Scheduling

Notes 7

Branch and Bound Algorithm

- A large number of real-world planning problems called combinatorial optimization problems. They are optimization problems, are easy to state, and have a finite but usually very large number of feasible solutions.
- Some of the examples of combinatorial optimization problem types are vehicle routing, crew scheduling, and production planning problems.
- Branch and Bound (B&B) is by far the most widely used tool for solving combinatorial optimization problems.

Branch and Bound Algorithm

- The Branch&Bound is a solution approach that can be applied to a number of different types of problems. It is based on the principles that the total set of feasible solutions can be partitioned into smaller subsets of solutions. These smaller subsets can then be evaluated systematically until the best solution is found.

Branch and Bound Algorithm

- Branch-and-Bound uses a partition of the solution space into subsets.
- Usually the subsets are arranged in a tree structure.
- Branch-and-bound organizes the search so that nodes in the search tree are expanded one-by-one.
- It expands a node if and only if that node has the potential to yield a better solution. This potential can be determined by calculating a bound on the value of solutions in the subtree rooted at the node.

Branch and Bound Algorithm

- Useful to solve small instances of hard problems
- Can be applied to all scheduling problems
- Takes exponential time in the worst-case
- Guarantees optimal solution

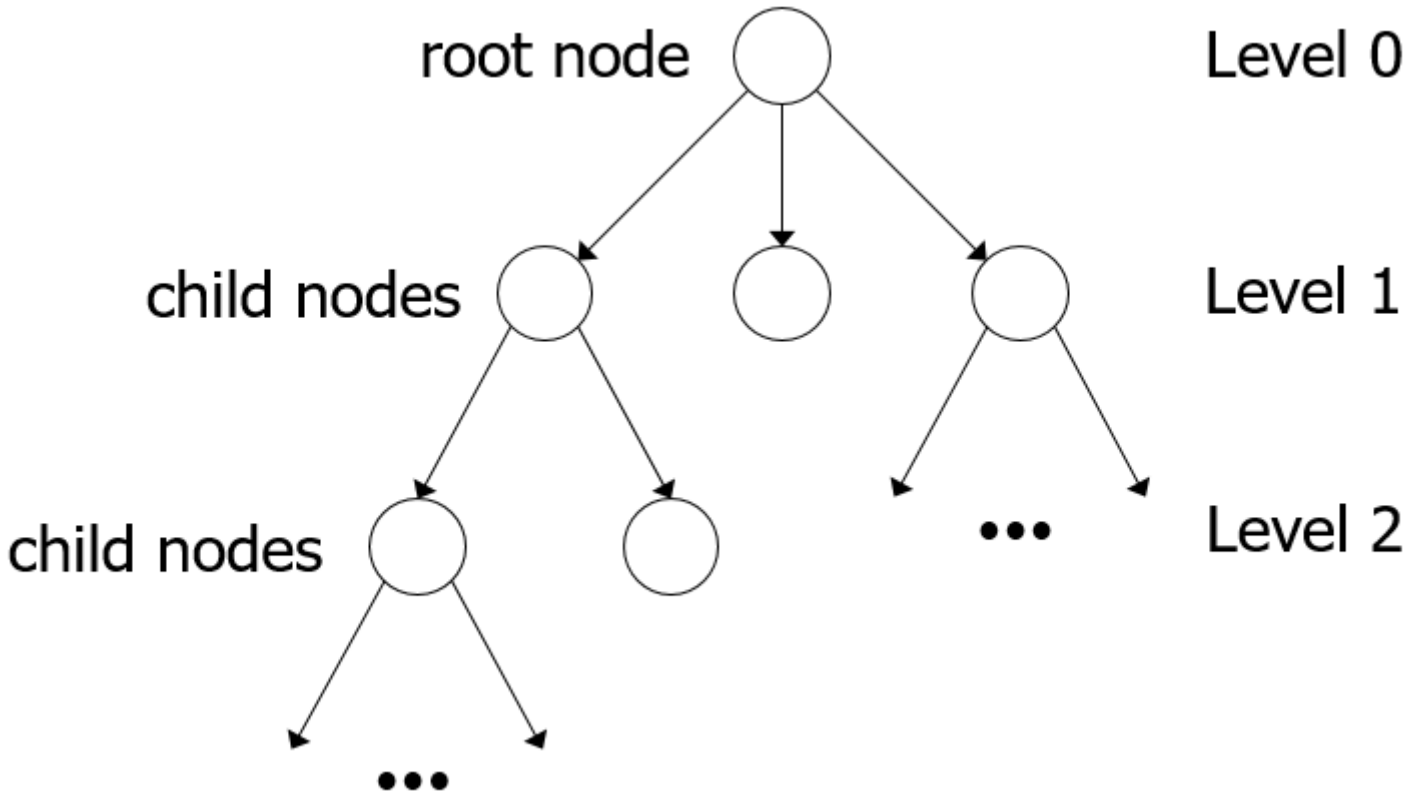
Branch and Bound Algorithm

- To generate a schedule we start with no job sequenced and indicate this by (X,X,X,X). Here “X” in the job sequence indicates that no job has yet been assigned to that position.
- Step 1: To construct a schedule starting from the first position, we move from node (X,X,X,X) to one of the four possible nodes (1,X,X,X); (2,X,X,X); (3,X,X,X); (4,X,X,X).
- Step 2: To assign the second job in the sequence, we branch from the each of these four nodes to three possibilities: branching from (1,X,X,X) gives (1,2,X,X), (1,3,X,X), and (1,4,X,X); branching from (2,X,X,X) gives three possibilities (2,1,X,X), (2,3,X,X), and (2,4,X,X), etc.

Branch and Bound Algorithm

- Step 3: Assigning the job to be processed in the third position immediately fixes the last job. This process is represented by a branching tree. Each node of a tree corresponds to a partial schedule with several jobs assigned to the first positions and the remaining jobs unassigned. To avoid full enumeration of all job permutations, we calculate in each step the Lower Bound (LB) of the value of the objective function for each partial schedule.

