IO543: Scheduling

Lecture 15

Brian Denton
Associate Professor, IOE

IOE Bld 2893
btdenton@umich.edu

Last Time

Job Shop Game:

- This was a “simulation” of a 4 machine job shop
- The game was played two times:
  - Once with random releases and sequencing at machines
  - Once with clever ideas to improve performance

Job Shop Game Example

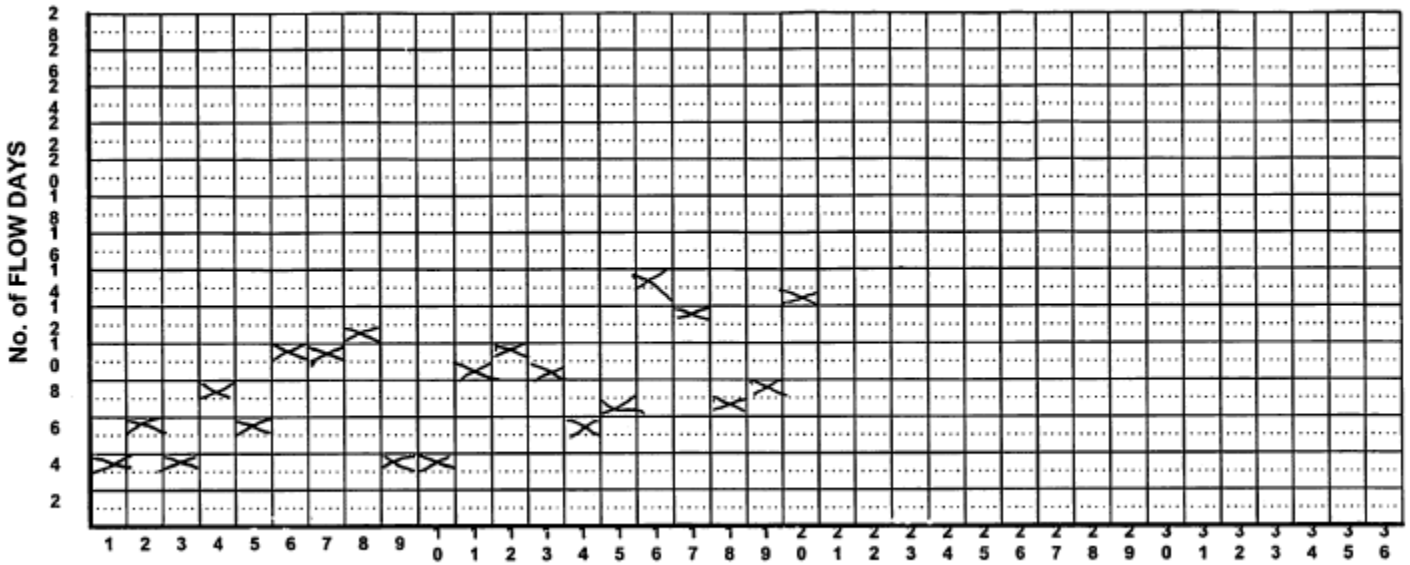
Example Strategy:

- Order products in a specific way so as to get flow as continuous as possible by reducing the bottleneck at machine B
- In the case of 2 or more jobs at a machine the operator picks the job released first

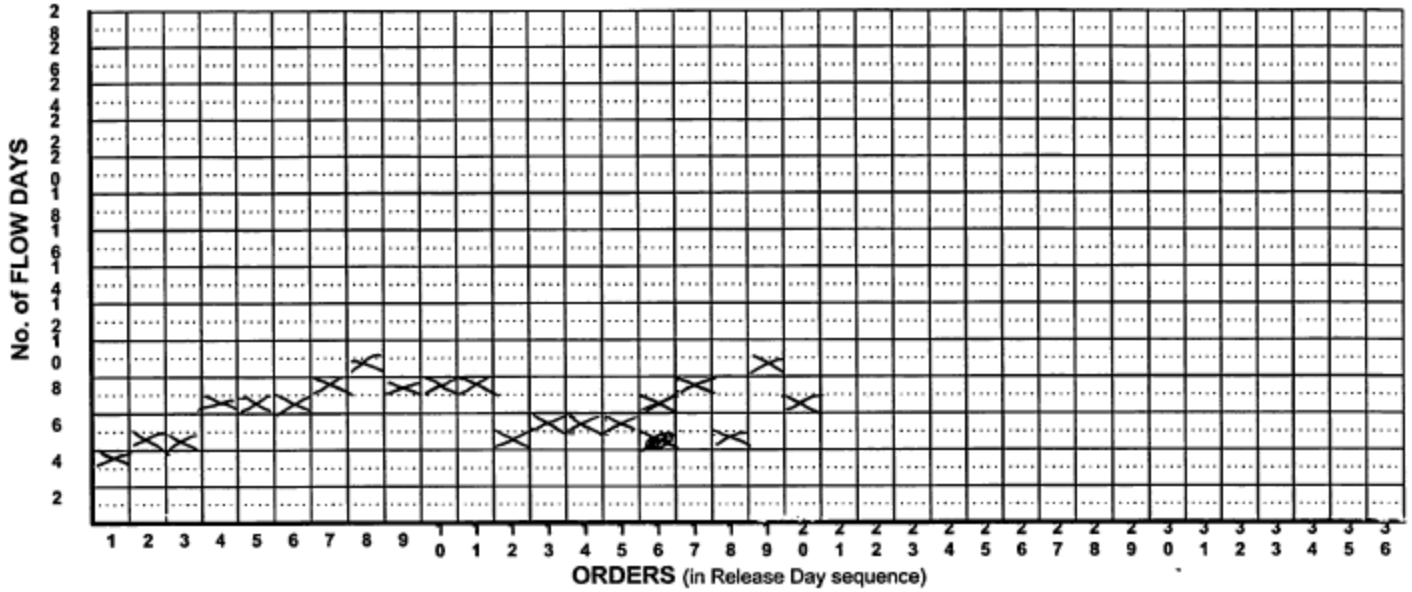
What do you expect to happen?

