Job Shop Scheduling Problems

- **n jobs, m machines**
- **Each job follows a predetermined route (a sequence of machines)**

Problem:

- **Operation (i,j) : Processing of job j on machine i**
- **Processing time p_{ij}**
- **Assume that jobs do not recirculate**
- **Minimize C_{max}**

visit a machine more than once

Example 1: 3 jobs, 4 machines

Jobs	Machine Sequence	Processing Times
1	1,2,3	$p_{11}=10$, $p_{21}=8$, $p_{31}=4$
2	2,1,4,3	$p_{22}=8$, $p_{12}=3$, $p_{42}=5$, $p_{32}=6$
3	1,2,4	$p_{13}=4$, $p_{23}=7$, $p_{43}=3$

Disjunctive Programming

- **A set of constraints is conjunctive, if each constraint in the set must be satisfied**
- **A set of constraints is disjunctive, if at least one of the constraints in the set must be satisfied**
- **Any integer program can be written as a disjunctive program**

Disjunctive Programming Formulation (Job Shop Problem, Minimize Makespan)

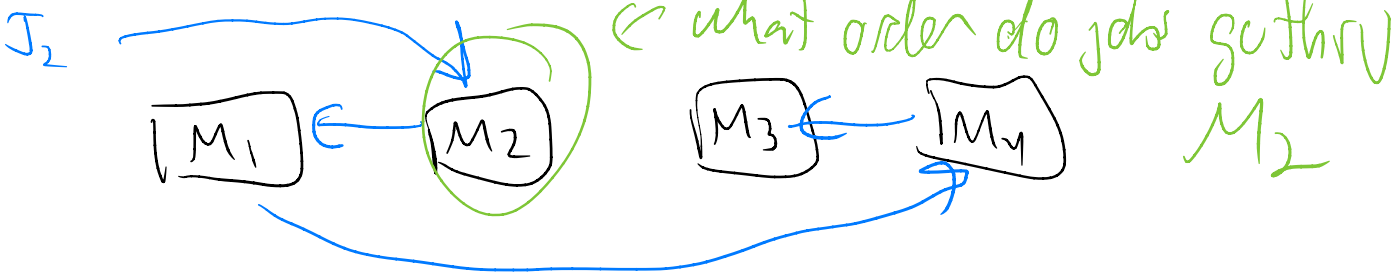
Variables:

t_{ij} = start time of job j on machine i, for all operations (i,j)

Min C_{max}

s.t

$C_{max} \geq t_{ij} + p_{ij}$	for all (i,j) ∈ N
$t_{kj} \geq t_{ij} + p_{ij}$	for all (i,j) ≺ (k,j) ∈ A
$t_{ij} \geq t_{ik} + p_{ik}$ or $t_{ik} \geq t_{ij} + p_{ij}$	for all (i,j) and (i,k) ∈ N
$t_{ij} \geq 0$	for all (i,j) ∈ N



Example 1: ***revisited,*** **3 jobs, 4 machines**

Jobs	Machine Sequence	Processing Times
1	1,2,3	p_{11}=10, p_{21}=8, p_{31}=4
2	2,1,4,3	p_{22}=8, p_{12}=3, p_{42}=5, p_{32}=6
3	1,2,4	p_{13}=4, p_{23}=7, p_{43}=3

1 123

Min C_{max}
s.t

$C_{max} \geq t_{31} + p_{31}$
$C_{max} \geq t_{32} + p_{32}$
$C_{max} \geq t_{43} + p_{43}$

n constr.

$t_{21} \geq t_{11} + p_{11}$
$t_{31} \geq t_{21} + p_{21}$
$t_{12} \geq t_{22} + p_{22}$
$t_{42} \geq t_{12} + p_{12}$
$t_{32} \geq t_{42} + p_{42}$
$t_{23} \geq t_{13} + p_{13}$
$t_{43} \geq t_{23} + p_{23}$

~ nm constr.

2<1
1<4
4<3

conjunctive

for i=1,2,3,4:

$t_{i1} \geq t_{i2} + p_{i2}$ **or** $t_{i2} \geq t_{i1} + p_{i1}$
$t_{i1} \geq t_{i3} + p_{i3}$ **or** $t_{i3} \geq t_{i1} + p_{i1}$
$t_{i2} \geq t_{i3} + p_{i3}$ **or** $t_{i3} \geq t_{i2} + p_{i2}$

~ n^2m constr.

$t_{ij} \geq 0,$ **for all (i,j) ∈ N**

$n = m = 20$

$20^3 = 8000$ constr.