Branch and Bound Algorithm



4/3/F/Cmax Example

The algorithm will be explained on a 4/3/F/Cmax problem.

ri= 0

We calculate a lower bound (lb) for Cmax and this shows if we continue to search this branch of the tree.

- For presentation suitability, for every Ji job,

$$a_i = p_{i1}$$

$$b_i = p_{i2}$$

$$c_i = p_{i3}$$

- While we analyze the bounds, we can expand the problem to n jobs.
- J1, J2, ……., Jn.
- We assume that we are on the XX…X node.
- So, to the first k position during the process, k jobs will be assigned.
- A= {J1, J2, ……,Jk} shows the assigned jobs.
- U means the the unassigned (n-k) jobs

- αi(k) = the completion time of Ji(k) on the first machine.
- βi(k)= the completion time of Ji(k) on the second machine.
- γi(k)= the completion time of Ji(k) on the third machine.

- $\propto_{i(1)}=a_{i(1)}$ $\propto_{i(k)}=\propto_{i(k-1)}+a_{i(k)}$
- $\beta_{i(1)}=a_{i(1)}+b_{i(1)}$ $\beta_{i(k)}=\max\{\propto_{i(k)},\beta_{i(k-1)}\}+b_{i(k)}$
- $\gamma_{i(1)}=a_{i(1)}+b_{i(1)}+c_{i(1)}$ $\gamma_{i(k)}=\max\{\beta_{i(k)},\gamma_{i(k-1)}\}+c_{i(k)}$

- To find the lower bound, we take three most suitable possibilities into account. We search how fast we can finish the jobs in the set U.
- First, we consider the possibility that the process on machine 1 is continuously.
- We assume that the last job Ji(n) should not wait that the machine 2 and machine 3 will be free.
- So we will see that Ji(n) will be processed on machine 2 and on machine 3 minimum.

- $C_{max} \geq \propto_{i(k)} + \sum_{ji \in U} a_i + min_{ji \in U}\{b_i + c_i\}$
- Then we consider the possibility that the jobs on machine 2 are continuous.
- We assume that machine 2 doesn't be free while waiting the jobs from machine 1.
- We assume that the last job Ji(n) should not wait the completion of Ji(n-1) and it can be completed on machine 3.
- $C_{max} \geq \beta_{i(k)} + \sum_{ji \in U} b_i + min_{ji \in U}\{c_i\}$

- Lastly, we assume that machine 1 and machine 2 are processing so that machine 3 will not wait freely.
- $C_{max} \geq \gamma_{i(k)} + \sum_{ji \in U} c_i$
- This means, no sequence could be better than these three equations' maximum. And we found the lower bound.

- lb(A)= max $\{\propto_{i(k)} + \sum_{ji \in U} a_i + min_{ji \in U}(b_i + c_i),$

$\beta_{i(k)} + \sum_{ji \in U} b_i + min_{ji \in U} c_i,$

$\gamma_{i(k)} + \sum_{ji \in U} c_i\}$

- This lower bound can be lower than the sub-sequences. While producing the three components of the lower bound, we assume that there is no waiting time and no free time at specific points.