Job Shop Game Example

Before

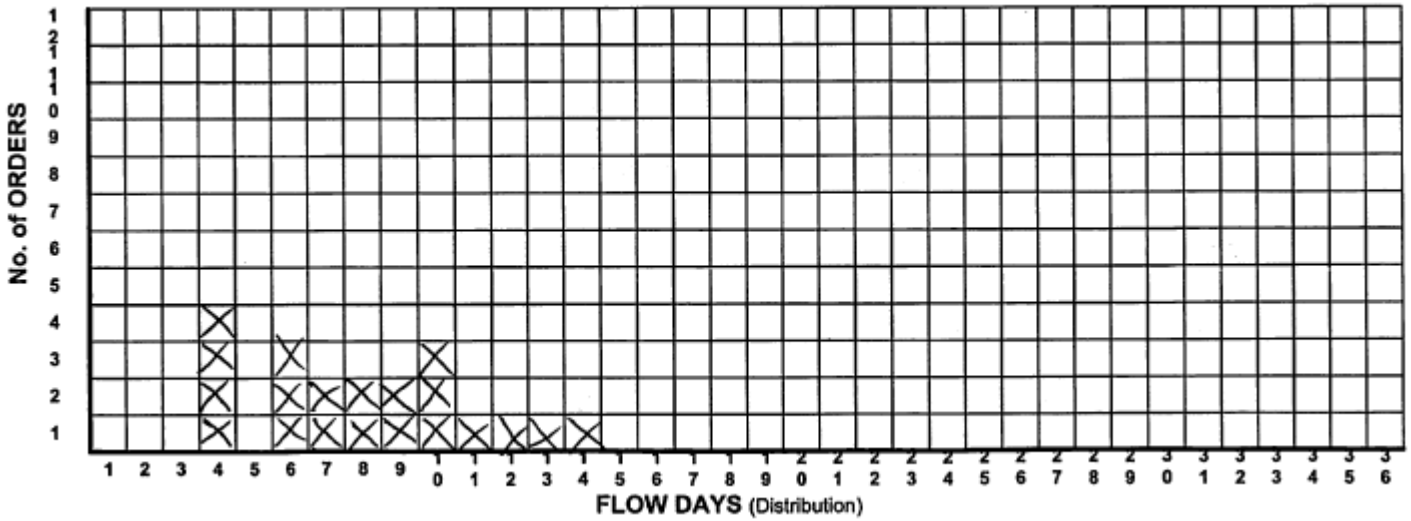


After

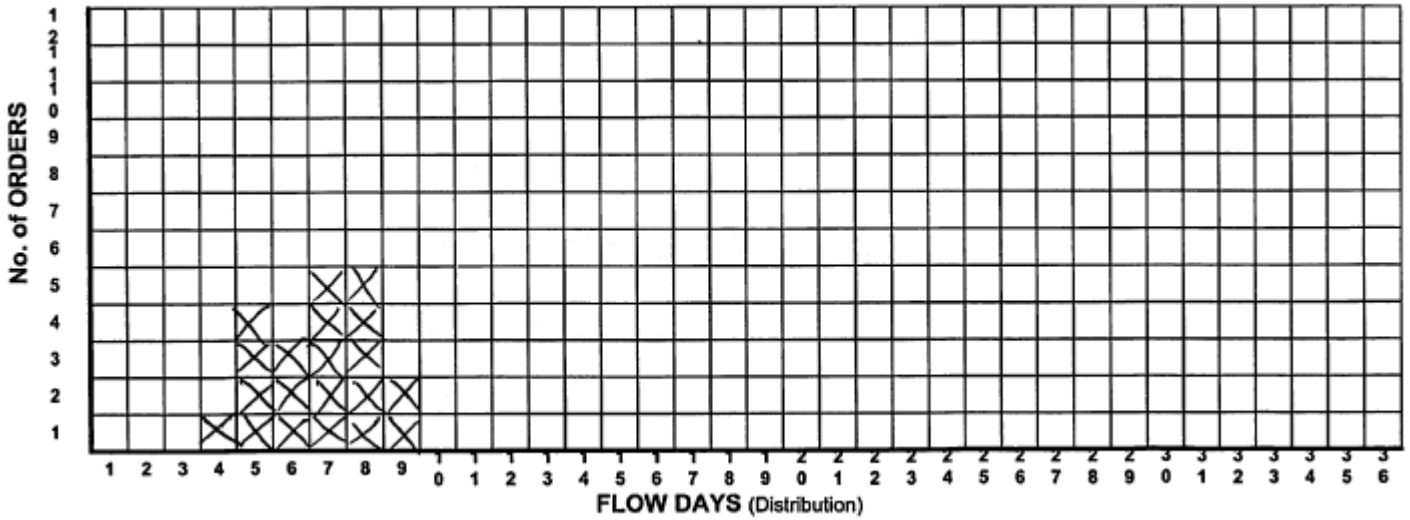


Job Shop Game Example

Before



After



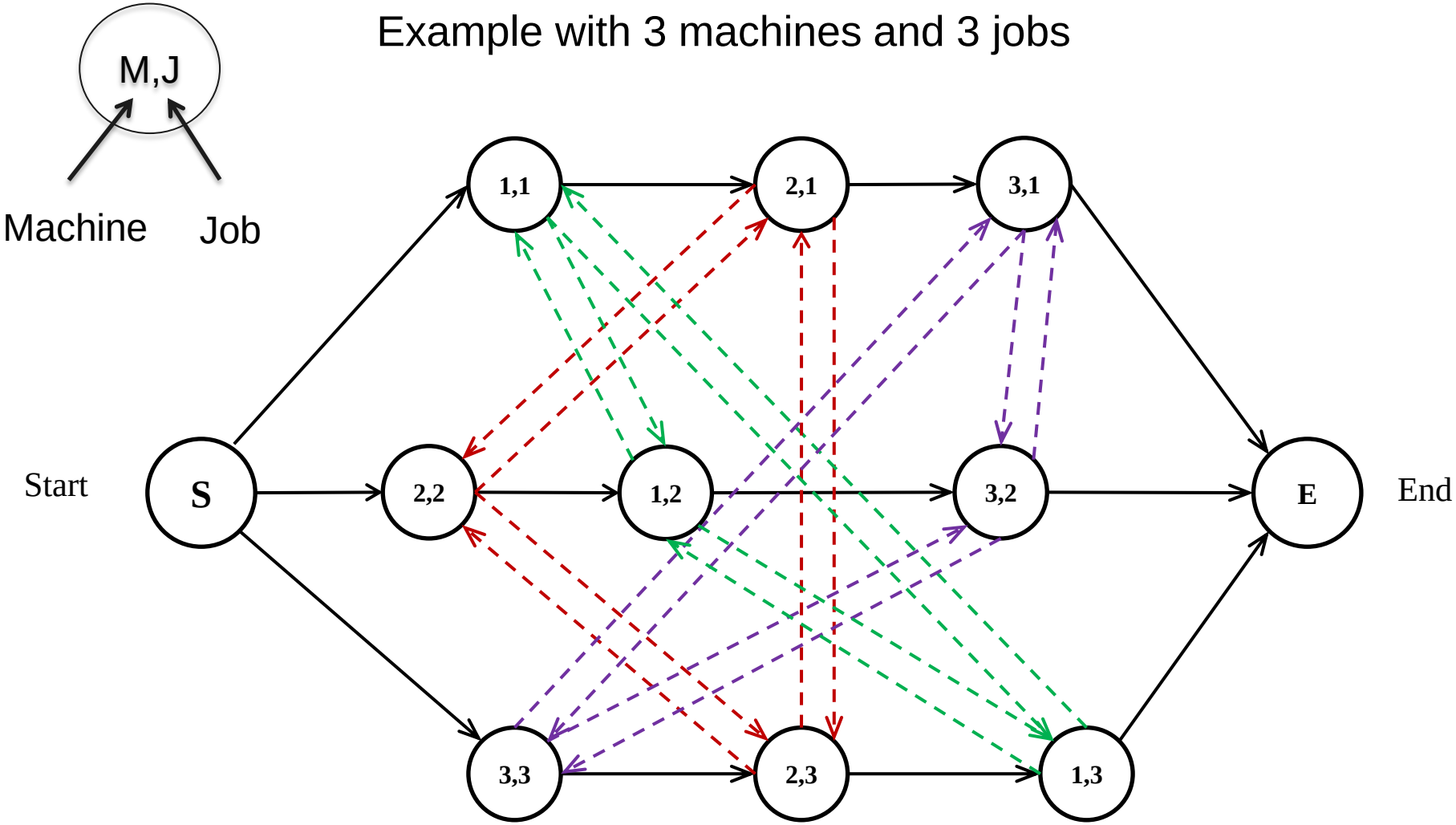
TSP Movie!

Travelling Salesman Movie:

Travelling Salesman is an award-winning, critically acclaimed feature film about a team of mathematicians who prove the impossible--then must wrestle with the massive implications of their world-changing discovery.

http://www.travellingsalesmanmovie.com/

Jobs Shop Disjunctive Network



Job Shop Disjunctive Program

$$min \quad C_{max}$$

A denotes the set of Arcs and N the set of nodes

subject to

$$y_{kj} - y_{ij} \geq p_{ij}$$ for all $(i,j) \rightarrow (k,j) \in A$

$$C_{max} - y_{ij} \geq p_{ij}$$ for all $(i,j) \in N$

$$y_{ij} - y_{il} \geq p_{il} \text{ or } y_{il} - y_{ij} \geq p_{ij}$$ for all (i,l) and (i,j), $i=1,\ldots,m$

$$y_{ij} \geq 0$$ for all $(i,j) \in N$

Job Shop Integer Program

$$min \quad C_{max}$$

subject to

$$y_{kj} - y_{ij} \geq p_{ij} \qquad \text{for all } (i,j) \rightarrow (k,j) \in A$$

$$C_{max} - y_{ij} \geq p_{ij} \qquad \text{for all } (i,j) \in N$$

$$y_{ij} - y_{il} + M(1-x_{ilj}) \geq p_{il} \qquad \text{for all } (i,l) \text{ and } (i,j),\ i=1,\ldots,m$$

$$y_{il} - y_{ij} + Mx_{ilj} \geq p_{ij} \qquad \text{for all } (i,l) \text{ and } (i,j),\ i=1,\ldots,m$$

$$y_{ij} \geq 0 \text{ for all } (i,j) \in N,\ x_{ijl} \text{ binary for all } (i,j,l)$$

Job Shop Integer Program

The integer programming formulation for $Jm||C_{max}$ can be used to solve **small instances**

Even moderate sized problems can be **hard to solve** using standard branch-and-bound because the LP relaxation is weak