IP Formulation:

Variables:

t_{ij} = start time of job j on machine i, for i = 1,2,3 and j =1,2,3

x_{ijk} = 1, if job j precedes job k on machine i,
0, otherwise **for i = 1,2,3,4**
j =1,2,3 and j < k

Min C_{max}

s.t

$$C_{max} \geq t_{31} + p_{31}$$
$$C_{max} \geq t_{32} + p_{32}$$
$$C_{max} \geq t_{33} + p_{43}$$

$$t_{21} \geq t_{11} + p_{11}$$
$$t_{31} \geq t_{21} + p_{21}$$
$$t_{12} \geq t_{22} + p_{22}$$
$$t_{42} \geq t_{12} + p_{12}$$
$$t_{32} \geq t_{42} + p_{42}$$
$$t_{23} \geq t_{13} + p_{13}$$
$$t_{43} \geq t_{23} + p_{23}$$

$$t_{i1} + p_{i1} \leq t_{i2} + M(1-x_{i12}) \quad \text{for i=1,2,3,4}$$
$$t_{i2} + p_{i2} \leq t_{i1} + M\, x_{i12}$$
$$t_{i1} + p_{i1} \leq t_{i3} + M(1-x_{i13})$$
$$t_{i3} + p_{i3} \leq t_{i1} + M\, x_{i13}$$
$$t_{i2} + p_{i2} \leq t_{i3} + M(1-x_{i23})$$
$$t_{i3} + p_{i3} \leq t_{i2} + M\, x_{i23}$$

$t_{ij} \geq 0$, for i=1,2,3,4 and j=1,2,3

$x_{ijk} \in \{0,1\}$ for i = 1,2,3,4 j =1,2,3 and j < k

Disjunctive Graph Representation:

- branch & bound
- Shifting Bottleneck heuristic

- **Directed graph G, nodes N, arc sets A and B, G=(N,A,B)**
- **There exists a node for each operation (i,j)**
- **<u>Conjunctive</u> arcs A represent routes of the jobs**

  Arc (i,j)⟶(k,j) denotes that operation (i,j) precedes (k,j)

  (defined for two operations of the same job)

- **<u>Disjunctive</u> arcs B represent sequence of jobs on a machine**

  Arc (i,j)⇢(i,k) denotes that operation (i,j) precedes (i,k)
  Arc (i,j)⇠(i,k) denotes that operation (i,k) precedes (i,j)

  Arcs in both directions exist, only one of the two is selected in a feasible schedule
  (defined for two operations on the same machine)

- **Length of an arc is the processing time of the operation from which the arc originates $(i,j) \xrightarrow{p_{ii}} (k,j)$**

- **Add a source node U and a sink node V**
- **Connect U to the <u>first</u> operation of each job by a conjunctive arc (going out of U) of length 0**
- **Connect V to the <u>last</u> operation of each job by a conjunctive arc (going into V)**

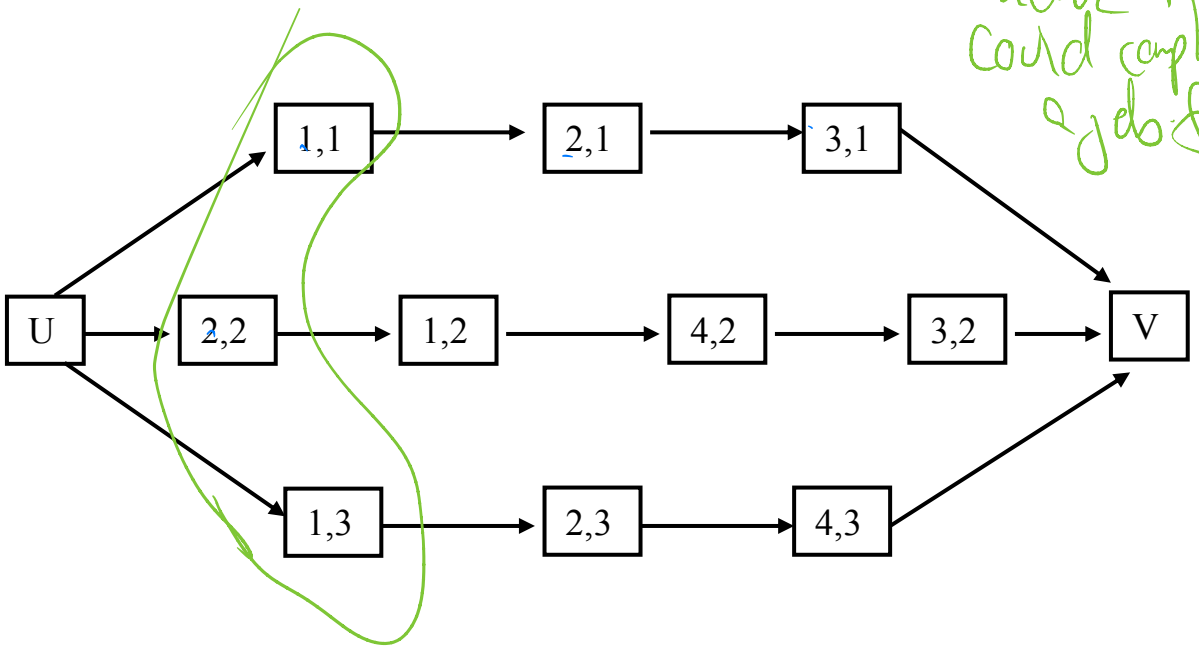
- **A *feasible schedule* corresponds to a subgraph S such that**
  - **S contains all the conjunctive arcs A**
  - **For each pair of disjunctive arcs between the same nodes, exactly one arc is contained in S**
  - **S contains no directed cycle**

