Chapter-7
Branch and Bound

7.1Feasible Solution vs. Optimal Solution

DFS, BFS, hill climbing and best-first search can be used to solve some searching problem for searching a feasible solution.

However, they cannot be used to solve the optimization problems for searching an optimal solution.

7.2 The branch-and-bound strategy

This strategy can be used to solve optimization problems without an exhaustive search in the average case. There are 2 mechanisms

- A mechanism to generate branches when searching the solution space.
- A mechanism to generate a bound so that many branches can be terminated.
- The backtracking uses a depth-first search with pruning, the branch and bound algorithm uses a breadth-first search with pruning.
- Branch and bound uses a queue as an auxiliary data structure

7.2.1 Branch-and-bound strategy

It is efficient in the average case because many branches can be terminated very early. Although it is usually very efficient, a very large tree may be generated in the worst case.Many NP-hard problems can be solved by B&B efficiently in the average case; however, the worst case time complexity is still exponential.

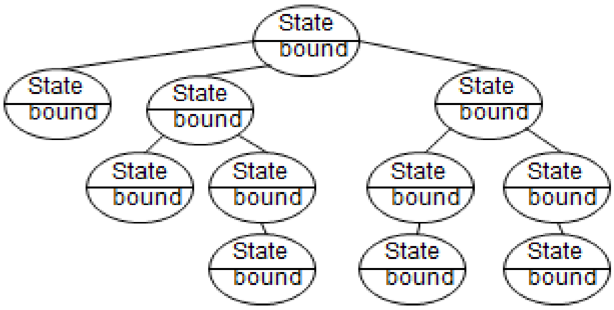
7.3The Branch and Bound Algorithm

i)Starting by considering the root node and applying a lower-bounding and upper-bounding procedure to it.

ii)If the bounds match, then an optimal solution has been found and the algorithm is finished.

iii)If they do not match, then algorithm runs on the child nodes

State-Space Tree with Bound



7.4 The 0/1 knapsack problem

The 0/1 knapsack problem

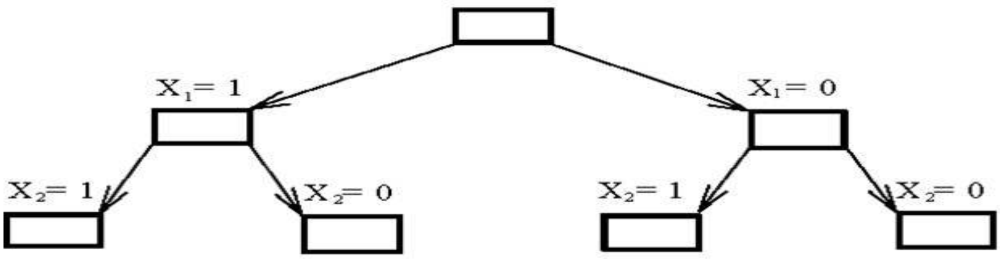


Fig. The Branching Mechanism in the Branch-and-Bound Strategy to Solve 0/1 Knapsack Problem.

How to find the upper bound?
Ans: by quickly finding a feasible solution in a greedy manner: starting from the smallest available i,

Scanning towards the largest i's until M is exceeded. The upper bound can be calculated.

The 0/1 knapsack problem with branch and bound

#items	W	v
I1	1	2
I2	2	3
I3	3	4

Given three items with knapsack capacity W=3

1) First we calculate value per weight ratio, and arrange table

#items	W	v	Vi/wi
I1	1	2	2

I2	2	3	1.5
I3	3	4	1.3

Next, start with root node, upper bound for the root node can be computed using formula as
$Ub = v + (W-w)(v_{i+1}/w_{i+1})$
Ub = 0 + (3-0) * 2 = 6 (v=0, w=0, W=3, v1/w1=2) -> root node
Next, include item 1 which is indicated by the left Branch, and exclude 1 which is indicated by right branch, shown in next slide.