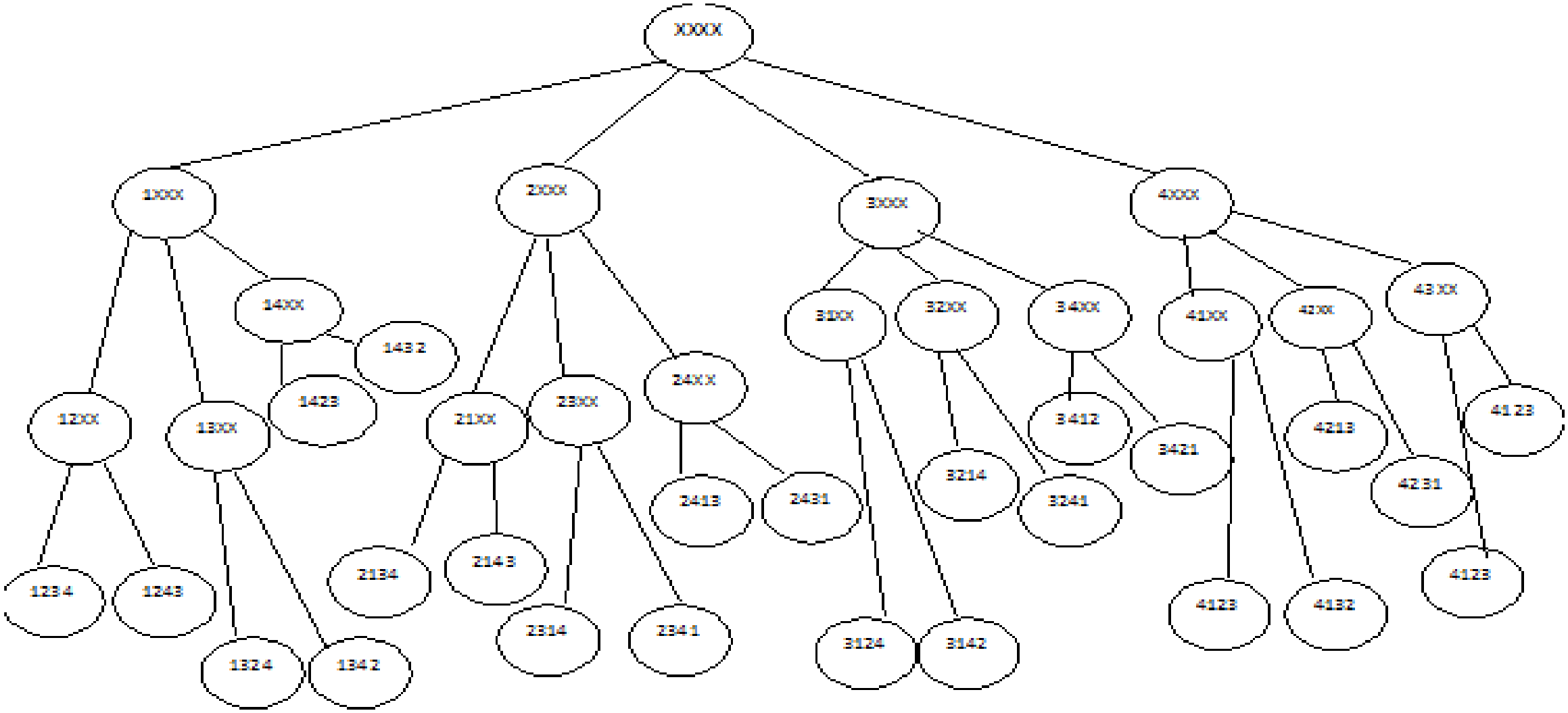
Example 4/3/F/Cmax

Jobs	Processing time		
	ai	bi	ci
1	1	8	4
2	2	4	5
3	6	2	8
4	3	9	2

- For ease of calculation, we make a table for the values $min_{ji \in U}(b_i +$

Jobs	bi+ci
2	9
3	10
4	11
1	12

Jobs	ci
4	2
1	4
2	5
3	8


XXXX
1XXX
2XXX
3XXX
4XXX
14XX
1432
12XX
13XX
1423
21XX
23XX
24XX
2413
2431
2134
2143
2314
2341
31XX
32XX
34XX
3412
3421
3214
3241
3124
3142
41XX
42XX
43XX
4213
4231
4123
1234
1243
1324
1342
4123
4132

- We start calculating lower bounds on every nodes of the Cmax according to 1234 sequence.
- We are going from XXXX, to 1XXX then to 12XX and finally to 1234.
- At 1234 node, we obtain the Cmax (not the lower bound)
- 1XXX:
- A={J1} U={J2,J3,J4}
- $\alpha 1=1$ $\beta 1=1+8=9$ $\gamma 1= 1+8+4=13$
- lb(A)= max{ 1+11+9, 9+15+2, 13+15}