orient the disj. arcs so that there are no cycles and critical path is minimized

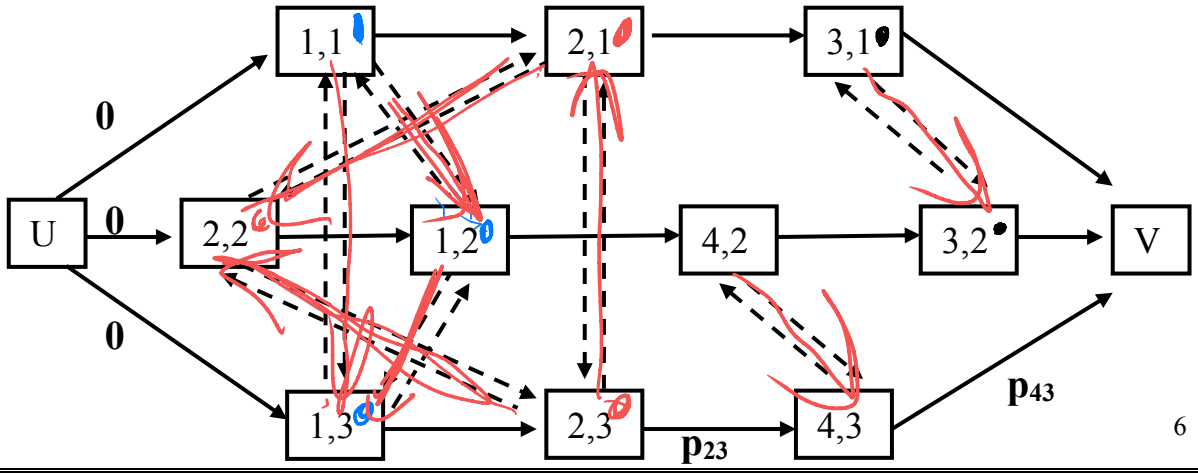
Example 1: *revisited,* 3 jobs, 4 machines

Jobs	Machine Sequence	Processing Times
1	1,2,3	$p_{11}=10$, $p_{21}=8$, $p_{31}=4$
2	2,1,4,3	$p_{22}=8$, $p_{12}=3$, $p_{42}=5$, $p_{32}=6$
3	1,2,4	$p_{13}=4$, $p_{23}=7$, $p_{43}=3$

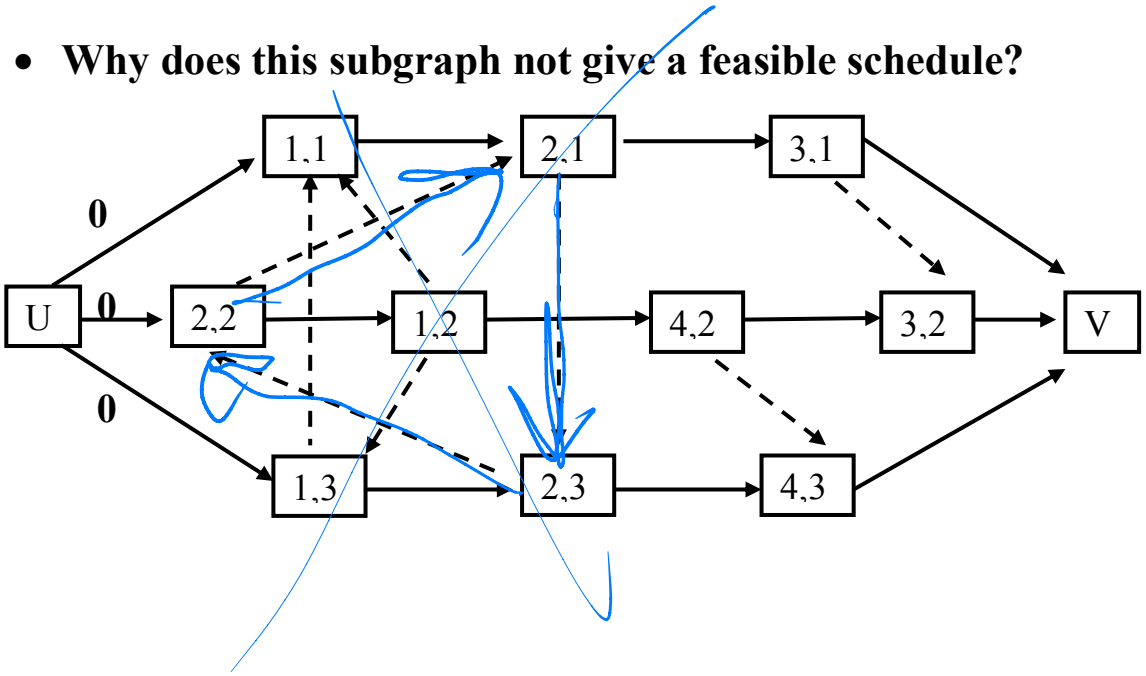
First, add the conjunctive arcs (set A)



Now, add the disjunctive arcs (set B) (dashed arcs)



- **Why does this subgraph not give a feasible schedule?**



AN ORIENTATION CORESPONDS TO A FEASIBLE SCHEDULE IF AND ONLY IF THE SUBGRAPH CONTAINS NO DIRECTED CYCLES

Makespan of a schedule is the critical path.