W=0	V=0
Ub=6	

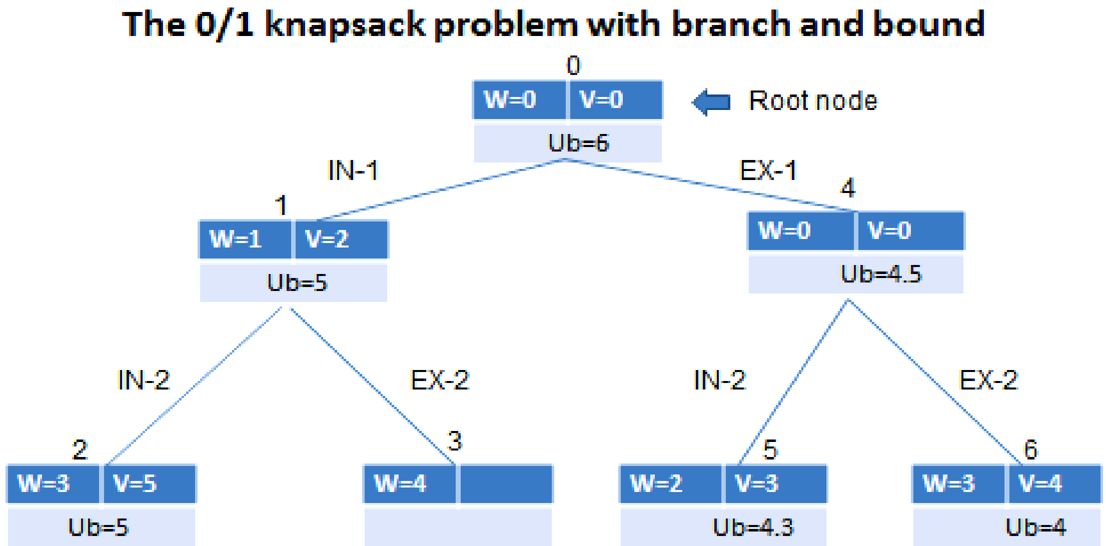


Fig. Implicit graph for knapsack problem, the optimal solution is Ub=5, of maximum value 5
calculating upper bound $Ub = v+(W-w)(v_{i+1}/w_{i+1})$

Upper bound calculation

Up=v+(W-w)(vi+1/wi+1)

