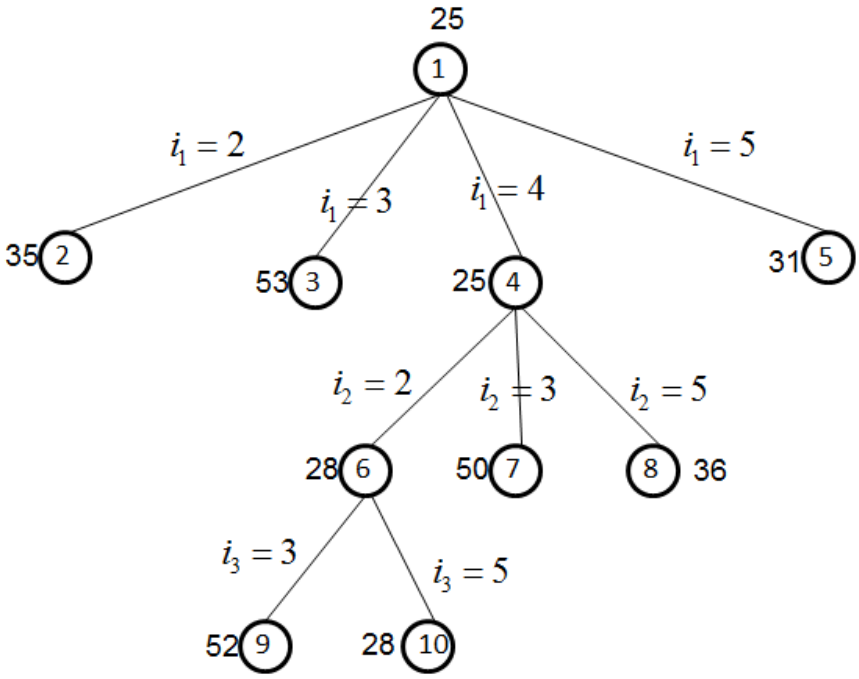


Figure ( ) Part of the state space tree

Consider the path(1,4,2,3) node 9: Change all entries of second row and third column to ∞ and set A(3,1) to ∞

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

11

Apply row reduction and column reduction $\Rightarrow r=2+11=13$

$$\hat{c}(3) = \hat{c}(2) + A(2,3) + r = 28+11+13=52$$

Consider the path (1,4,2,5) node 10: Change all entries of second row and fifth column to ∞ and set A(5,1) to ∞

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

Apply row reduction and column reduction $\Rightarrow r=0$

$$\hat{c}(3) = \hat{c}(5) + A(5,3) + r = 28+0+0=28$$

Since minimum cost is 28, select node 5.

The matrix obtained for path (2,5) is considered as reduced cost matrix.

Consider the path (1,4,2,5,3) node 11: Change all entries of fifth row and third column to ∞ and set A(3,1) to ∞

$$A = \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ 0 & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & 0 & \infty & \infty \end{bmatrix} \Rightarrow \begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \end{bmatrix}$$