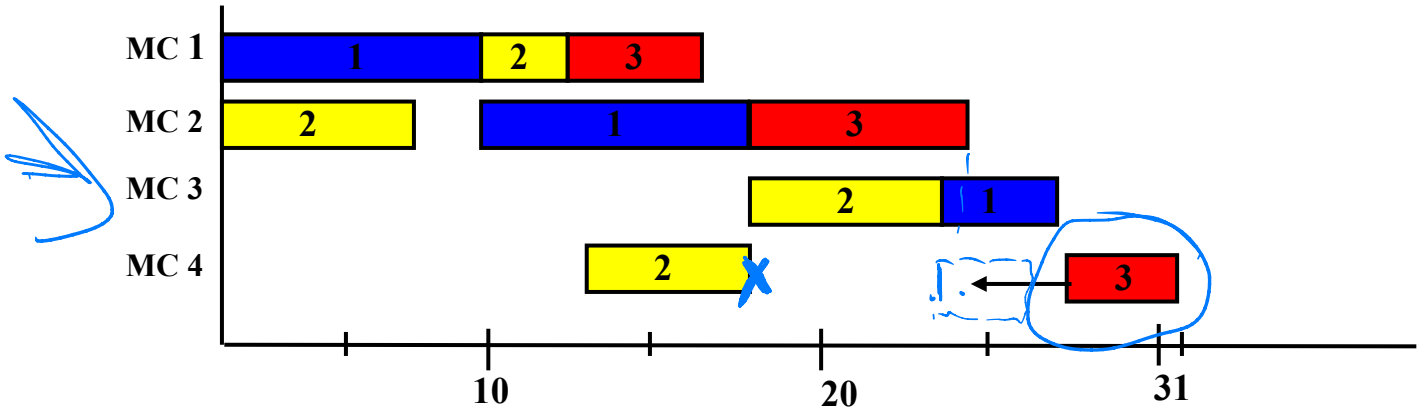
- **How can we minimize the makespan?**

  Choose the arcs so as to minimize the length of the critical path.

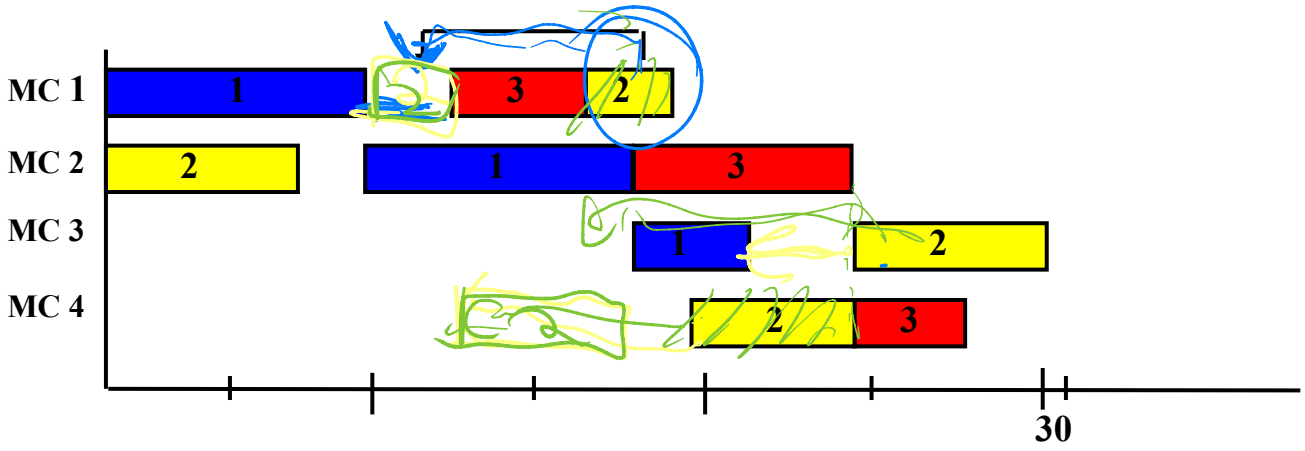
Active Schedules:

Two types of modification of a feasible schedule to obtain another feasible schedule

- **A *left shift*: move an operation left to start it earlier**
- **A *left jump*: move an operation left into an idle slot to start it earlier**