Specialized branch-and-bound methods have been created for $Jm||C_{max}$

Review: Branch-and-Bound

Branch-and-Bound for IPs:

- Branch-and-bound **implicitly** enumerates all solutions using a **tree**
- Subproblems are generated by **branching** on selected fractional decision variables until all variables are integer
- By solving **LP subproblems** many branches may be eliminated from consideration (**fathomed**)

Review: Binary Tree

Knapsack Problem:

$$\max z = 12x_1 + 22x_2 + 16x_3 + 8x_4$$
$$\text{s.t.} \quad 4x_1 + 7x_2 + 5x_3 + 3x_4 \leq 14$$
$$x_i = 0 \text{ or } 1 \quad (i = 1,2,3,4)$$

- Drop the binary constraints and solve the root node **LP relaxation**:

  $x_1 = 0.5$, $x_2 = x_3 = 1$, $x_4 = 0$, $z = 44$

- Choose a fractional variable to branch on and create two subproblems:

  $x_1 = 0$ $\qquad$ $x_1 = 1$

- Solve the two subproblem relaxations

Job Shop Branch-and-Bound

- Branch-and-Bound for $Jm \mid \mid C_{max}$ uses branching on **active schedules**

  Definition: *A feasible schedule is **active** if it cannot be altered in any way such that some operation is completed earlier and no other operation is completed later*

- There is always an **active schedule** that **minimizes makespan**

Total Enumeration

- A **partial schedule** Ω at a given node is the set of all operations for which all predecessors have been scheduled

- The earliest completion time for the current partial schedule is determined by:

$$t(\Omega) = \min_{(i,j)\in\Omega}\{r_{ij} + p_{ij}\}$$

Release time, $\mathrm{r_{ij}}$, is the **longest path** from the start node, *S*, to node *(i,j)* for the network defined by the partial schedule.