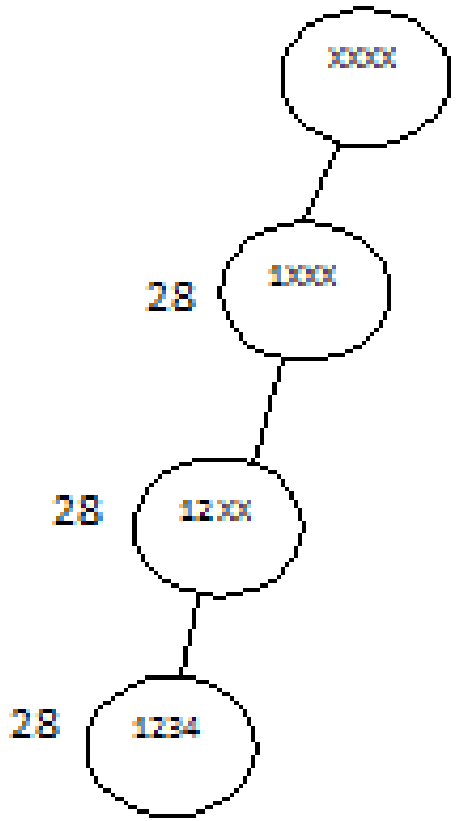
  = 28

- 12XX:
- A={J1,J2} U={J3,J4}
- $\alpha 2 = 1+2=3$ $\beta 2 = \max\{3,9\}+4 = 13$ $\gamma 2 = \max\{13,13\}+5 = 18$
- $lb(A) = \max\{3+9+10, 13+11+2, 18+10\}=28$

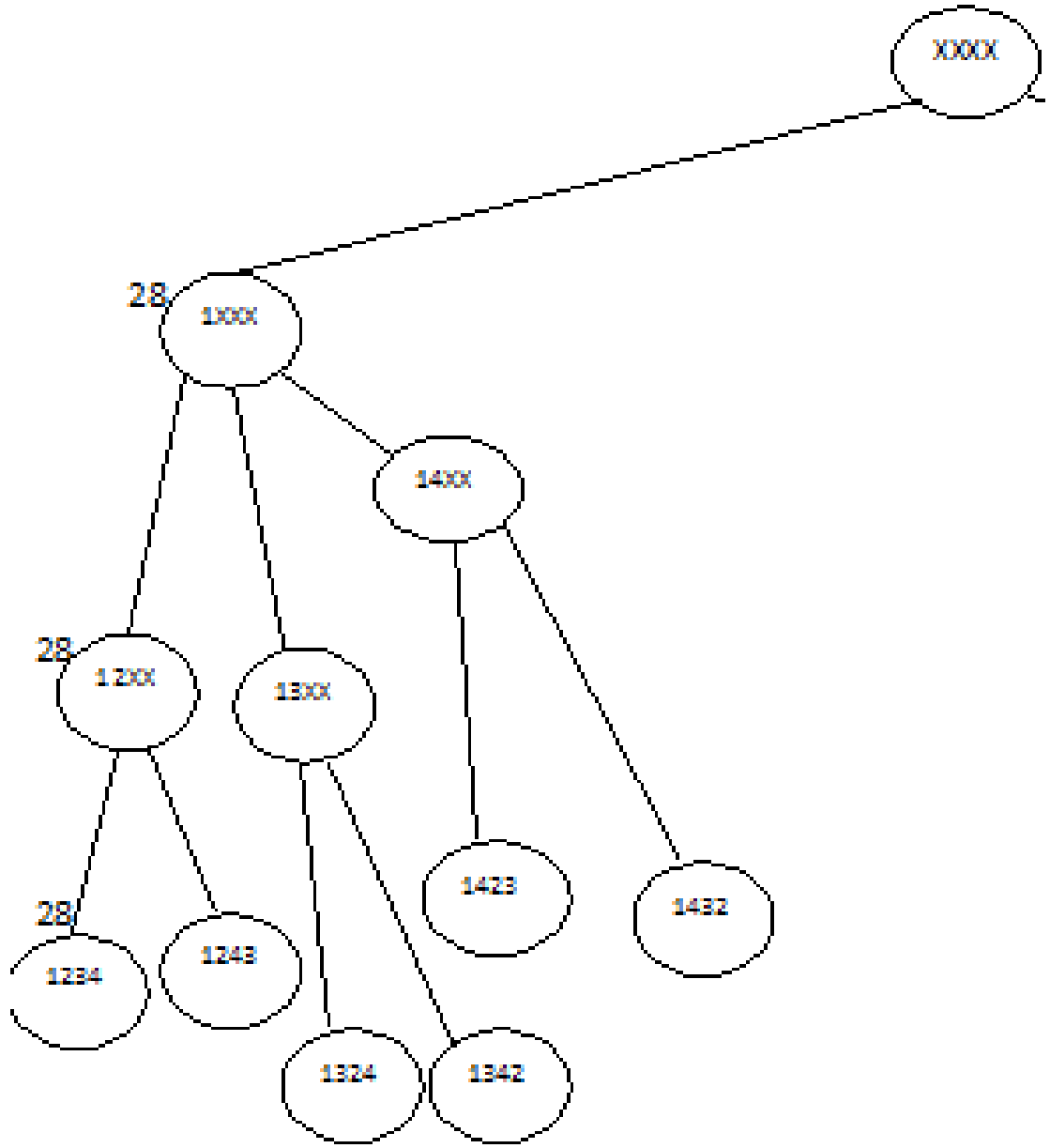
- 1234:
- A={J1,J2,J3,J4}
- $\alpha 3 = 3+6=9$ $\beta 3 = \max\{9,13\}+2 = 15$ $\gamma 3 = \max\{15,18\}+8 = 26$
- $\alpha 4 = 9+3=12$ $\beta 4 = \max\{12,15\}+9 = 24$ $\gamma 4 = \max\{24,26\}+2 = 28$