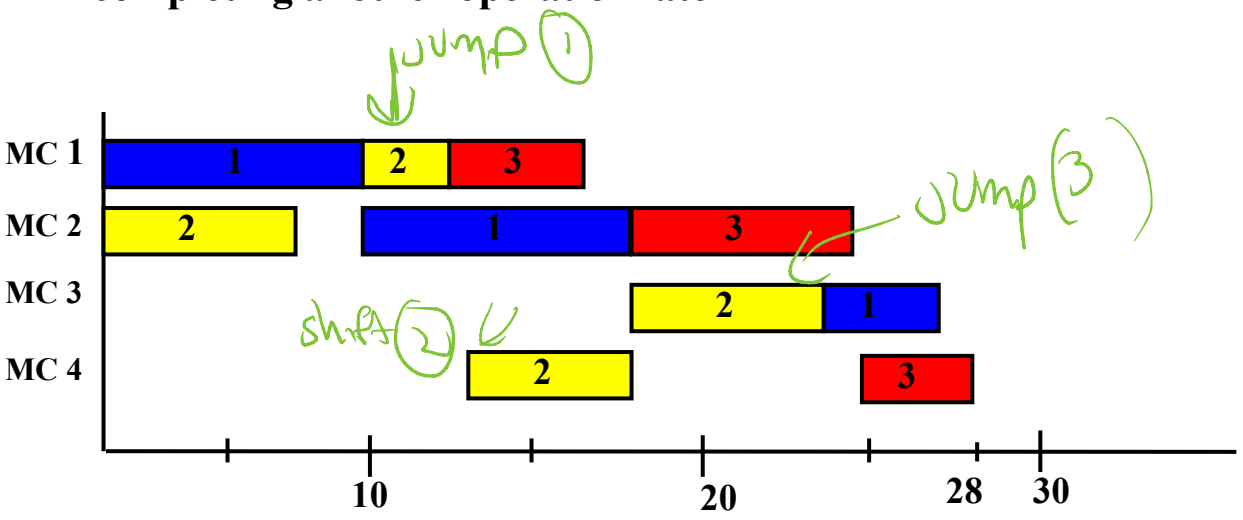


Left shift



Left jump

- **A feasible schedule is *active* if it cannot be modified by a left shift or a jump to complete an operation earlier without completing another operation later**



Active schedule

Branch and Bound for Disjunctive Programs:

- **There exists an optimal solution that is an active schedule**
- **We can restrict our search to only active schedules**

Generating All Active Schedules

- **At every level of the search tree one operation is fixed**
- **At a given level, operations that are already fixed make up a partial schedule**
- **No operation is considered until all its predecessors are scheduled**
- **If all the predecessors of an operation are scheduled, then the operation is said to be AVAILABLE**

- **Given a partial schedule, all AVAILABLE OPERATIONS can be found easily**

Notation:

AO = set of all operations whose predecessors have already been scheduled (AVAILABLE OPERATIONS)
(denoted by ⊘ in the textbook)

r_{ij} = earliest starting time of operation (i,j) in AO

- **Given a partial schedule and an available operation (i,j), how do we calculate r_{ij} ?**

<u>Algorithm OFA</u>

Step 1. (Initialization)

AO = {first operation of each job}
$r_{ij} = 0$ for all (i,j) in AO

Step 2. (Machine Selection)

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

and let i^* be the machine corresponding to the operation that minimizes $r_{ij} + p_{ij}$

Step 3. (Branching)

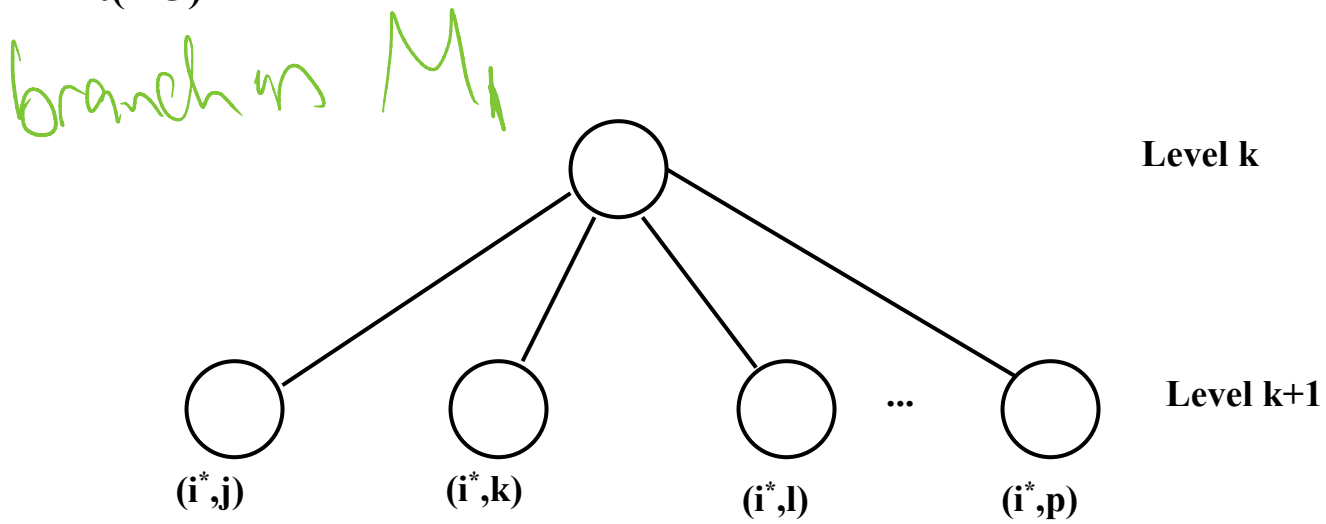
For each operation (i^*,j) on machine i^* such that $r_{i^*j} < t(AO)$,

- Construct a branch by fixing that operation as the next operation on machine i^*
- delete the operation from AO
- add the immediate follower of the operation to AO

If AO is empty, stop. Else, return to Step 2.

- The key condition that yields active schedules is the inequality $r_{i^*j} < t(AO)$ in Step 3.