The **partial schedule network** contains all of the conjunctive and disjunctive arcs for scheduled jobs

Total Enumeration

The following algorithm generates a tree that enumerates all active schedules for $Jm \mid \mid C_{max}$:

Step 1: Let Ω contain the first operation of each job; Let $r_{ij} = 0$, for all $(i, j) \in \Omega$.

Step 2: Select a pending node; Compute for the current partial schedule

$$t(\Omega) = \min_{(i,j)\in\Omega} \{r_{ij} + p_{ij}\}$$

and let i^* denote the machine that achieves the minimum

Step 3: Let Ω' be the set of operations on machine i^* such that $r_{i^*j} < t(\Omega)$. Create a branch for each operation in Ω'. For each new node: delete the operation from Ω, and include its immediate follower in Ω. Return to Step 2.

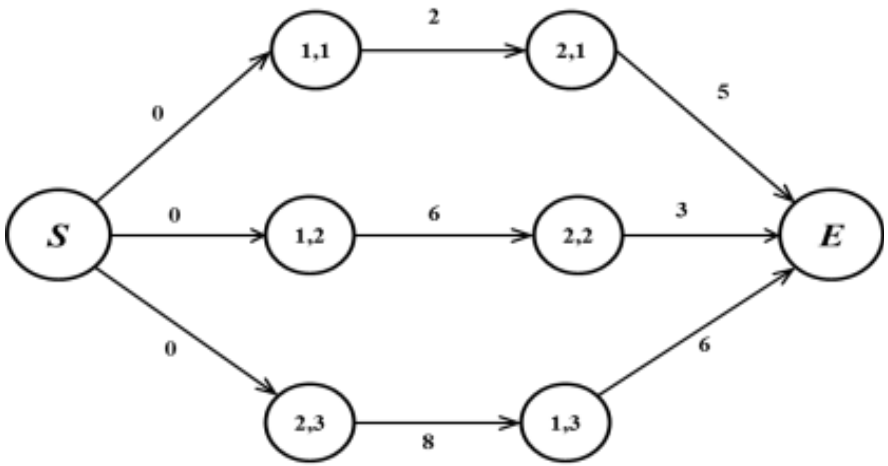
Example

Example: Total enumeration for $J_2 \,||\, C_{max}$

Job	*Sequence*	*Processing Time*
1	1,2	$p_{11} = 2$, $p_{21} = 5$
2	1,2	$p_{12} = 6$, $p_{22} = 3$
3	2,1	$p_{23} = 8$, $p_{13} = 6$

Example

Initial directed graph with conjunctive arcs:



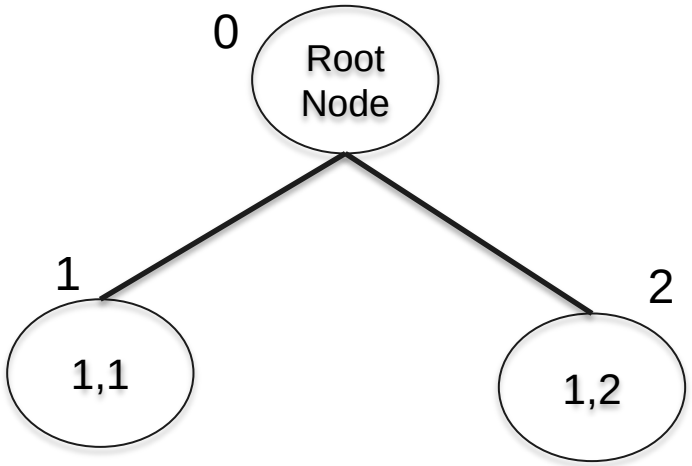
<u>Node 0 (Root Node):</u>

Ω = { (1,1) , (1,2) , (2,3) }

t(Ω) = min(0+2 , 0+6 , 0+8) = 2

i* = 1

⟹ **Ω'** = { (1,1) , (1,2) }



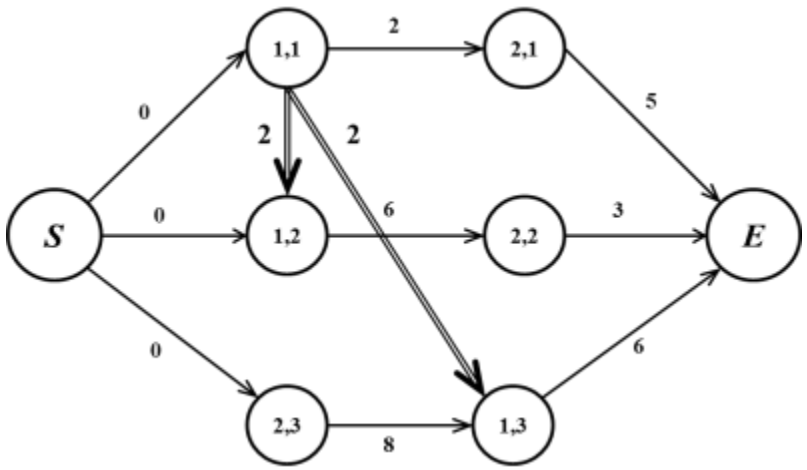
Ω = { **<u>(2,1)</u>** , (1,2) , (2,3) }