- **Cmax = 28 (1234 sequence)**

- 1234 is the first test sequence. If the lower bound on a node is equal or greater than the test Cmax, we can not improve Cmax while investigating that branch. So we eliminate that node and all the nodes after that.
- If the lower bound on a node is less than the test Cmax, we can not eliminate that node, we should investigate further.
- If the Cmax on an final node is less than the test sequence's Cmax, that sequence will be the new test sequence.


XXXX
28
1XXX
28
12XX
28
1234

- The lower bound for 12XX is 28. It means that we could not improve the Cmax, when we investigate 1243. Similarly, the lower bound for 1XXX is 28. We can eliminate 13XX , 14XX and the branches behind.


XXXX
28
1XXX
14XX
28
12XX
13XX
28
1234
1243
1423
1432
1324
1342

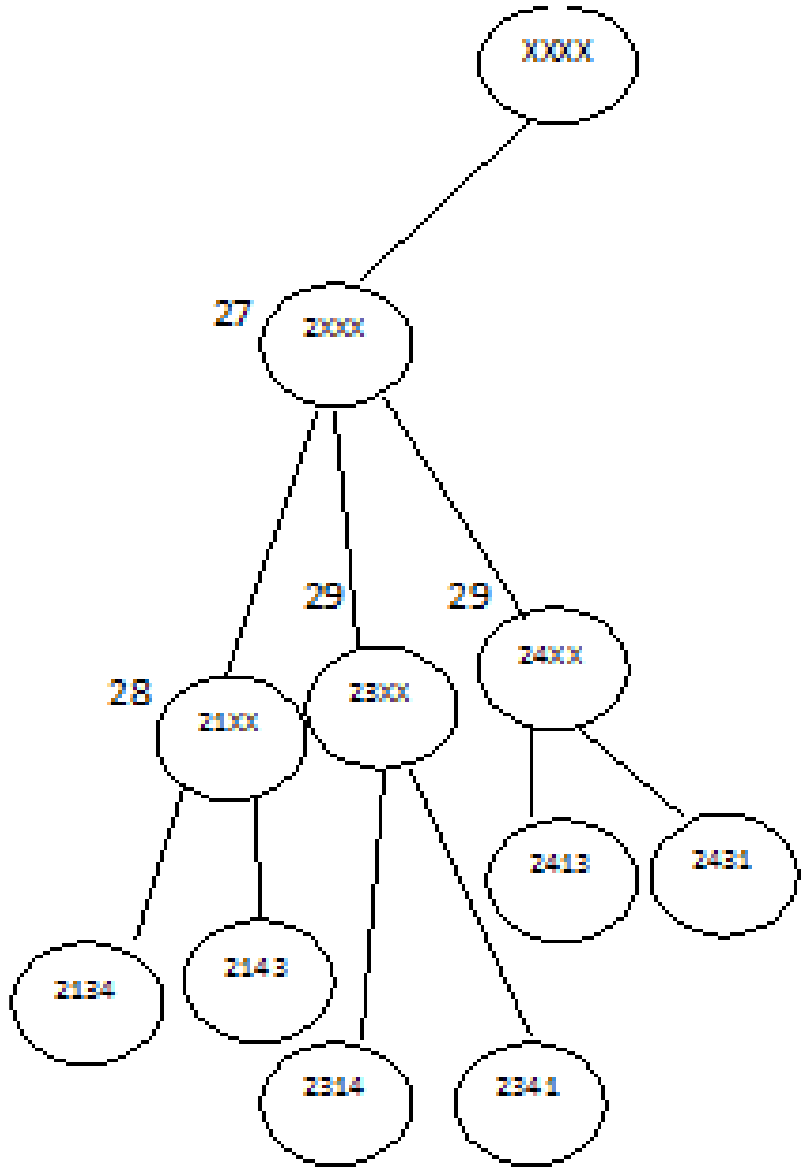
- We continue with 2XXX
- A={J2} U={J1,J3,J4}
- $\alpha 2=2$ $\beta 2=2+4=6$ $\gamma 2= 2+4+5=11$
- lb(A)= max{2+10+10, 6+19+2, 11+14}= **27**
- This lower bound is less than the test sequence Cmax. We should analyze the branches on the node 2XXX further.