- There cannot be any operation that can be completed before t(AO)



Create a branch for each available operation on machine i^* with earliest start time smaller than t(AO)

- Can you think of a lower bound that can be obtained at a node of the enumeration tree?

GIVEN A FEASIBLE SCHEDULE REPRESENTED BY A DISJUNCTIVE GRAPH, HOW DO YOU FIND THE CRITICAL PATH?

- Finding the critical path is equivalent to finding the longest path from u to v in the given *acyclic* directed graph

- We calculate a label for each node
- Let r_{ij} be the label of node (i,j) corresponding to operation (i,j)

- **Let IP_{ij} be the set of immediate predecessors of (i,j)**
  IP_{ij} contains all nodes that have an arc originating from itself and ending at node (i,j)

- **Start with $r_u = 0$ (source node has label equal to 0)**
- **Repeat**
  - **Pick a node (i,j) such that the labels of all nodes in IP_{ij} have been calculated**
  - **For every node in IP_{ij} calculate {label of the node + length of the arc from this node to (i,j)} and set r_{ij} to the maximum of these terms**

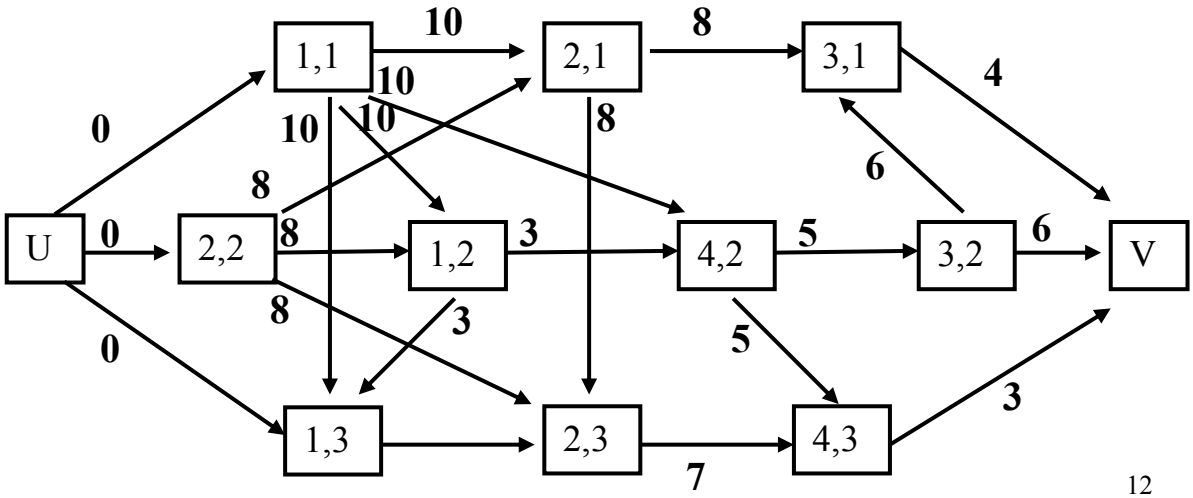
  Until the label of v, the sink node, is calculated

1. **Length of the longest u-v path equals r_v, the label of the sink node**

2. **To find the longest path itself trace back the nodes that gave the maximum term in the label calculations**

3. **The label of node (i,j), r_{ij} , equals**
   - **the length of the longest path from the source node to (i,j)**
   - **the earliest possible start time of operation (i,j)**

Example:



- **Can you think of a lower bound that can be obtained at a node of the enumeration tree?**
  LB_1 =

- **How can you improve this lower bound?**

  IDEA:

1. **WHEN WE CALCULATE THE MAKESPAN OF A PARTIAL SCHEDULE, WE ALLOW SEVERAL OPERATIONS TO BE PROCESSED ON A MACHINE AT THE SAME TIME**

2. **PICK ONE OF THE MACHINES AND ALLOW ALL OTHER MACHINES TO PROCESS MULTIPLE OPERATIONS SIMULTANEOUSLY EXCEPT THE CHOSEN MACHINE.**

CALCULATION OF THE BOUND:

1. **Consider machine i**
2. **Compute r_{ij} for all operations (i,j) on machine i**
3. **Find the longest path from (i,j) to the sink for all (i,j) on machine i; let l_{ij} be the length of this path**
4. **Set a due date for (i,j): $d_{ij} = LB_1 - l_{ij} + p_{ij}$**
5. **Solve the following single machine problem on machine i:**
   - **Operations (i,j) on machine i are the jobs**
   - **Operation (i,j) has release time r_{ij} and due date d_{ij}**
   - **No preemptions allowed**
   - **Minimize L_{max}, the maximum lateness**