Apply row reduction and column reduction $\Rightarrow r = 0$

$$\hat{c}(3) = \hat{c}(5) + A(5,3) + r = 28 + 0 + 0 = 28$$

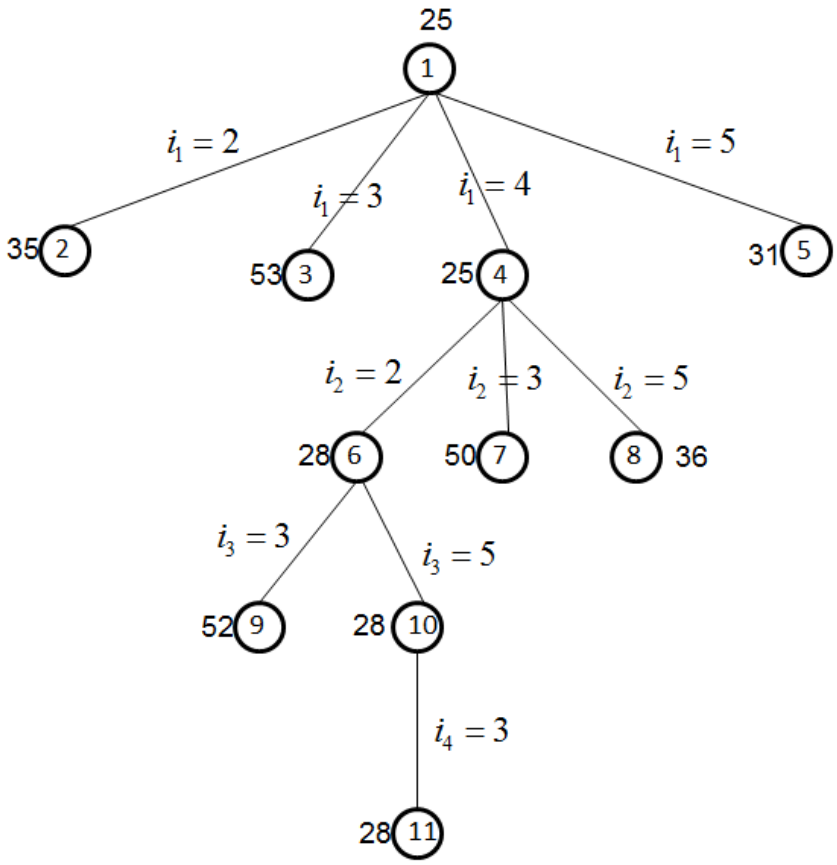


Figure ( ) Final State Space Tree

The path is 1-4-2-5-3-1
Minimum cost=10+6+2+7+3=28

LCBB terminates with 1,4,2,5,3,1 as the shortest length tour.

0/1 Knapsack problem::LCBB solution

The 0/1 knapsack problem states that, there are n objects given and capacity of knapsack is m. Then select some objects to fill the knapsack in such a way that it should not exceed the capacity of knapsack and maximum profit can be earned.

The 0/1 knapsack problem can be stated as

$Max \;\; z = p_1x_1 + p_2x_2 + ... + p_nx_n$

$w_1x_1 + w_2x_2 + ... + w_nx_n \leq m$

X_i=0 or 1.

A branch and bound technique is used to find solution to the knapsack problem. But we cannot directly apply the branch and bound technique to the knapsack problem. Because the branch and bound deals with only minimization problems. We modify the knapsack problem to the minimization problem. The modified problem is

$Min \;\; z = -p_1x_1 - p_2x_2 - ..... - p_nx_n$

$w_1x_1 + w_2x_2 + ... + w_nx_n \leq m$

X_i=0 or 1.

Let $\hat{c}(x)$ and $\hat{u}(x)$ are the two cost functions such that $\hat{c}(x) \leq c(x) \leq \hat{u}(x)$ satisfying the requirements where $c(x) = -\sum_{i=1}^{n} p_i x_i$. The c(x) is the cost function for answer node x, which lies between two functions called lower and upper bounds for the cost function c(x). The search begins at the root node. Initially we compute the lower and upper bound at root node called $\hat{c}(1)$ and $\hat{u}(1)$. Consider the first variable x1 to take a decision. The x1 takes values either 0 or 1. Compute the lower and upper bounds in each case of variable. These are the nodes at the first level. Select the node whose cost is minimum i.e.

$c(x) = \min\{c(lchild(x)), c(rchild(x))\}$

$c(x) = \min\{\hat{c}(2)), \hat{c}(3))\}$

The problem can be solved by making a sequence of decisions on the variables $x_1, x_2, \ldots, x_n$ level wise. A decision on the variable x_i involves determining which of the values 0 or 1 is to be assigned, to it by defining c(x) recursively.

The path from root to the leaf node whose height is maximum is selected and is the solution space for the knapsack problem.

Problem 1) LCBB. Consider the knapsack instance n=4, (p1,p2,p3,p4)=(10,10,12,18), (W1,W2,W3,W4)=(2,4,6,9), m=15. Let us trace the working of an LCBB search.

Solution: Let us use the fixed tuple formulation.