Root node 0 calculated in previous slide

Node 1 : up=2+(3-1)*1.5=2+3=5

Node 2: up=5+(3-3)*1.3=5+0=5

Node 3= not required

Node 4= 0+(3-0)*1.5=4.5

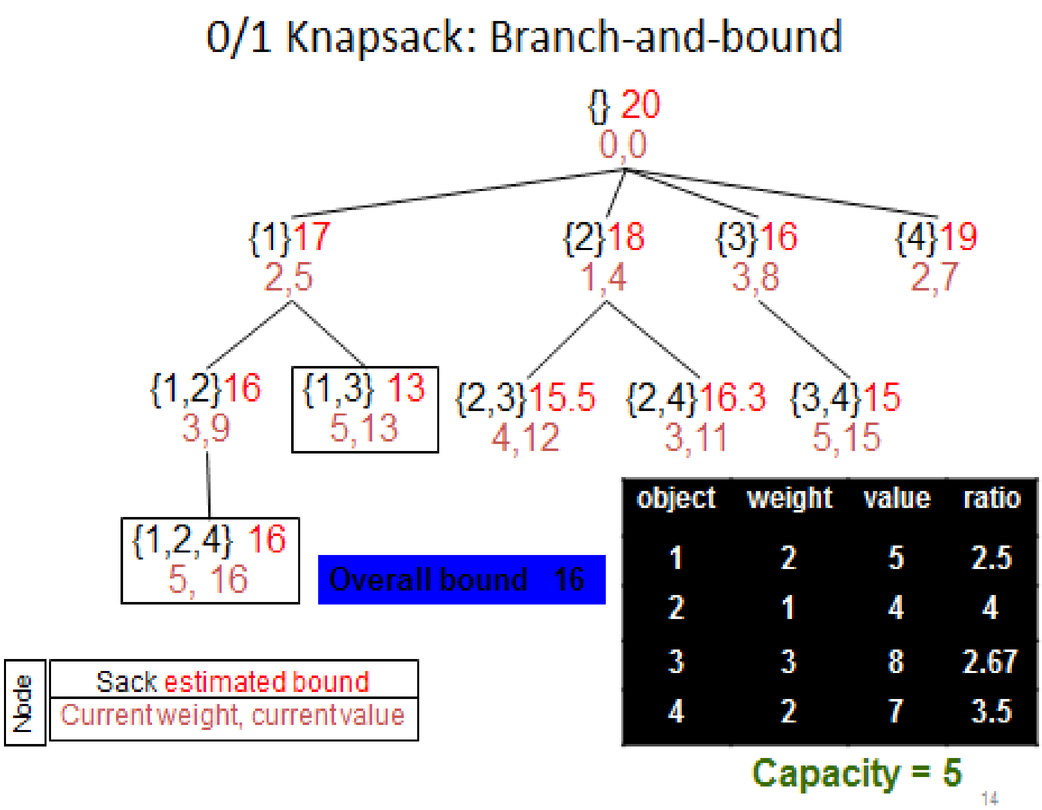
Node 5=3+(3-2)*1.3=3+1.3=4.3

Node6=4+(3-3)*1.3=3+0=4

The 0/1 knapsack problem with branch and bound

At every level we compute the upper bound, and explore the node while selecting the item. Finally the node with maximum upper bound is selected as an optimum solution. In the example in previous slide, node with item 1 and 2 gives the optimum solution i.e maximum value of 5 to the given knapsack problem.

The number above the nodes indicates the order in which the nodes are generated.



Exercise

Solve following instance of knapsack problem by using branch and bound technique:

#items	W	V	W=16
I1	9	10	
I2	6	6	
I3	7	5	
I4	3	1	

7.5 Least Cost Branch Bound

0/1 knapsack problem-1

(p1, p2, p3, p4) = (10, 10, 12, 18)

Least-cost Branch Bound solution (w1, w2, w3, w4) = (2, 4, 6, 9) M=15, n=4

Normal knapsack	0/1 knapsack
1. w=2+4+6+9x3/9=15	1. w=2+4+6=12
p=10+10+12+18x3/9=38	p=10+10+12=32
2. At x1=0	2. At x1=0
w=4+6+9x5/9=15	w=4+6=10
P=10+12+18x5/9=32	p=10+12=22
3. At x1=1, x2=0	3. At x1=1,x2=0
w=2+6+9x7/9=15	w=2+6=8
p=10+12+18x7/9=36	p=10+12=22
4. At x1=1, x2=1, x3=0	4. At x1=1, x2=1, x3=0
w=2+4+9=15	w = 2+4+9=15
p=10+10+18=38	p = 10+10+18=38
5. At x1=1, x2=1, x3=0, x4=0	5. At x1=1, x1=1, x3=0, x4=0
w=2+4=6	w = 2+4=6
p=10+10=20	p = 10+10=20

The solutions are: 1,1,01 and 1,1,0,0

But the optimal solution is at 1,1,0,1 [max profit]

