Branch and Bound algorithms

Dr. Priyadarshan Dhabe,
Ph.D (IIT Bombay)

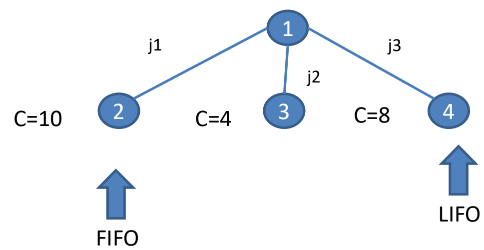
Branch and bound –general strategy

- Used for solving the **optimization problems**
- Mostly for **minimization problem** (but if we negate the function under consideration then maximization problems can also be solved)

$$\boxed{\begin{array}{c}\text{Find x such that} \\ \underset{\min}{f(x)}\end{array}}$$

- It works in a Breadth First manner (as opposed to backtracking)
- For each **branch** it uses **bound** (number) and a given branch is chosen based on the bound, for **further exploration** if multiple choices are there.

State space search tree generation in branch and bound- Least cost approach



Each node has cost C and next node will be selected based on the minimum cost

https://www.youtube.com/watch?v=tKvAniEbeqM

0/1 knapsack problem using branch and bound- Least cost approach

N=4
M=15

P	10	10	12	18
w	2	4	6	9

Find objects to be kept in the sack to maximize the profit. But branch and bound solves minimization problem. Thus, negate all the profits

N=4
M=15

P	-10	-10	-12	-18
w	2	4	6	9

It works by computing lower bound c and upper bound u on each node and takes the decision.
c-lower bound- fractions are allowed(for calculations only)
u- upper bound- fractions are not allowed
-decides a branch to pursue based on bounds

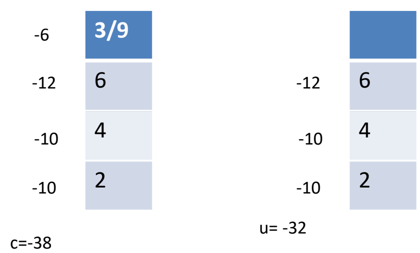
https://www.youtube.com/watch?v=tKvAniEbeqM

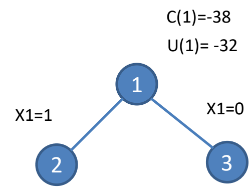
0/1 knapsack problem using branch and bound- Least cost approach

N=4
M=15

P	-10	-10	-12	-18
w	2	4	6	9

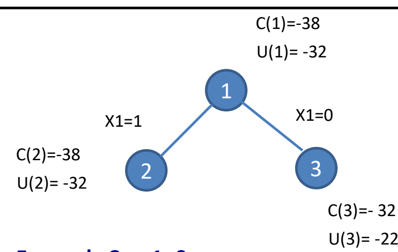
For node 1



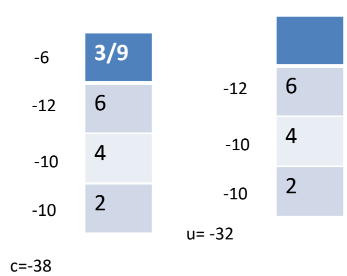


3/9*(-18)=- 6

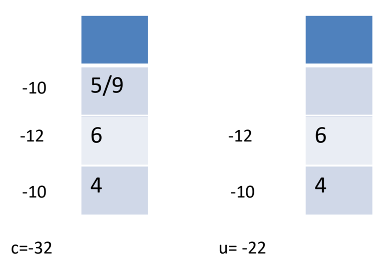
P	-10	-10	-12	-18
w	2	4	6	9



For node2 - x1=1

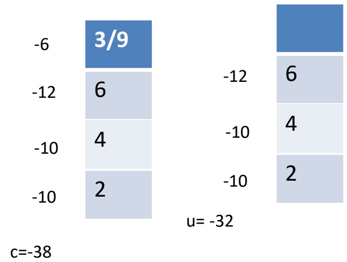


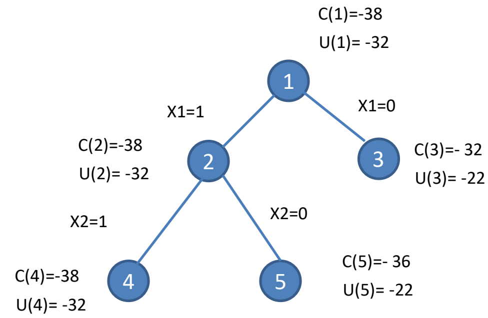
For node 3 - x1=0



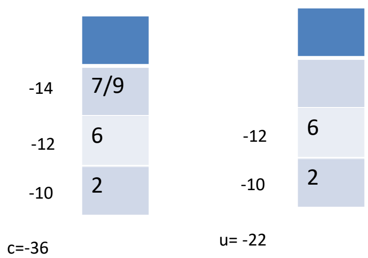
Choose the next node for exploration as a node with minimum lower bound c, thus we will pick node 2.