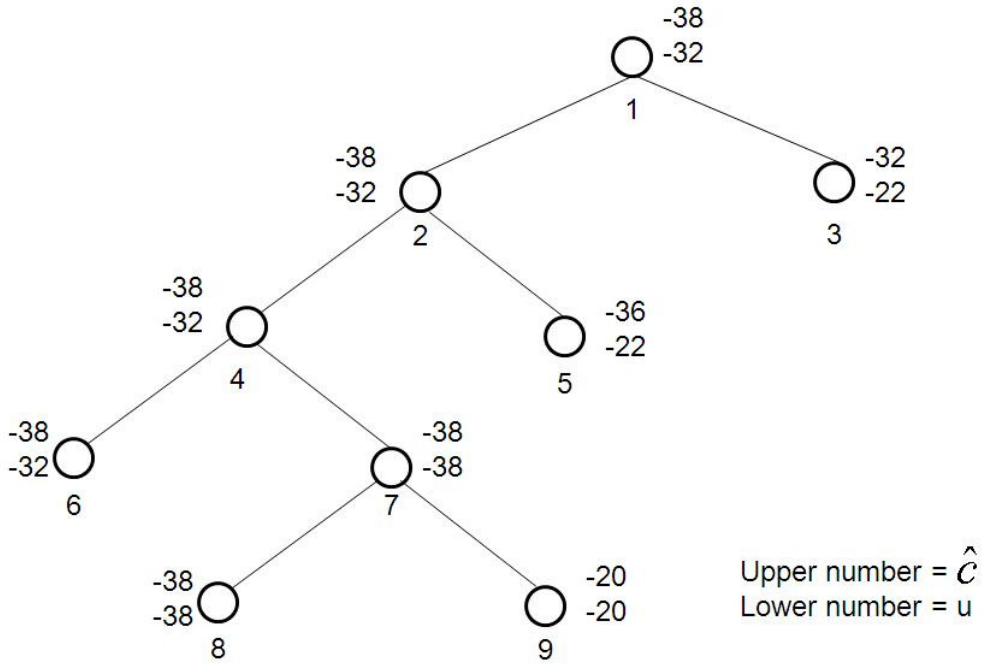
The search begins with root node as E-node.

cp - current profit
cw- current weight
k – k number of decisions
m - Capacity of knapsack

Algorithm Ubound(cp,cw,k,m)

```
{
   b := cp;
   c := cw;
       for i:= k+1 to n do
       {
          if  ( c + w[i] <= m ) then
           {
                c := c + w[i];
                b := b - p[i];
           }
       }
  return b;
}
```