6. $LB = LB_1 + L_{max}$

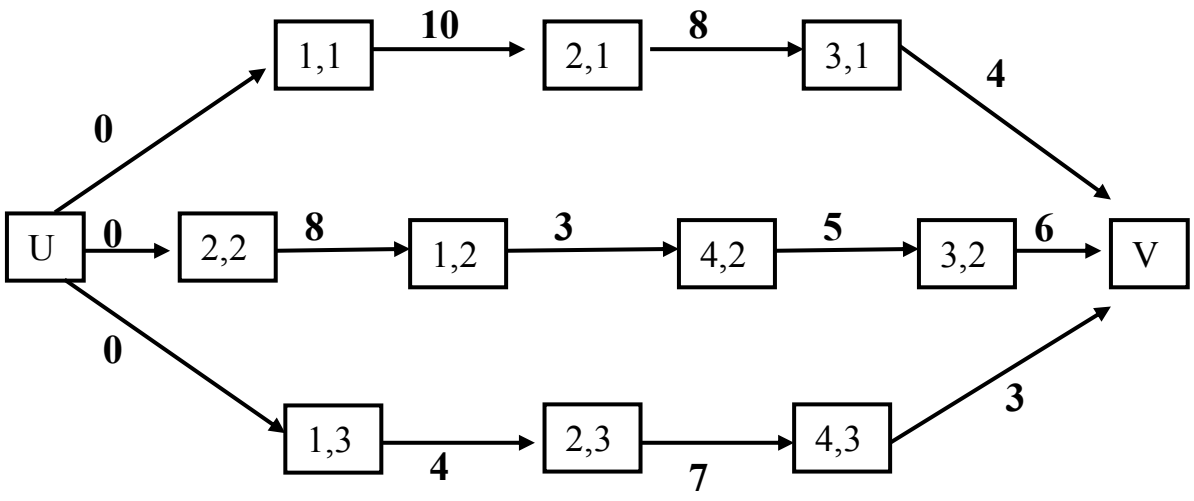
NOTE 1: THE OPTIMAL SEQUENCE OF THE SINGLE MACHINE PROBLEM IMPLIES NEW DISJUNCTIVE ARCS; ADD THE NEW ARCS TO THE CURRENT SUBGRAPH AND CALCULATE THE MAKESPAN ON THE NEW (TEMPORARY) GRAPH. THIS MAKESPAN IS THE CALCULATED LOWER BOUND

NOTE 2: THE LOWER BOUND CALCULATION CAN BE REPEATED FOR ALL MACHINES SO THAT THE LARGEST LOWER BOUND IS KEPT

BRANCH AND BOUND

Example 1: ***revisited,*** **3 jobs, 4 machines**

- **Start with a subgraph consisting of only the conjunctive arcs**
- **Calculate the makespan of this graph**



Makespan =

- **Apply algorithm OFA to branch and generate all nodes at level 1**

Step1. Initialize

$AO = \{(1,1), (2,2), (1,3)\}$
$r_{ij} = 0$ for all (i,j) in AO

Step 2. (Machine Selection)

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

$t(AO) = \min \{ 0+10, 0+8, 0+4\} = 4$

and let i^* be the machine corresponding to the operation that minimizes $r_{ij} + p_{ij}$

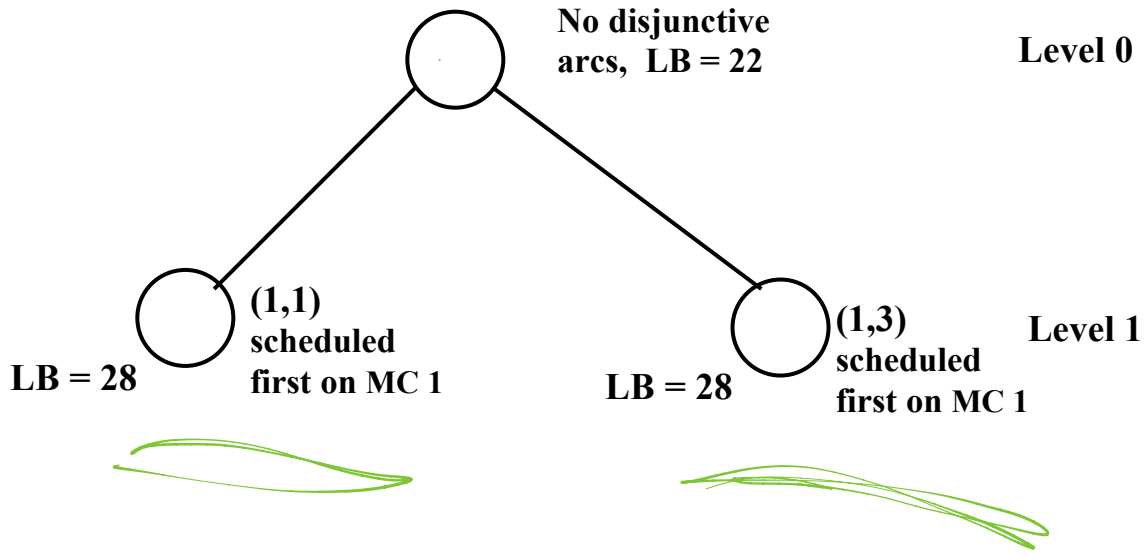
$i^* = 1$

Step 3. (Branching)

For each operation (i^*,j) on machine i^* such that $r_{i^*j} < t(AO)$,

- Construct a branch by fixing that operation as the next operation on machine i^*
- delete the operation from AO
- add the immediate follower of the operation to AO