- **21XX:**
- **A={J2,J1} U={J3,J4}**
- **α1= 2+1=3 β1= max{3,6}+8= 14 γ1= max{14,11}+4= 18**
- **lb(A)= max{2+9+10, 14+11+2, 18+10}= 28**

- 23XX:
- A={J2,J3} U={J1,J4}
- $\alpha 3 = 2+6=8$ $\beta 3 = \max\{8,6\}+2 = 10$ $\gamma 3 = \max\{10,11\}+8 = 19$
- $lb(A) = \max\{8+4+11, 10+17+2, 19+6\} = 29$

- 24XX:
- A={J2,J4} U={J1,J3}
- $\alpha 4 = 2+3=5$ $\beta 4 = \max\{5,6\}+9 = 15$ $\gamma 4 = \max\{15,11\}+2 = 17$
- $lb(A) = \max\{5+7+10, 15+10+4, 17+12\} = 29$

- The lower bound on these notes are not less than the Cmax of the test sequence. So, we don’t need to analyze further.


XXXX
27
2XXX
29
29
28
21XX
23XX
24XX
2134
2143
2314
2341
2413
2431

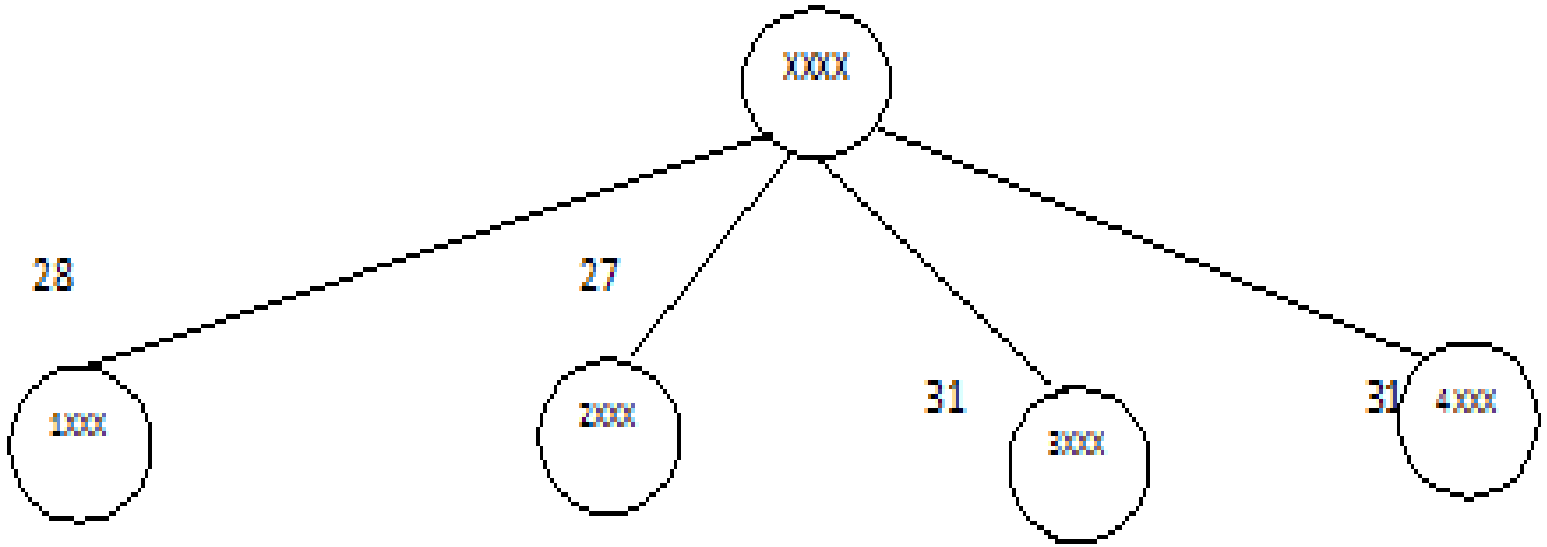
- **3XXX:**
- **A={J3} U={J1,J2,J4}**
- **$\alpha 3 = 6$ $\beta 3 = 8$ $\gamma 3 = 16$**
- **lb(A)= max{6+6+9, 8+21+2, 16+11}= 31**
- **This lower bound is greater than 28, we don't analyze further.**