Function U(.) for Knapsack problem



LC Branch and Bound tree

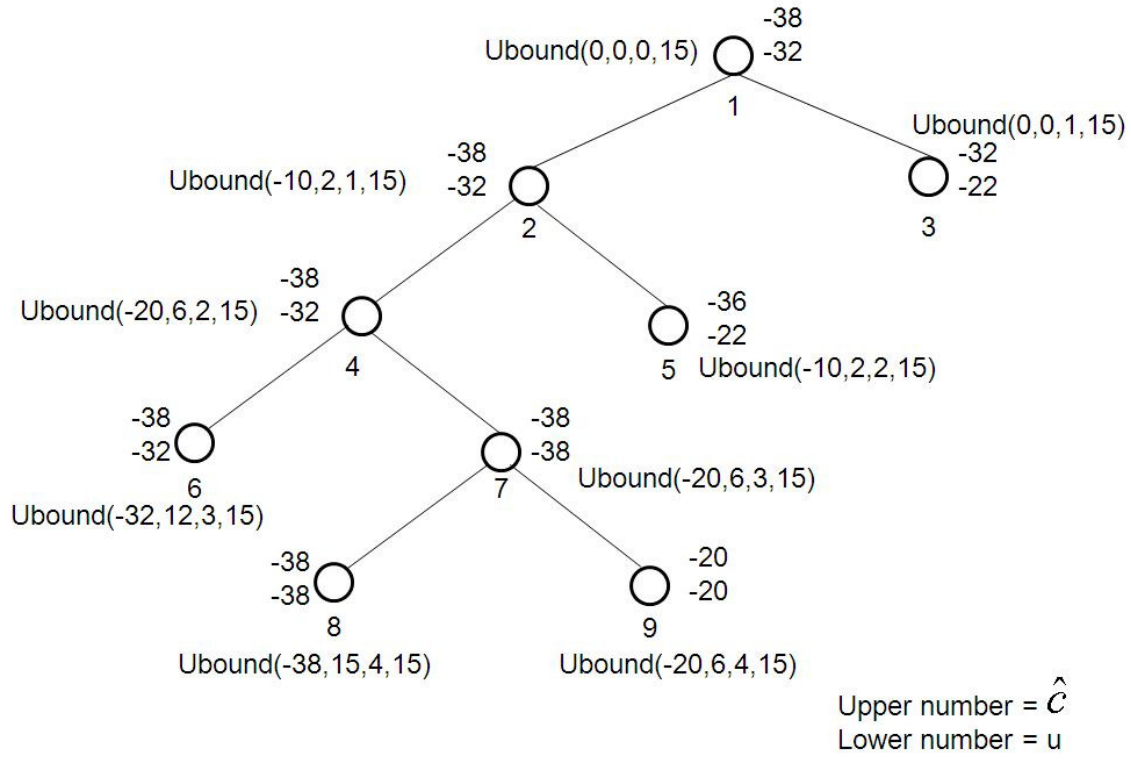


Figure shows the Upper bound function call for each node

Node 1 Ubound(0,0,0,15)

i=1 i=2 i=3 i=4

c=2 c=6 c=12

b=-10 b=-20 b=-32

U(1)=-32

$\hat{c}(1)$ = -32+3/9 x 18= -38

To calculate lower bound we allow fractions

Node2 Ubound(-10,2,1,15) x1=1

i=2 i=3 i=4

c=6 c=12

b= -20 b= -32

U(2)=-32

$\hat{c}(2)$ = -32+3/9 x 18= -38

To calculate lower bound we allow fractions

Node 3 Ubound(0,0,1,15)

i=2 i = 3 i =4

c=4 c = 10

b= -10 b = -22

U(3)=-22

$\hat{c}(3)$ = -22+5/9 x 18= -32

To calculate lower bound we allow fractions

Node 4 Ubound(-20,6,2,15)

i=3 i=4

c=12

b=-32

U(4)=-32

$\hat{c}(4)$ = -32+3/9 x 18= -38

To calculate lower bound we allow fractions

Node 5 Ubound(-10,2,2,15)
i=3 i=4
c=8
b=-22
U(5)=-22
$\hat{c}(5) =$ -22+7/9 x 18= -36
To calculate lower bound we allow fractions

Node 6 Ubound(-32,12,3,15)
i=4

U(6)=-32
$\hat{c}(6) =$ -32+3/9 x 18= -38
To calculate lower bound we allow fractions

Node 7 Ubound(-20,6,3,15)
i=4
c=15
b=-38
U(7)=-38
$\hat{c}(7) =$ -38

Node 8 Ubound(-38,15,4,15)

U(8)=-38
$\hat{c}(8) =$ -38

Node 9 Ubound(-20,6,4,15)