Ω = { (1,1) , **<u>(2,2)</u>** , (2,3) }

Example

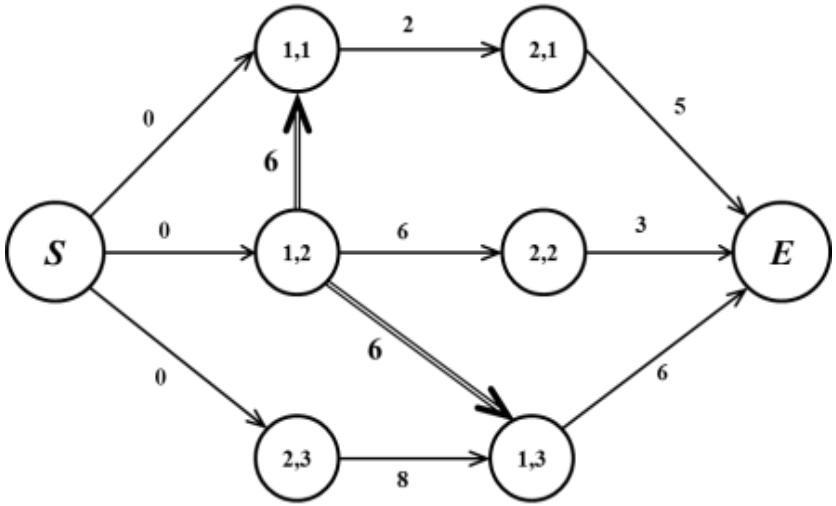
Node 1:

Ω = { (2,1) , (1,2) , (2,3) }
t(Ω) = min(2+5 , 2+6 , 0+8) = 7
i* = 2
⟹ **Ω'** = { (2,1), (2,3) }



Node 2:

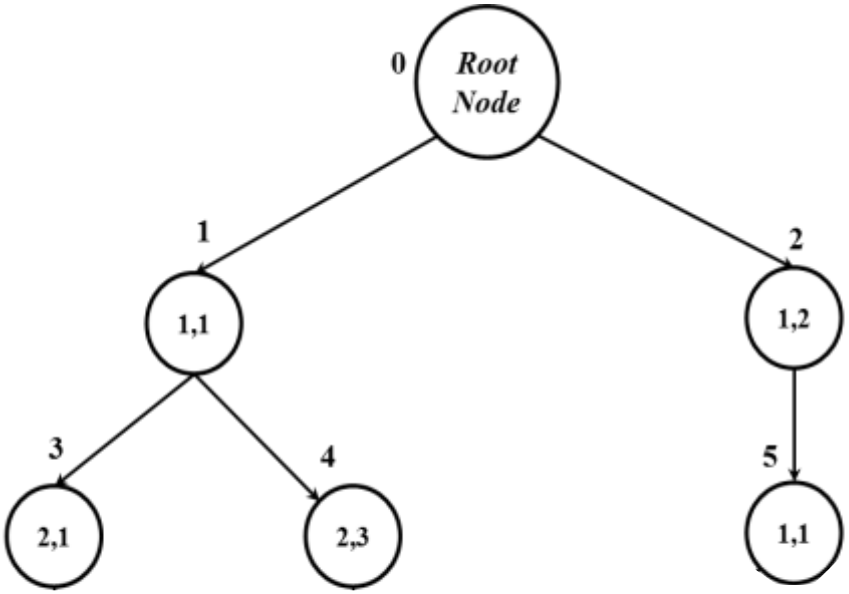
Ω = { (1,1) , (2,2) , (2,3) }
t**(Ω)** = min(6+2 , 6+3 , 0+8) = 8
i* = 1
⟹ **Ω'** = { (1,1) }



Example

Node 1:

Ω = { (2,1) , (1,2) , (2,3) }
t(Ω) = min(2+5 , 2+6 , 0+8) = 7
i* = 2
⟹ **Ω'** = { (2,1), (2,3) }



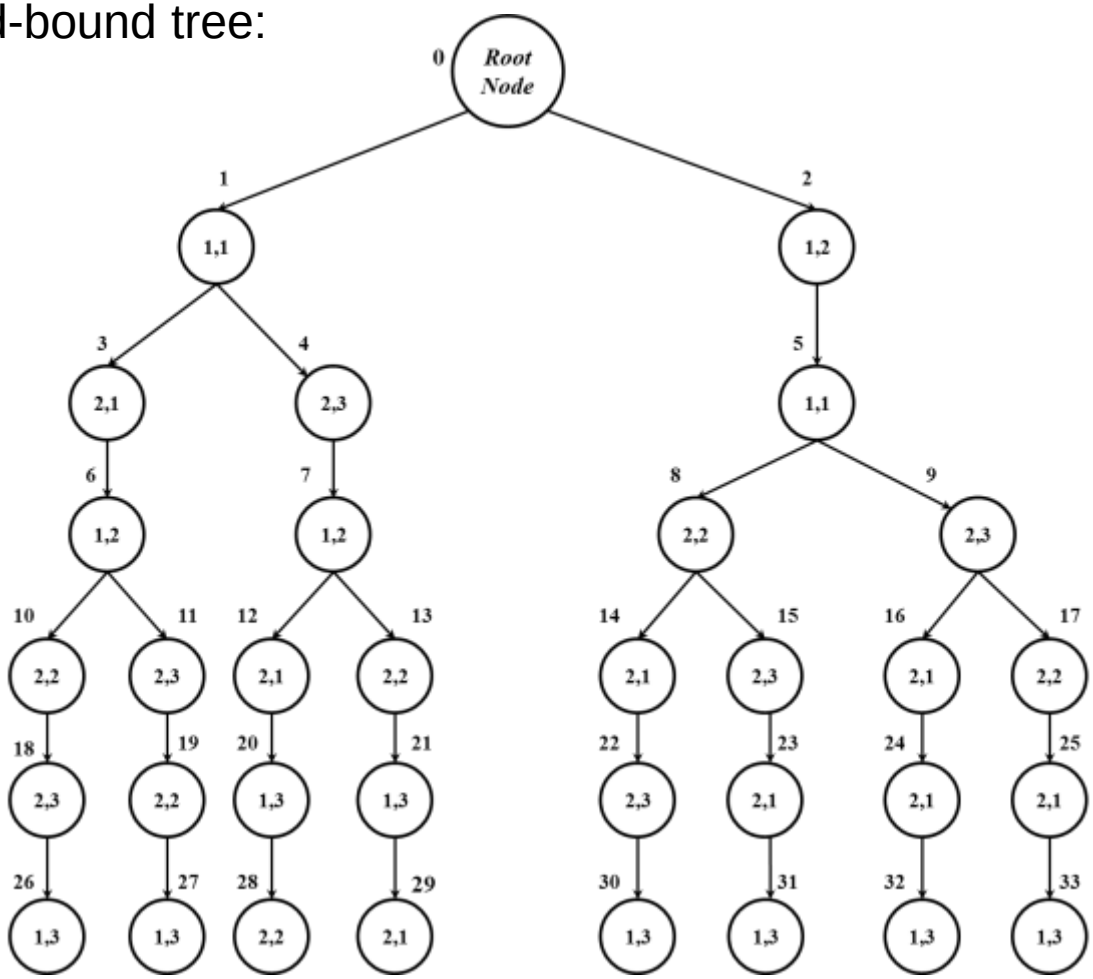
Node 2:

Ω = { (1,1) , (2,2) , (2,3) }
t**(Ω)** = min(6+2 , 6+3 , 0+8) = 8
i* = 1
⟹ **Ω'** = { (1,1) }

Finish off the tree….

Example

Complete branch-and-bound tree:



Each end node corresponds to an **active schedule**: 8 in total in this example.

Job Shop Branch-and-Bound

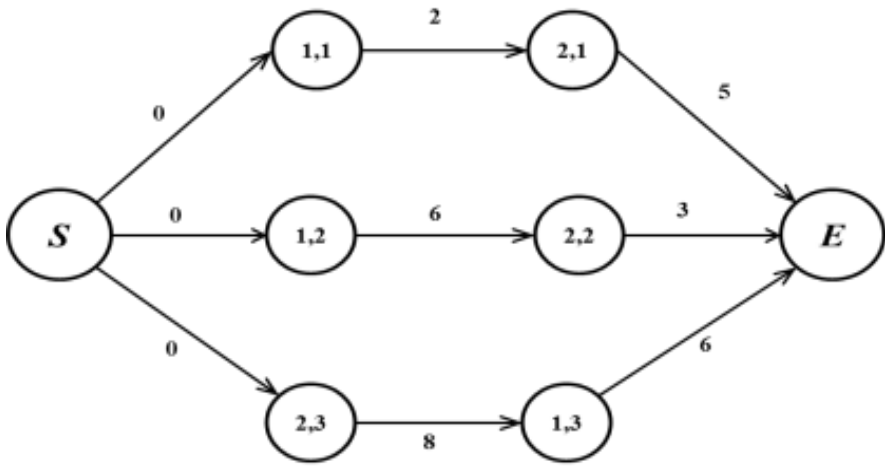
Example: Use branch-and-bound to solve the following instance of $J_2 \| C_{max}$

Job	*Sequence*	*Processing Time*
1	1,2	$p_{11} = 2$, $p_{21} = 5$
2	1,2	$p_{12} = 6$, $p_{22} = 3$
3	2,1	$p_{23} = 8$, $p_{13} = 6$

Job Shop Branch-and-Bound

A **bound** can be obtained at each node by finding the **longest path** in the network for the **partial schedule**

Example: For the **root node** we have the following directed network



The longest path is 14, thus 14 is a lower bound on C_{max}

Job Shop Branch-and-Bound

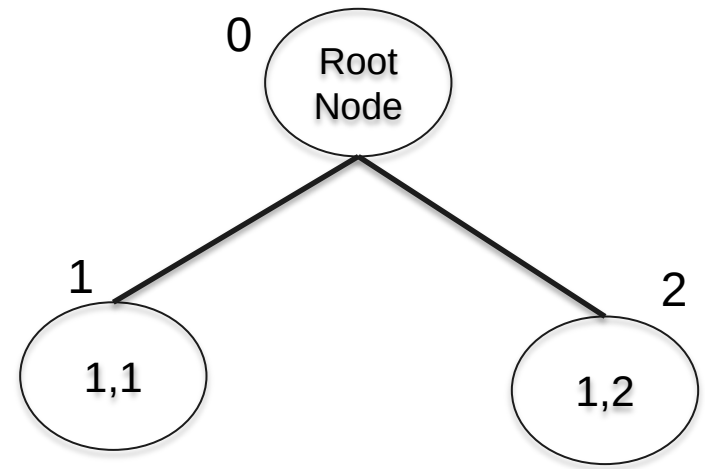
Node 0 (Root Node):

Ω = { (1,1) , (1,2) , (2,3) }

t(Ω) = min(0+2 , 0+6 , 0+8) = 2

i* = 1

⟹ **Ω'** = { (1,1) , (1,2) }



Ω = { (2,1) , (1,2) , (2,3) }

Ω = { (1,1) , (2,2) , (2,3) }

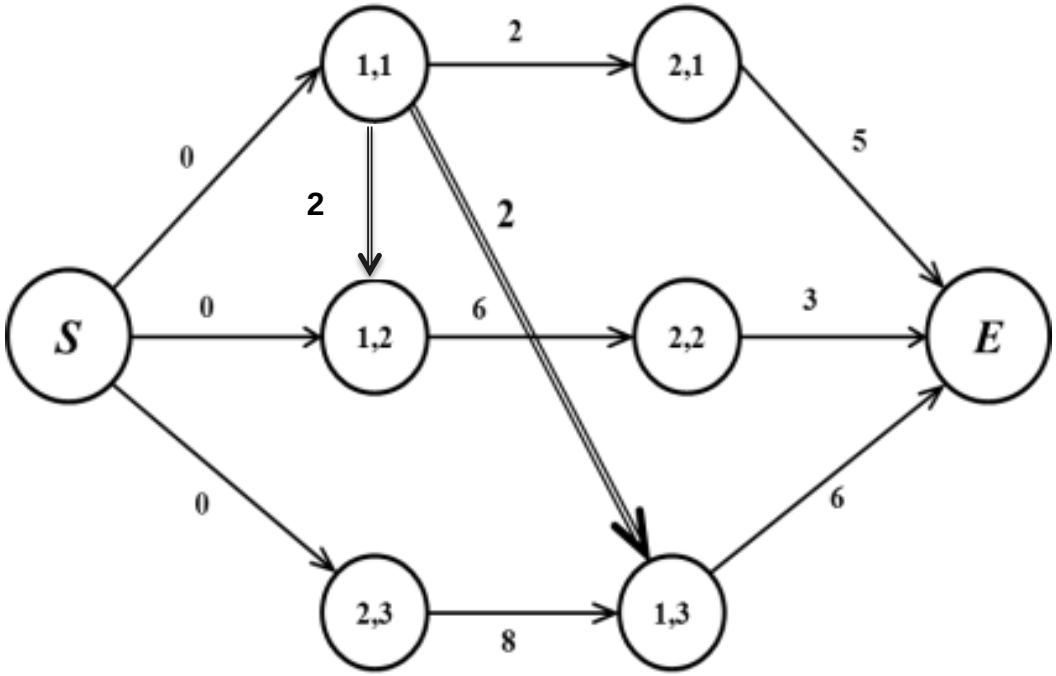
What are the lower bounds on C_{max} for nodes 1 and 2?