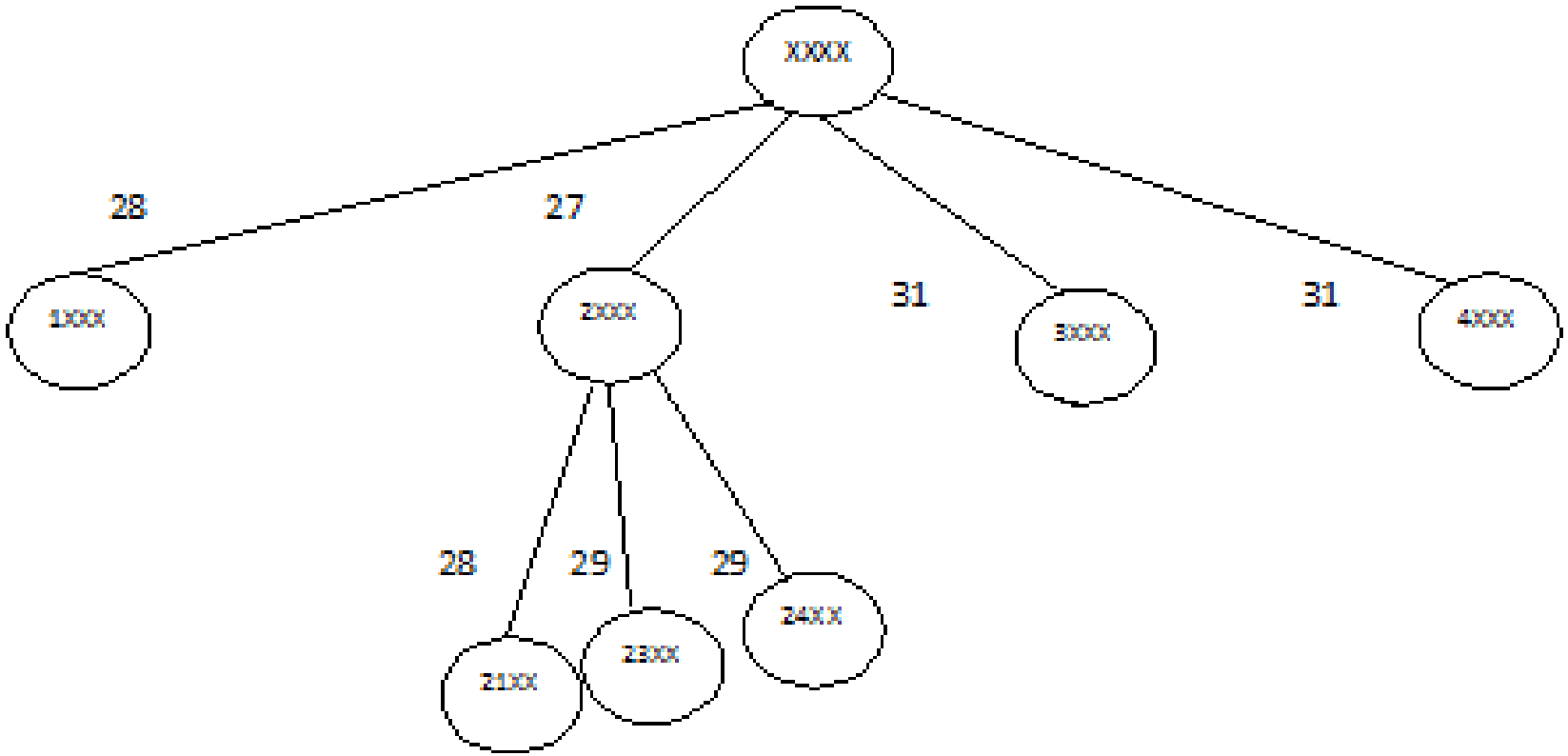
- **4XXX:**
- **A={J4} U={J1,J2,J3}**
- **α4= 3 β4= 12 γ4= 14**
- **lb(A)= max{3+9+9, 12+14+4, 14+17}=31**
- **This lower bound is greater than 28, we don’t analyze further.**