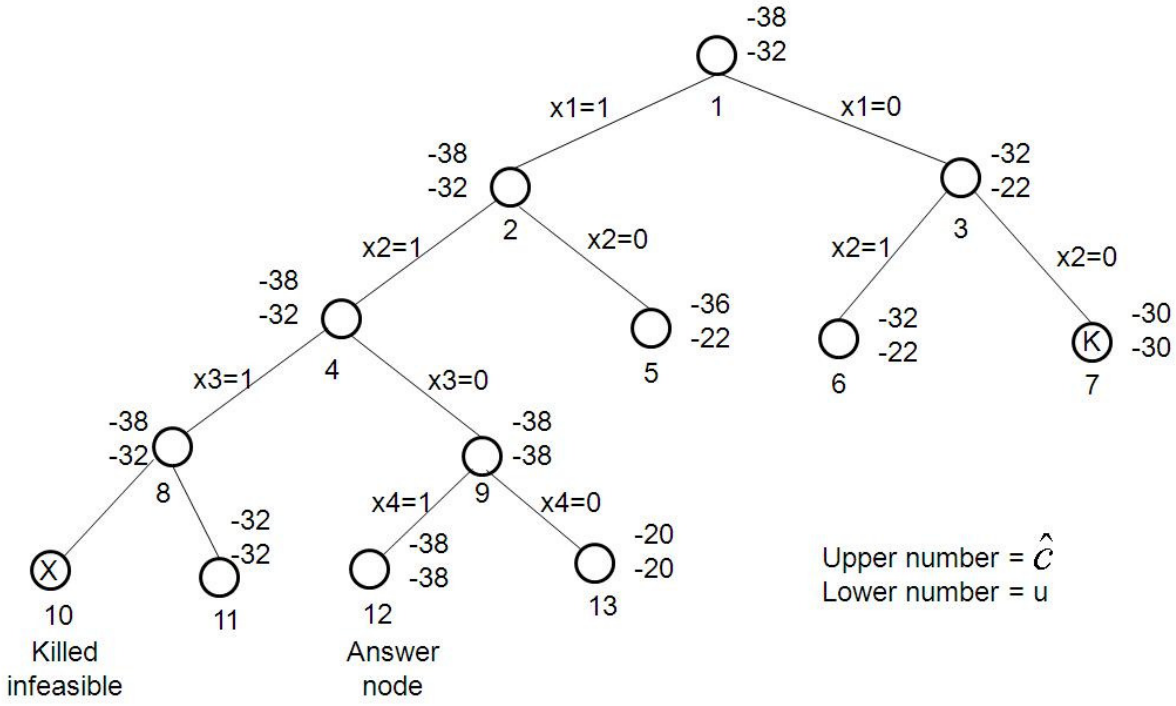
U(1)=-20
$\hat{c}(1) =$ -20

Node 8 is a solution node (solution vector)
X1=1
X2=1
X3=0
X4=1

p1x1+p2x2+p3x3+p4x4
10x1+10=1+12x0+18x1
=38
We need to consider all n items.
The tuple size is n

0/1 Knapsack problem using FIFO Branch and Bound Solution

Consider the knapsack instance n=4, (p1,p2,p3,p4) = (10,10,12,18),
(W1,W2,W3,W4) = (2,4,6,9), m=15. Let us trace the working of an FIFOBB search.



FIFO Branch and Bound Tree-Knapsack solution

Now let us trace through the FIFOBB algorithm using the same knapsack instance. The problem is to find most valuable subset of the items that fit in the knapsack.

Initially the root node i.e. node 1 be the E-Node. The Queue of Live nodes is empty.
Since this is not the solution Upper is initialized to u(1) = -32.

We assume the children are generated from left to right. Nodes 2 and 3 are generated and are added to the queue.

2	3				

Node 2 becomes the next E Node. Its children nodes are generated. Node 4 and 5 and are added to Queue.

3	4	5			

Node 3 the next e-Node, is expanded. Its Children nodes are generated. Node 6 and 7 gets added to Queue.

4	5	6			

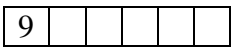
Node 7 is immediately killed as $\hat{c}(7) >$ Upper.

Node 4 is expanded next. Nodes 8 and 9 are generated and are added to queue.

5	6	8	9		

Then upper is updated to U(9)= -38.
Nodes 5 and 6 are the next two nodes to become E-Nodes. Neither is expanded as for each $\hat{c}() >$ upper.

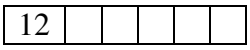
Node 8 is the next E-Node. Nodes 10 and 11 are generated.



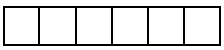
Node 10 is infeasible and so killed.

Node 11 has $\hat{c}() >$ upper and so is also killed.

Node 9 is expanded Next. When node 12 is generated, upper and answer are updated to -38 and 12 respectively. Node 12 joins the queue of live nodes.



Node 13 is killed before it can get onto the queue of live nodes as $\hat{c}(13) >$ upper.
The only remaining live node is node 12. It has no children and the search terminates.



The value of upper and the path from node 12 to the root is the output additional information is needed to determine the x_i values on this path.

The solution is : x1=1, x2=1, x3=0,x4=1

Problem 2) Solve 0/1 Knapsack problem using LCBB method

n=5, (p1,p2,p3,p4,p5)=(10,15,6,8,4), (w1,w2,w3,w4,w5)=(4,6,3,4,2) and m=12

Solution:

Convert the profits to –ve. , (p1,p2,p3,p4,p5)=(-10,-15,-6,-8,-4).
Calculate the lower bound and upper bound for each node.
Place the first item in knapsack. i.e. weight =4. Remaining weight is 12-4=8
Place second item i.e. with weight 6. Remaining weight = 8-6=2.
Since fractions are not allowed in calculation of upper bound, so we cannot place the third and fourth items. place fifth item .