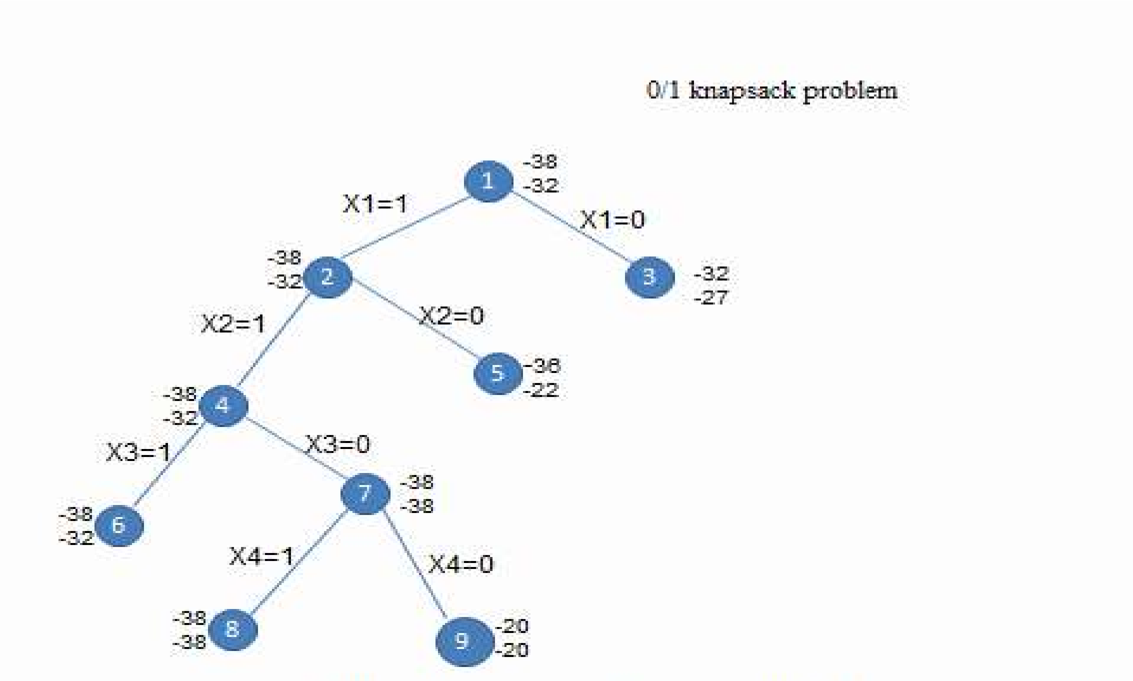


Fig.LC branch-bound tree for example 8.2 page no. 384

Example

N=5, m=15

w,1,w2,w3,w4,w5=4,4,5,8,9

P1,p2,p3,p4,p5=4,4,5,8,9

0/1 knapsackproblem-2

(p1, p2, p3, p4) = (4, 4, 5, 8, 9)

(w1, w2, w3, w4) = (4, 4, 5, 8, 9) M=15, n=5

Normal knapsack	0/1 knapsack
1. w=4+4+5+8x2/8=15 p=4+4+5+8x2/8=15	1. w=4+4+5=13 p=13
2. At x1=0 w=4+5+8X6/8=15 P=15	2. At x1=0 w=4+5=9 p=9
3. At x1=1, x2=0 w=4+5+8X6/8=15	3. at x1=1, x2=0 w=4+5=9

p=15

4. At x1=1, x2=1, x3=0
   w=4+4+8x7/8=15
   p=15

5. At x1=1, x2=1, x3=0, x4=0
   w=4+4+5+9x2/9=15
   p=15

   6.x1=1,x2=1,x3=1,x4=1,x5=0
   7. x1=1,x2=1,x3=1,x4=0,x5=0
   W=4+4+5=13
   P=13

p=9

4. At x1=1, x2=1, x3=0
   w=4+4=8
   p=8

5. At x1=1, x2=1, x3=0, x4=0
   w=4+4+5=13
   p=13

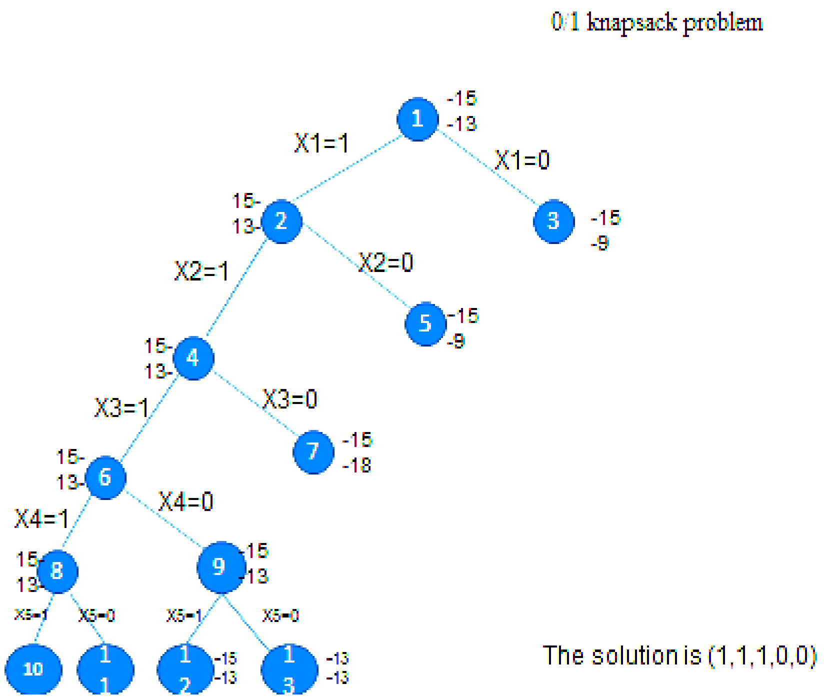
   6.x1=1,x2=1,x3=1,x4=1,x5=0
   7. w=13,p=13 x1=1,x2=1,x3=1,x4=0,x5=0

Example

N=5, m=12

(p1, p2, p3, p4, p5) = (10,1 5, 6, 8, 4)

(w1, w2, w3, w4, w5) = (4, 4, 5, 8, 9)   0/1 knapsackproblem-3



How to find the lower bound?

Ans: by relaxing our restriction from $X_i = 0$ (or) 1 to $0 \leq X_i \leq 1$ (knapsack problem)

Let $-\sum_{i=1}^{n} P_i X_i$ be an optimal solution for 0/1 knapsack problem and

$$-\sum_{i=1}^{n} P_i X_i' \text{ be an optimal solution for fractional knapsack problem.}$$

Let $Y = -\sum_{i=1}^{n} P_i X_i$, $Y' = -\sum_{i=1}^{n} P_i X_i'$.

$$\Rightarrow Y' \leq Y$$

How to expand the tree?
By the best-first search scheme

That is, by expanding the node with the best lower bound. If two nodes have the same lower bounds, expand the node with the lower upper bound.