For node4 - x1=1, x2=1





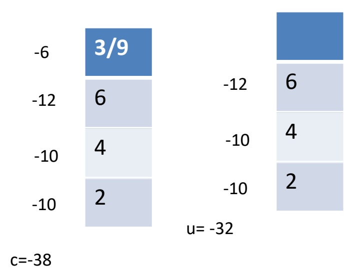
For node 5 - x1=1, x2=0

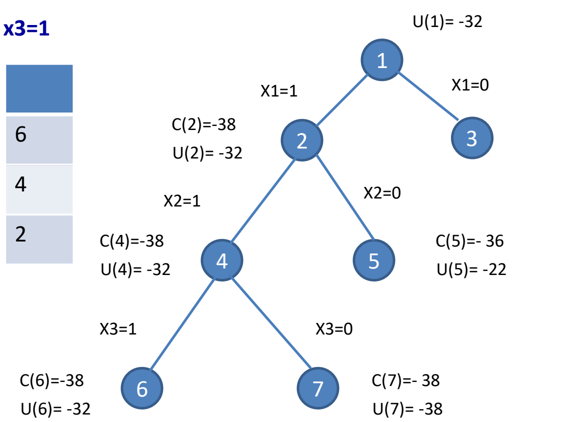


Expand the node 4 since it has minimum c value

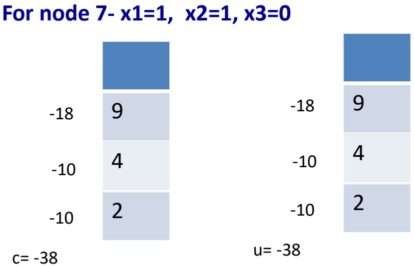
P	-10	-10	-12	-18
w	2	4	6	9

For node 6- x1=1, x2=1, x3=1





For node 7- x1=1, x2=1, x3=0



Since node 6 and 7 has same c, chose node with minimum u i.e node 7

P	-10	-10	-12	-18
w	2	4	6	9

Node 8 – x=1, x2=1, x3=0, x4=1

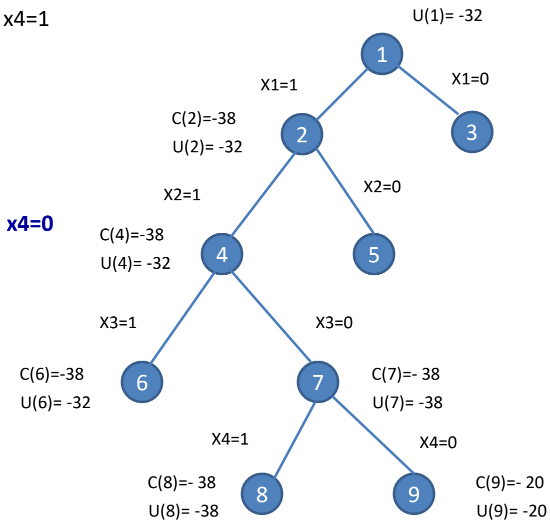
C(8)=- 38
U(8)= -38

Node 9 – x=1, x2=1, x3=0, x4=0

C(9)=- 20
U(9)= -20

Since node 8 is having minimum c, choose 8

Knapsack instance (x1,x2,x3,x4)= (1,1,0,1)



Maximum profit obtained is = -(-38)=+38

https://www.youtube.com/watch?v=tKvAniEbeqM

Solve following 0/1 knapsack problem using branch and bound- Least cost approach

N=4-objects
M=15- Knapsack capacity

P	10	10	12	9
w	2	4	6	9

- Solve the same problem using **FIFO branch and bound** and **LIFO branch and bound**
  - **(hint updating global upper bound, if a node has lower bound greater than the global upper bound kill that node)**

https://www.youtube.com/watch?v=WXWt5xNrOf4