Profit earned=-10-15-4=-29=upper bound

To calculate lower bound, place third item in bag considering fractions are allowed

$$\therefore Lower\ bound = -10-15-\frac{2}{3}\times 6 = -29$$

$$\therefore\ \hat{u}(1) = -29,\quad \hat{c}(1) = -29.$$

For node 2, x1=1 means, we should place first item in the bag.

$$\hat{c}(2) = -10-15-\frac{2}{3}\times 6 = -29$$

$$\hat{u}(2) = -10-15-4 = -29$$

For node 3, x1=0

$$\hat{c}(3) = -15-6-\frac{3}{4}\times 8 = -15-6-6 = -27$$

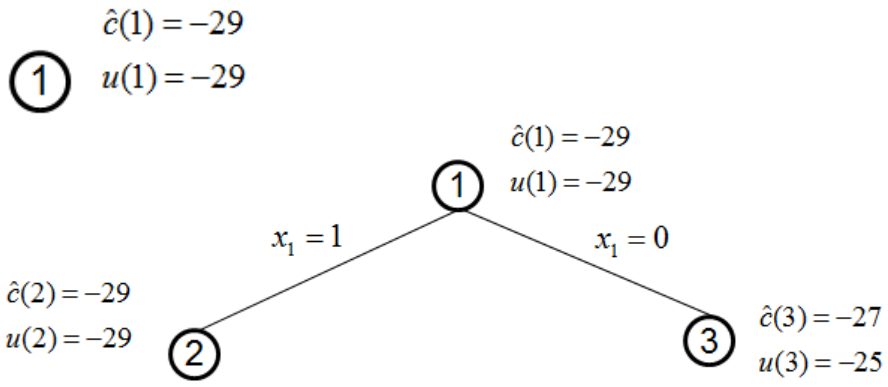
$$\hat{u}(3) = -15-6-4 = -25$$

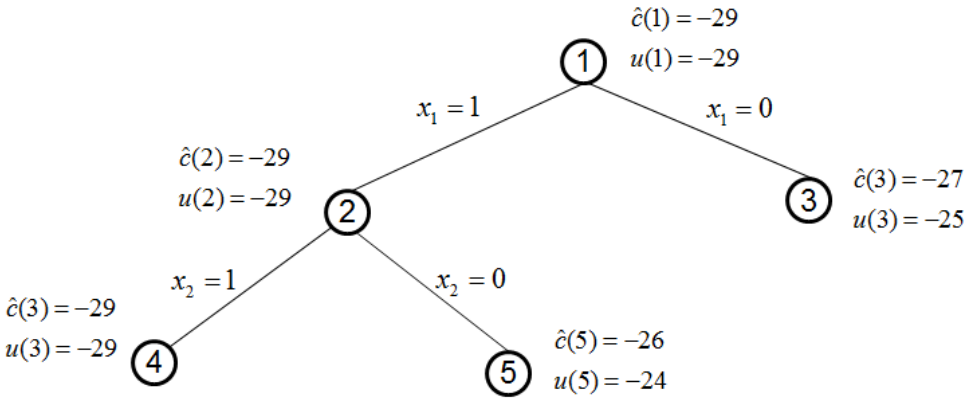
Select the minimum of lower bounds i.e

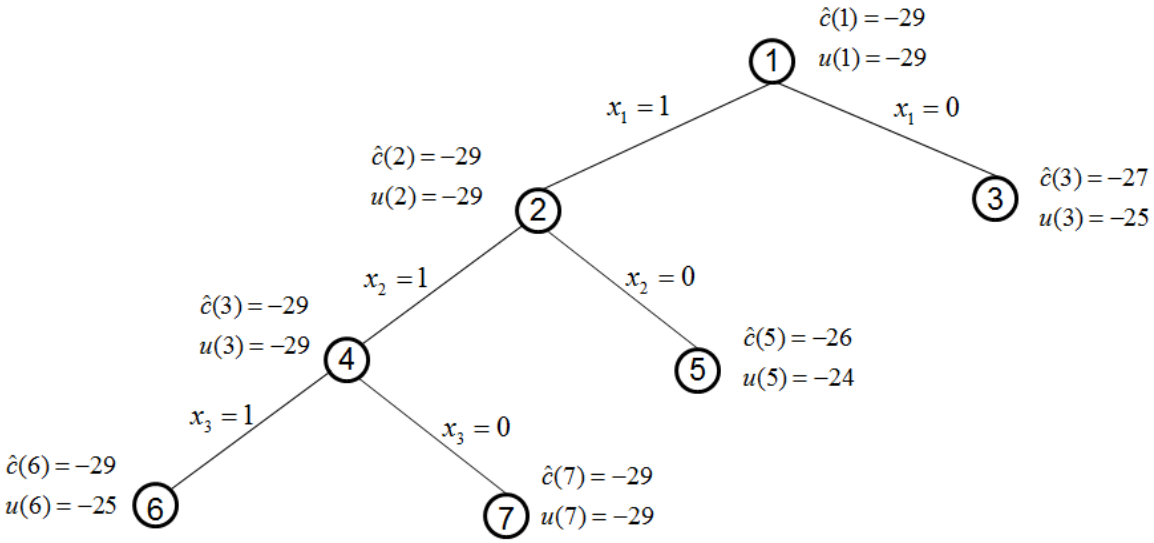
$$c(x) = \min\{c(lchild(x)), c(rchild(x))\}$$