Efficiency of Branch and Bound
In many types of problems, branch and bound is faster than branching, due to the use of a breadth-first search instead of a depth-first search

The worst case scenario is the same, as it will still visit every node in the tree

7.6 Traveling salesman problem

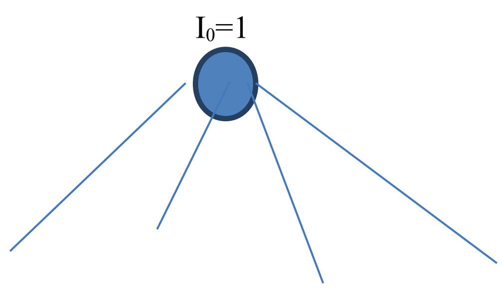
$$\begin{bmatrix} \infty & 20 & 30 & 10 & 11 \\ 15 & \infty & 16 & 4 & 2 \\ 3 & 5 & \infty & 2 & 4 \\ 19 & 6 & 18 & \infty & 3 \\ 16 & 4 & 7 & 16 & \infty \end{bmatrix} \quad \begin{matrix} 10 \\ 2 \\ 2 \\ 3 \\ 4 \end{matrix} \quad \text{minimum of each row}$$

Step-1

The cost reduction is taking minimum value is reduced from the other values in the row. Minimum value in the row is called row reduction. Row reduction value is the total sum of the row reduction values in each row. After applying reduction we get the below matrix.

$$\begin{bmatrix} \infty & 10 & 20 & 0 & 1 \\ 13 & \infty & 14 & 2 & 0 \\ 1 & 3 & \infty & 0 & 2 \\ 16 & 3 & 15 & \infty & 0 \\ 12 & 0 & 3 & 12 & \infty \end{bmatrix} \Rightarrow A \quad \begin{bmatrix} \infty & 10 & 17 & 0 & 1 \\ 12 & \infty & 11 & 2 & 0 \\ 0 & 3 & \infty & 0 & 2 \\ 15 & 3 & 12 & \infty & 0 \\ 11 & 0 & 0 & 12 & \infty \end{bmatrix} \Rightarrow 25$$

Cumulative reduction: the sum of the row reduction value + sum of the column reduction value cumulative reduction is 25



i1=2 I2=3 I3=4 I4=5

Step-2

$\hat{c}$ (s)= $\hat{c}$ (R)+ A(i, j)+r

where $\hat{c}$ (s)= cost of function at node s

$\hat{c}$ (R) = lower bound of i th node in the (i, j) path

A(i,j) = value of (i, j) in the reduced cost matrix A

r=reduced cost

At node 2 path (1, 2) – make all 1st row values to ∞ 2nd column to ∞
& (2, 1) element ∞ & remaining same
At node 2 path (1, 2)

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

At each and every matrix we apply row reduction & column reduction and finally finding reduction values, which is treated as 'small 'r'
If there is no value for 'r', it takes '0' value.

Step 3

At node 3 path (1, 3) make all 1st row values to ∞, 3 rd column to ∞ & (3,1) element ∞

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

1st row are ∞'s, 3 rd column are ∞'s, (3, 1) position are ∞'s, r=11

$\hat{c}$ (s)= $\hat{c}$ (R)+ A(i, j)+r

$\hat{c}$ (3)= 25+17+11=53

Step 4

At node 4 path (1,4)
make all 1^{st} row & 4^{th} column & (4,1) element ∞

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

1^{st} row are ∞'s
4^{th} column are ∞ 's
(4, 1) position are ∞'s
r=0
$\hat{c}(s) = \hat{c}(R) + A(i,j) + r$
$\hat{c}(4) = 25+0+0 = 25$

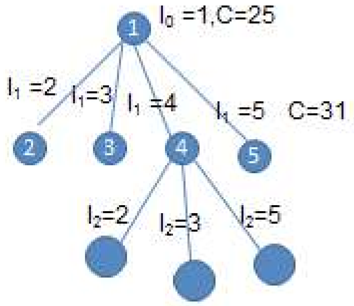
Step 5
At node 5 path(1,5) make all I st row + 5 th column + (5,1)

$$\begin{pmatrix} \infty & \infty & \infty & \infty & \infty & \infty \\ 12 & \infty & 11 & 2 & \infty & 2 \\ 0 & 3 & \infty & 0 & \infty & 0 \\ 15 & 3 & 12 & \infty & \infty & 3 \\ \infty & 0 & 0 & 12 & \infty & 0 \\ 0 & 0 & 0 & 0 & 0 & r=0+5=5 \end{pmatrix}$$

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

Min. cost = min{$\hat{c}(2)$, $\hat{c}(3)$, $\hat{c}(4)$, $\hat{c}(5)$}=25 at node 4 we have branch and bound.