Job Shop Branch-and-Bound

Node 1:

Add disjunctive arcs given *Job 1* is sequenced first on machine 1

The longest path in the network is 14, thus 14 is a lower bound on C_{max} for Node 1

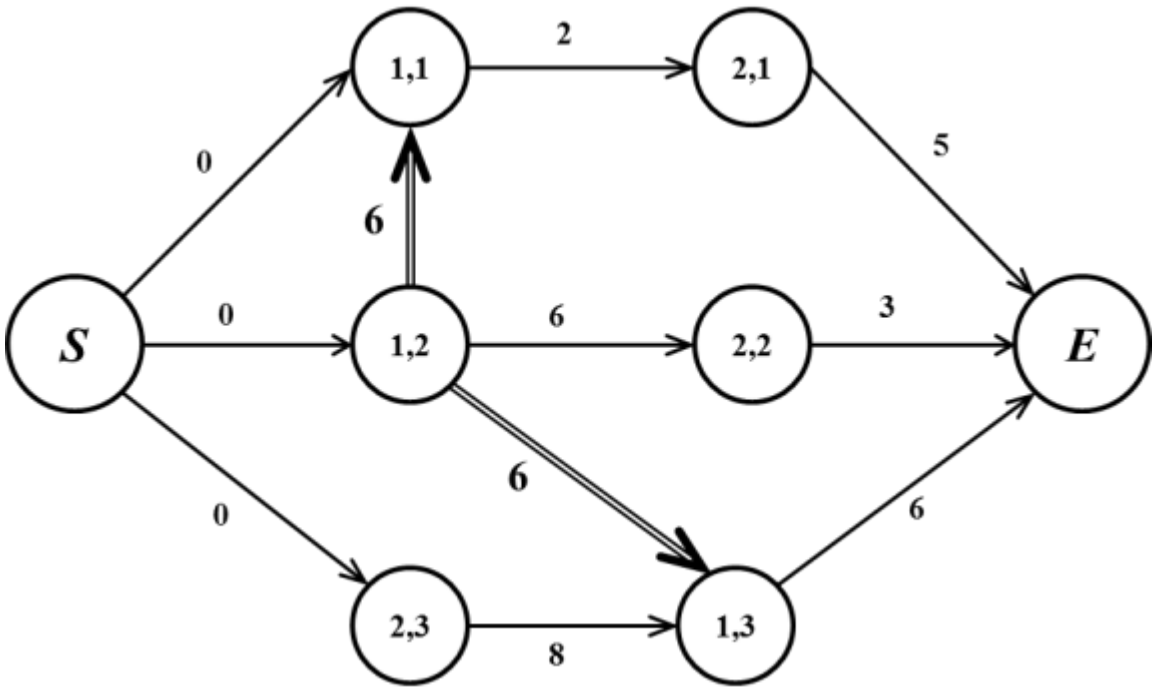


Job Shop Branch-and-Bound

Node 2:

Add disjunctive arcs given *Job 2* is sequenced first on machine 1

The longest path in the network is 14, thus 14 is a lower bound on C_{max} for Node 2