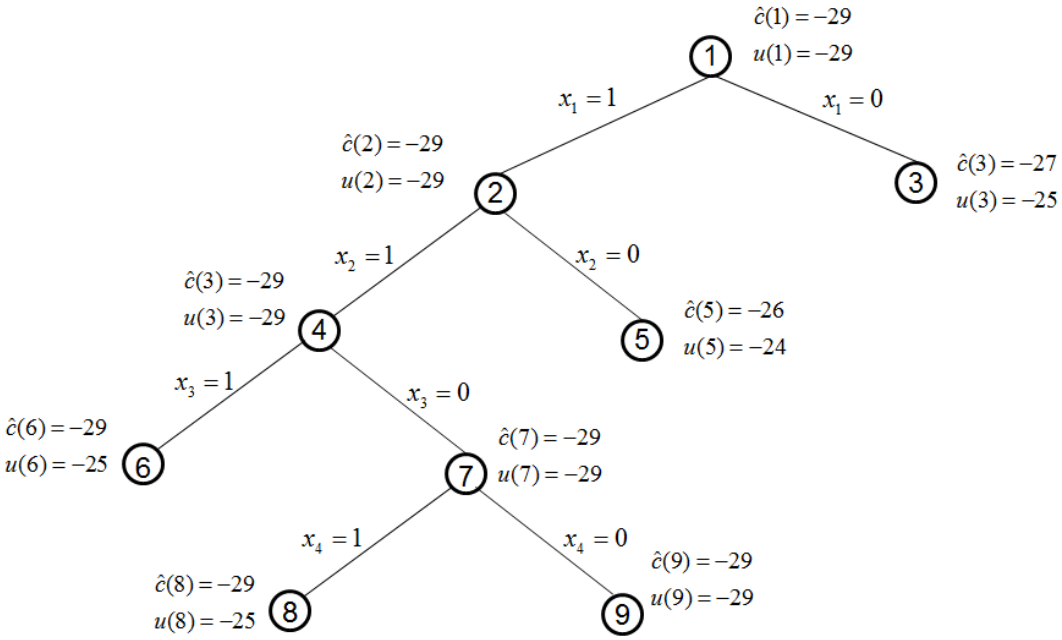
$$c(x) = \min\{\hat{c}(2)), \hat{c}(3))\} = \min\{-29,-27\} = -29 = \hat{c}(2)$$