Step 5 At node 5 path(1,5)



Step 6
At node 6 path(1,4,2) here 1,4 are visited, 1^{st} ,4^{th} rows are ∞'s , 2, 4 columns ∞'s (2,1)-> ∞'s

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

∞	∞	11	∞	0	→ 2
∞	∞	∞	∞	2	
∞	∞	∞	∞	∞	→ 3
11	∞	0	∞	∞	

-----5

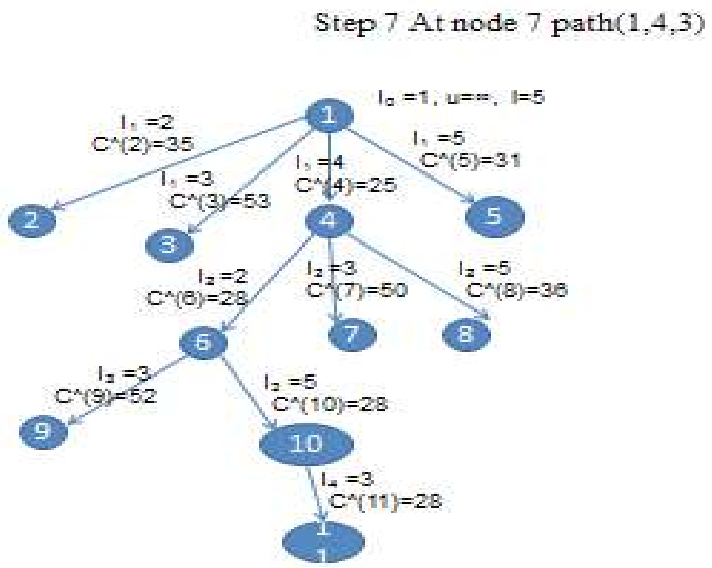
$\hat{c}$ (6)= 25+3+0=28

Step 7

At node 7 path(1,4,3) here 1,4 are visited, 1^{st} ,4^{th} rows are ∞'s , 4 ,3 rd columns ∞'s (3,1)-> ∞'s

∞	∞	∞	∞	∞	→ ∞
12	∞	11	∞	0	→ 0
∞	3	∞	∞	2	→ 2
∞	∞	∞	∞	∞	→ ∞
11	0	∞	∞	∞	→ 0

$\hat{c}$ (7)= 25+12+13=50



Step 8

At node 8 path(1,4,5) here 1,4 are visited, 1^{st} ,4^{th} rows are ∞'s ,4^{th}, 5^{th} columns ∞'s (5,1)-> ∞'s

∞	∞	∞	∞	∞	→ ∞
12	∞	11	∞	∞	→ 11
0	3	∞	∞	∞	→ 0
∞	∞	∞	∞	∞	→ ∞
∞	0	0	∞	∞	→ 0
0	0	0	∞	∞	

11+0=11

$\hat{c}(8) = 25+0+11=36$

Step 9

At node 9 path(1,4,2,3), Hence 1,4 ,2 are visited, 1^{st} ,4^{th} , 2^{nd} rows are ∞'s , 4 2,3 rd columns ∞'s (3,1)-> ∞'s

$$\begin{bmatrix} \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & \infty \\ \infty & \infty & \infty & \infty & 2 \\ \infty & \infty & \infty & \infty & \infty \\ 11 & \infty & \infty & \infty & \infty \end{bmatrix} \quad 2 \qquad\qquad r = 11+2=13$$

11

$\hat{c}(9) = 28+11+13=52$

Step 10

At node 10 path(1,4,2,5) Hence 1,4 ,2 are visited, 1^{st} ,9^{th} , 2^{nd} rows are ∞'s , 4^{th}, 2^{nd} ,5^{th} columns ∞'s (5,1)-> ∞'s