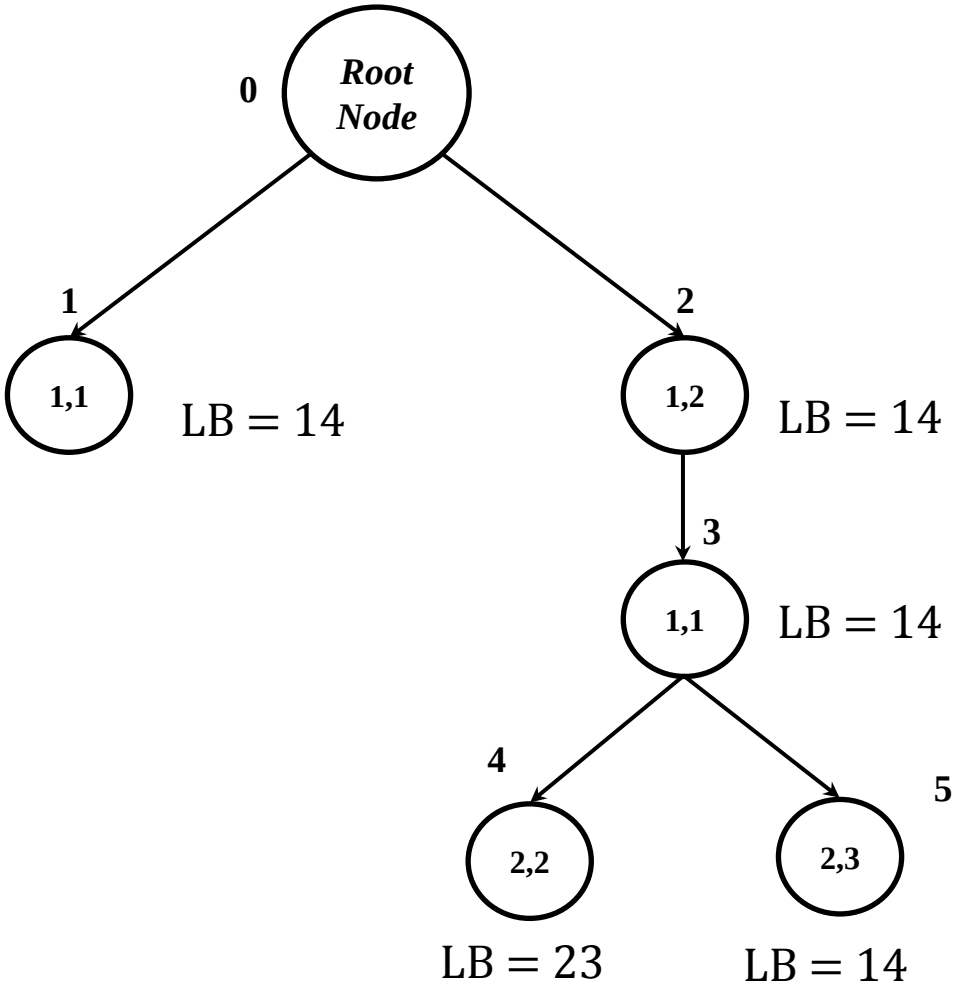


Job Shop Branch-and-Bound

Nodes 1 and 2 have the same lower bound (LB=14), so randomly select one to branch

In this example we selected Node 2

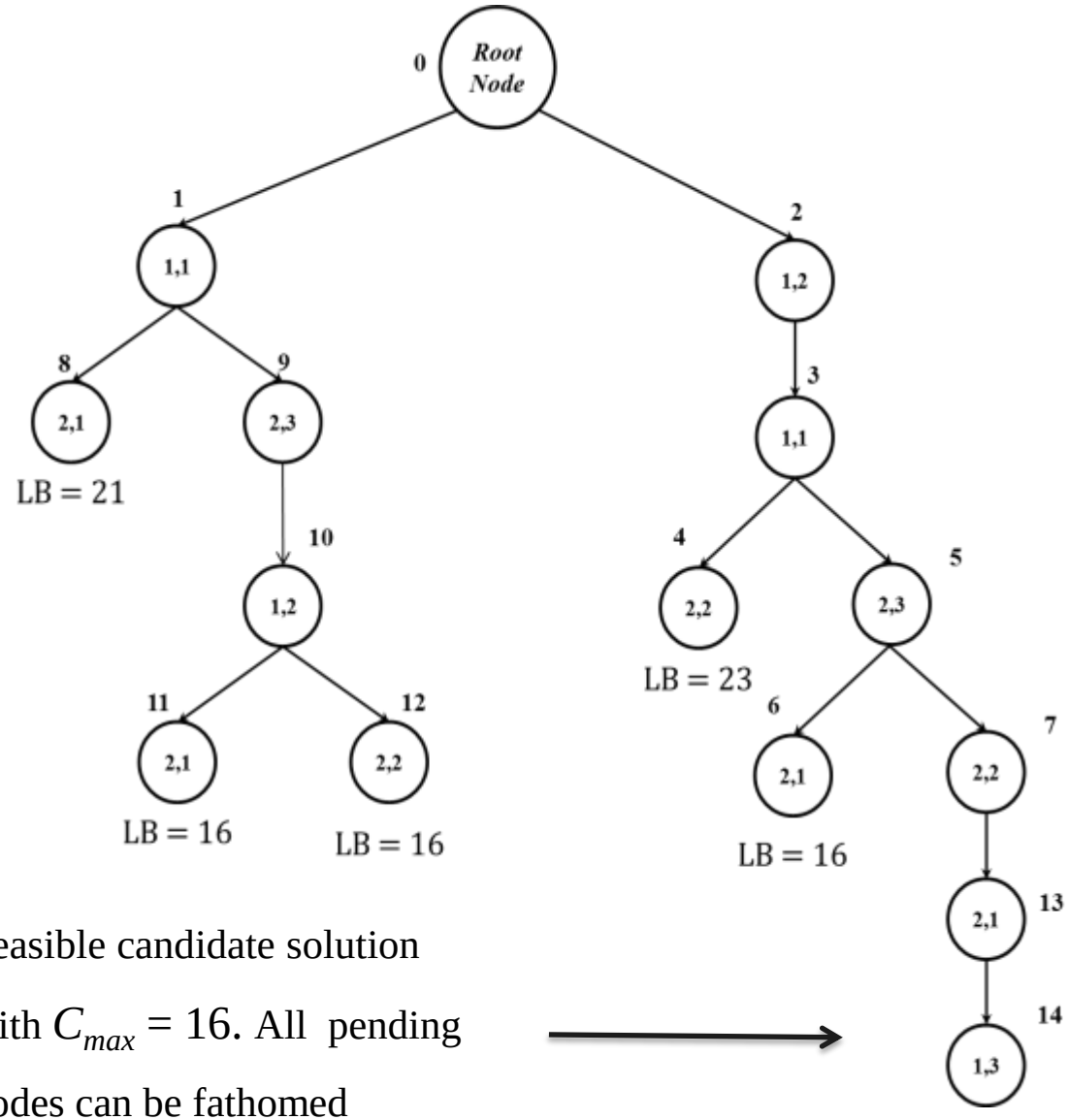
Node 3 also has LB=14 so select it to branch to Nodes 4 and 5



Find the optimal solution….

Job Shop Branch-and-Bound

Final branch-and bound tree (**note:** nodes are ordered according to the order they are generated)



Optimal Solution:

Machine	*Sequence*
1	2 , 1 , 3
2	3 , 2 , 1

$C_{max} = 16$

Feasible candidate solution with C_{max} = 16. All pending nodes can be fathomed

Reading Reminder

Today's lecture and the next lecture are based on Sections 7.1-7.3. We will not cover 7.4 or 7.5

We will skip Chapter 8 on Open Shops

Chapter 9 is important background material for stochastic scheduling.