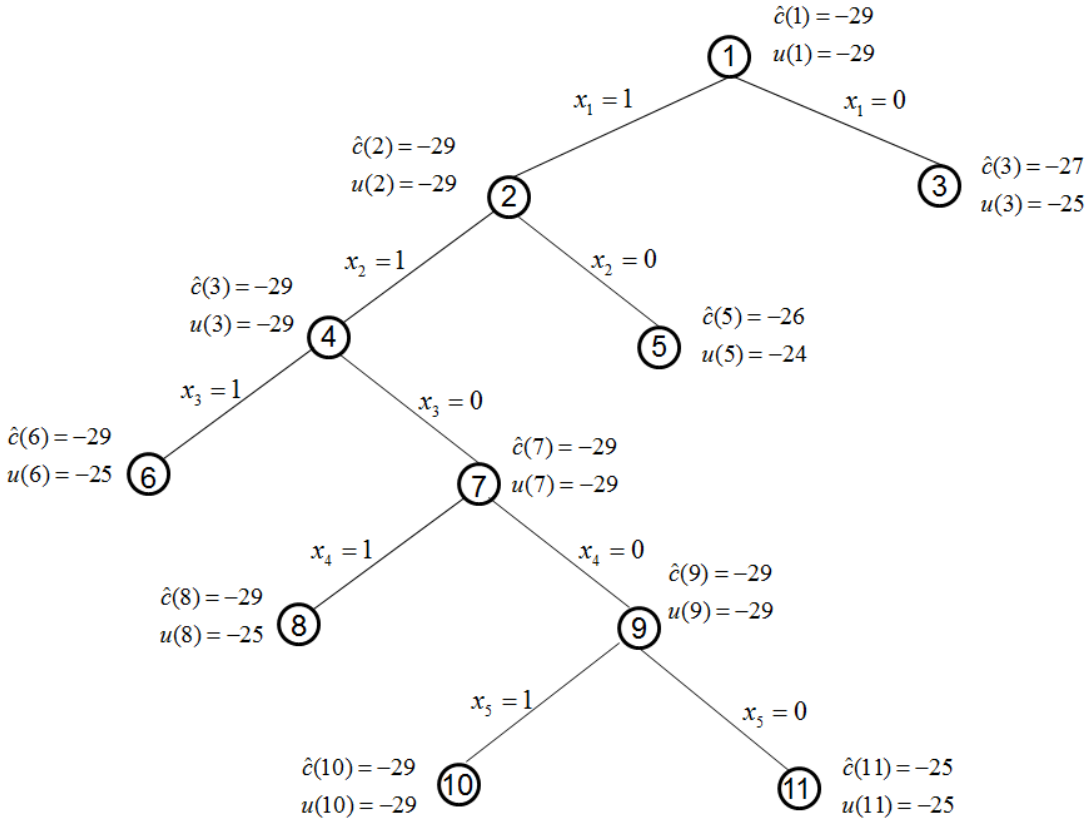
Chose node 2.




ĉ(1) = −29
u(1) = −29
1
x1 = 1
x1 = 0
ĉ(2) = −29
u(2) = −29
2
3
ĉ(3) = −27
u(3) = −25
x2 = 1
x2 = 0
ĉ(3) = −29
u(3) = −29
4
5
ĉ(5) = −26
u(5) = −24


ĉ(1) = −29
u(1) = −29
1
x1 = 1
x1 = 0
ĉ(2) = −29
u(2) = −29
2
3
ĉ(3) = −27
u(3) = −25
x2 = 1
x2 = 0
ĉ(3) = −29
u(3) = −29
4
5
ĉ(5) = −26
u(5) = −24
x3 = 1
x3 = 0
ĉ(6) = −29
u(6) = −25
6
7
ĉ(7) = −29
u(7) = −29


ĉ(1) = −29
u(1) = −29
1
x1 = 1
x1 = 0
ĉ(2) = −29
u(2) = −29
2
3
ĉ(3) = −27
u(3) = −25
x2 = 1
x2 = 0
ĉ(3) = −29
u(3) = −29
4
5
ĉ(5) = −26
u(5) = −24
x3 = 1
x3 = 0
ĉ(6) = −29
u(6) = −25
6
7
ĉ(7) = −29
u(7) = −29
x4 = 1
x4 = 0
ĉ(8) = −29
u(8) = −25
8
9
ĉ(9) = −29
u(9) = −29



LCBB Tree

The path is 1-2-4-7-9-10
The solution for knapsack problem is (x1,x2,x3,x4,x5)=(1,1,0,0,1)
Maximum profit = 10+15+4=29.

Important Questions

1)Explain the general method of Branch and Bound.

2)Explain the properties of LC-search

3)Explain the method or reduction to solve TSP problem using Branch and Bound

4)solve the following instance of travelling sales person problem using LCBB.

$$\begin{bmatrix} \infty & 7 & 3 & 12 & 8 \\ 3 & \infty & 6 & 14 & 9 \\ 5 & 8 & \infty & 6 & 18 \\ 9 & 3 & 5 & \infty & 11 \\ 18 & 14 & 9 & 8 & \infty \end{bmatrix}$$

5)Draw the portion of the state space tree generated by LCBB for the knapsack instance n=5, (p1,p2,p3,p4,p5)=(10,15,6,8,4), (w1,w2,w3,w4,w5)= 4,6,3,4,2) and m=12

6)Apply branch and bound to the above 0/1 knapsack problem and explain.

7) Explain the principles of FIFO branch and Bound.

8)Draw the portion of the state space tree generated by FIFO branch and bound for the following instances. N=4, m=15, (p1,p2,p3,p4)=(10,10,12,18), (w1,w2,w3,w4)=(12,4,6,9).