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

0

$\hat{c}(10) = 28+0+0=28$

Here the unvisited node is 3

Step 11

At node 11 path(1,4,2,5,3) Hence 1,4 ,2,5 are visited, 1^{st} ,4^{th} ,2^{nd},5^{th} rows are ∞'s , 4 2,5,3 rd columns ∞'s (3,1)-> ∞'s

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

$\hat{c}(11) = 28+0+0=28$

Final travelling salesman problem path is(1,4,2,5,3)

The total cost for TSP= 10+6+2+7+3=28

Exercise-1

Obtain optimal solution using dynamic reduction method. Draw a portion of state space tree using Branch & Bound technique. The cost matrix is given

C=

∞	11	10	9	6
8	∞	7	3	4
8	4	∞	4	8
11	10	5	∞	5
6	9	5	5	∞

Answer Total=28

Exercise -2

Consider the traveling salesperson instance defined by the cost matrix

∞	7	3	12	8
3	∞	6	14	9
5	8	∞	6	18
9	3	5	∞	11
18	14	9	8	∞

obtain the reduced cost matrix

Exercise-3

To		1	2	3	4
From	1	∞	3	9	7
	2	3	∞	6	5
	3	5	6	∞	6
	4	9	7	4	∞

Answer: Further, this tour must be minimal since its cost equals our lower bound.

Rechecking the tour's cost with the original cost matrix C,

We have C12 + C24 + C43 + C31 = 3 + 5 + 4 + 5 = 17.

We summarize the preceding reasoning with the decision tree.

7.7 15 puzzle problem

The 15-puzzle is invented by sam loyd in 1878. It consists of 15 numbers tiles on a square frame with a capacity of 16 tiles. We are given an intial arrangement of the tiles and the objective is to transform it into the goal arrangement through a series of legal moves. For example, in the given below fig sometimes, for a given initial arrangement it may not lead to a goal arrangement. In the following, we provide a theorem for testing whether or not a given initial arrangement may lead to a goal arrangement.

15 puzzle problem

An initial arrangement

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Goal Arrangement

15 puzzle problem

Theorem: The goal state is reachble from the intial state iff $\sum_{i=1}^{16}$ LESS(t) + X is even where POSITION(t) = position number in the initial state of the tile numbered i. (POSITION(16) denoted the position of empty spot.)

LESS(t)= the number of tiles j such that j<I and POSITION(j) > POSITION(t)

1, if in the initial state, the empty spot is at one of the shaded positions

0, if it is at one of the un shaded positions.