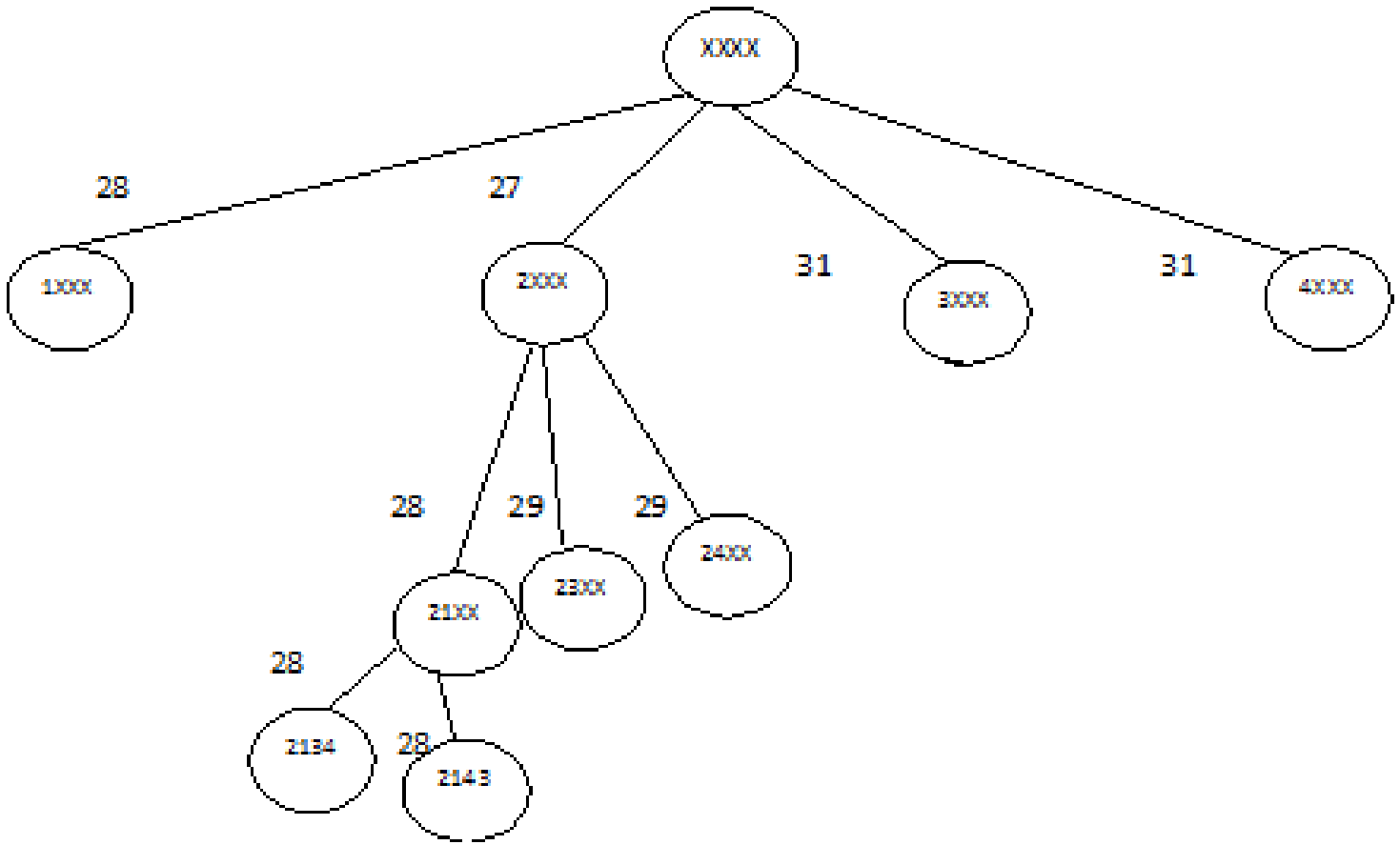
- We found that no sequence can have a Cmax lower than 28.
- The beginning test sequence **1234** is optimal.

- This search algorithm is called depth first search.
- We chose a branch and we systematically eliminated some of the branches or we came to the end note. At every node we compare the lower bounds with the Cmax of the test sequence. The end note will be eliminated or will be the new test sequence.

- **Another search strategy is ‘’frontier search’’ or ‘’branch-from-lowest-bound’’.**




XXXX
28
27
31
31
1XXX
2XXX
3XXX
4XXX
28
29
29
21XX
23XX
24XX


XXXX
28
27
31
31
1XXX
2XXX
3XXX
4XXX
28
29
29
21XX
23XX
24XX
28
28
2134
2143

Resources

- Sıralama ve Programlama, Hüseyin Başlıgil
- Çizelgeleme Ders Notları, Prof. Dr. Hüseyin Başlıgil
- http://syllabus.cs.manchester.ac.uk/pgt/2017/COMP60342/COMP60342-2015-lec4.BranchandBound.pdf
- http://web.tecnico.ulisboa.pt/mcasquilho/compute/_linpro/TaylorB_module_c.pdf
- http://www.imada.sdu.dk/Employees/jbj/DM85/TSPtext.pdf
- http://web-static.stern.nyu.edu/om/faculty/pinedo/scheduling/shakhlevich/handout06.pdf
- www.columbia.edu/~cs2035/courses/ieor4405.S03/bb.doc
- http://cs.uef.fi/pages/franti/asa/DAA-Branch-and-bound.ppt
- http://www.stern.nyu.edu/om/faculty/pinedo/book2/downloads/Twente-Holland/Lecture%203.ppt