15 puzzle problem

POSITION (12) = 8

1	3	4	15
2		5	12
7	6	11	14

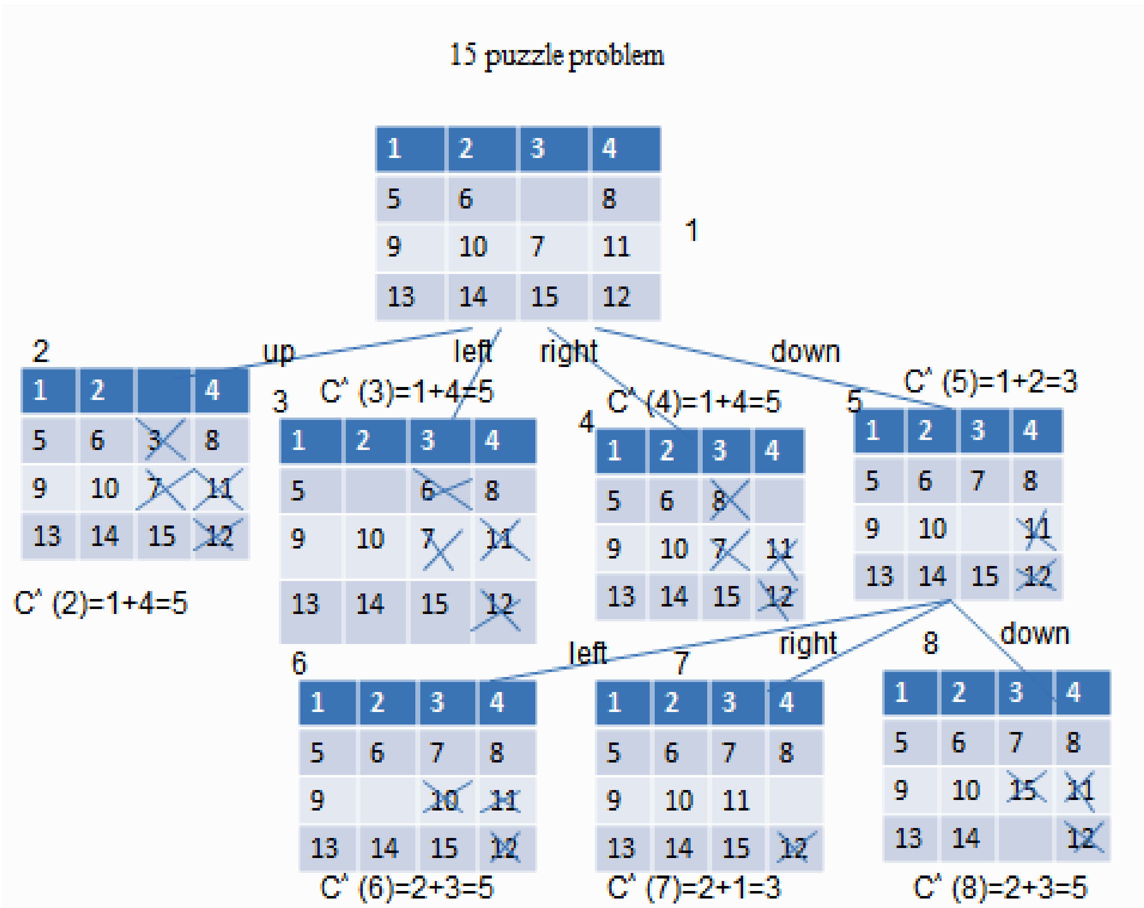
8	9	10	13

For example,
LESS(1) = 0
LESS(4) = 1
LESS(12) = 6
X = 0

Example: A state space tree organization is given in below fig

$\hat{C}(x) = f(x) + \hat{g}(x)$

$\hat{C}(x)$= estimated cost at node x
f(x) = length of the path from root node to ‘x’.
$\hat{g}(x)$=No. of non-blank tiles that are not in goal state

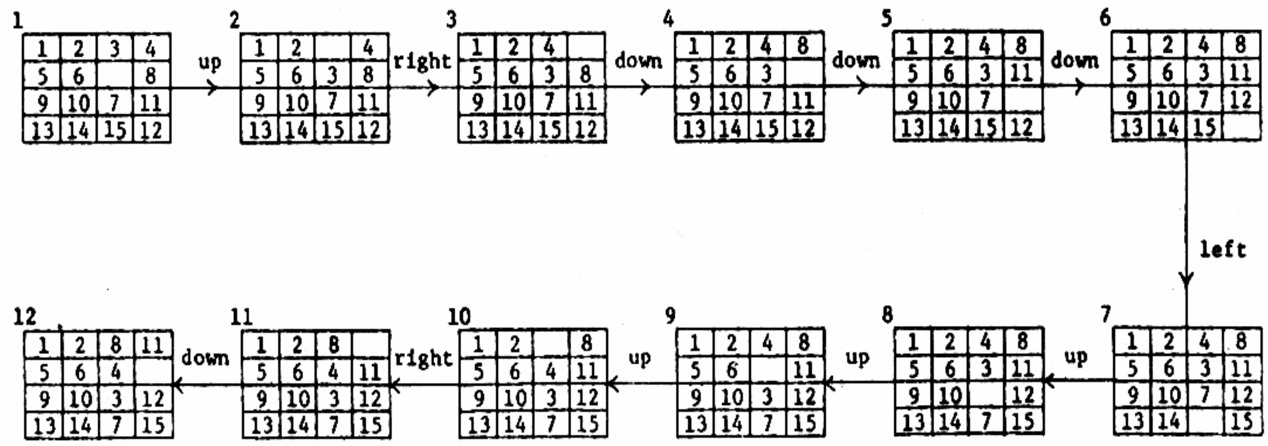


5

15 puzzle problem

Iteration	Live nodes	E-node
0	C(1)	Node 1
1	C(2)=1+4, C(3)=1+4, C(4)=1+2, C(5)=1+4	Node 4
2	C(2)=1+4, C(3)=1+4, C(5)=1+4, C(11)=2+3, C(12)=2+3	Node 10
3	C(2)=1+4, c(3)=1+4, c(5)=1+4, C(11)=2+3, C(12)=2+3, c(22)=4+2, C(23)=4+0(goal node)	goal

15 puzzle problem



First ten steps in a depth first search

Exercise

Solve the following 15 puzzle problem using B&B

1	**2**	**3**	**4**
5	6	11	7
9	10		8
13	14	15	12

****