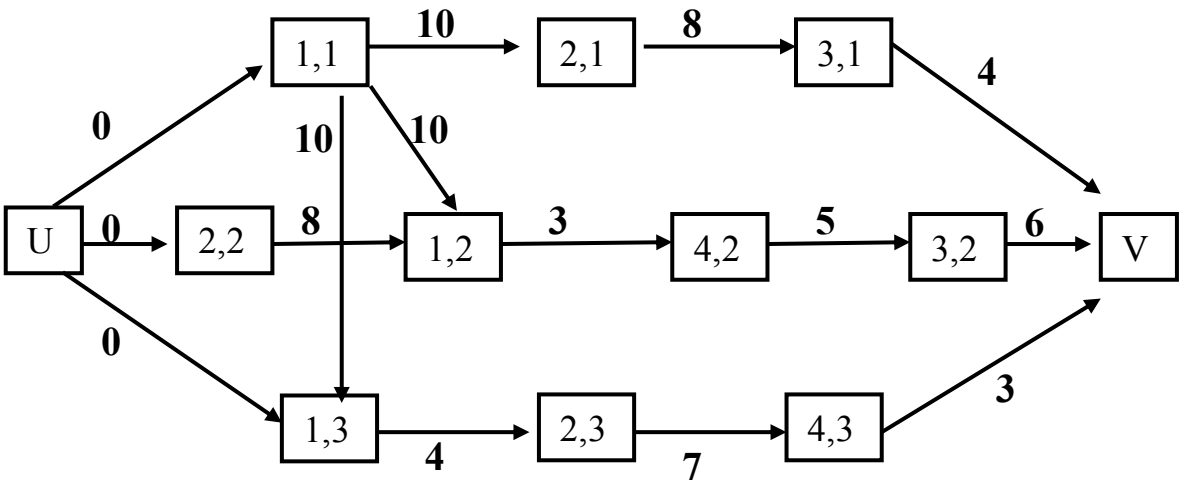
Branch on operations (1,1) and (1,3)



- **If (1,1) is scheduled first, then two disjunctive *arcs going out of* (1,1) are added to the graph**

[NEXT, OBTAIN LOWER BOUNDS]

- **Calculate the makespan on the new graph**



$LB_1 = 24$

- **Improve this lower bound**

CALCULATION OF THE BOUND AT (1,1):

1. **Consider machine 1**
2. **Compute r_{11} , r_{12}, and r_{13}**
3. **Find the longest path from (1,j) to the sink for all (1,j) on machine 1; find l_{11} , l_{12}, and l_{13}**
4. **Set a due date for (i,j): $d_{ij} = LB_1 - l_{ij} + p_{ij}$,**
5. **Solve the following single machine problem on machine 1:**

Jobs	1	2	3
p_{1j}	10	3	4
r_{1j}			

d_{1j}

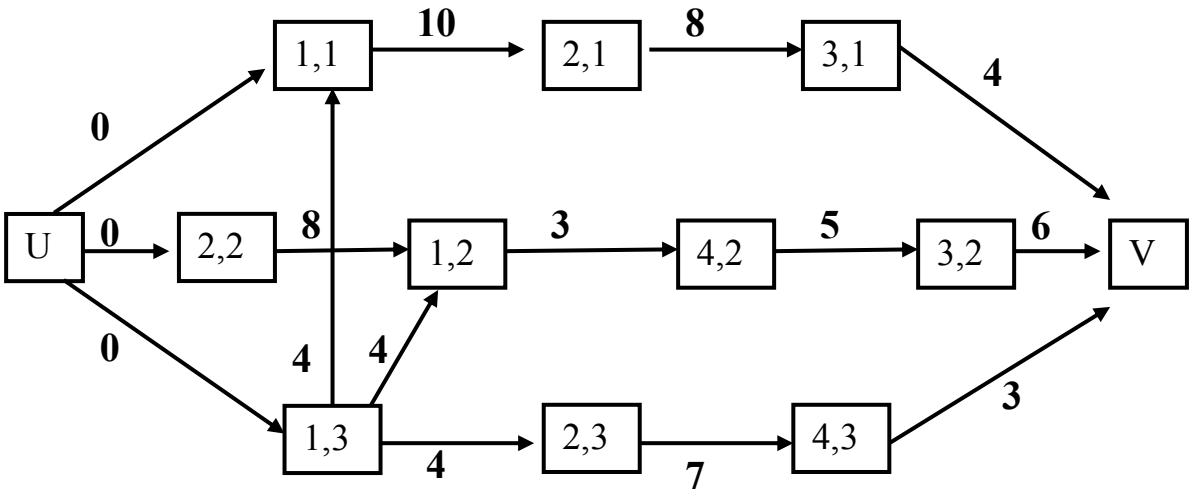
Minimize L_{max}, the maximum lateness

The optimal sequence = 1-2-3 and $L_{max} = 3$
$LB = LB_1 + L_{max} = 24 + 3 = 27$

- **Repeat the LB calculation for machine 2**
  The optimal sequence = 2-1-3 and $L_{max} = 4$
  $LB = LB_1 + L_{max} = 24 + 4 = 28$ **KEEP THIS BOUND!**

- **Repeat the LB calculation for machines 3 and 4, best LB = 28**

- **If (1,3) is scheduled first, then two disjunctive *arcs going out of* (1,3) are added to the graph**



Makespan = LB_1 =

- **Best lower bound obtained by solving single machine problems is 28**

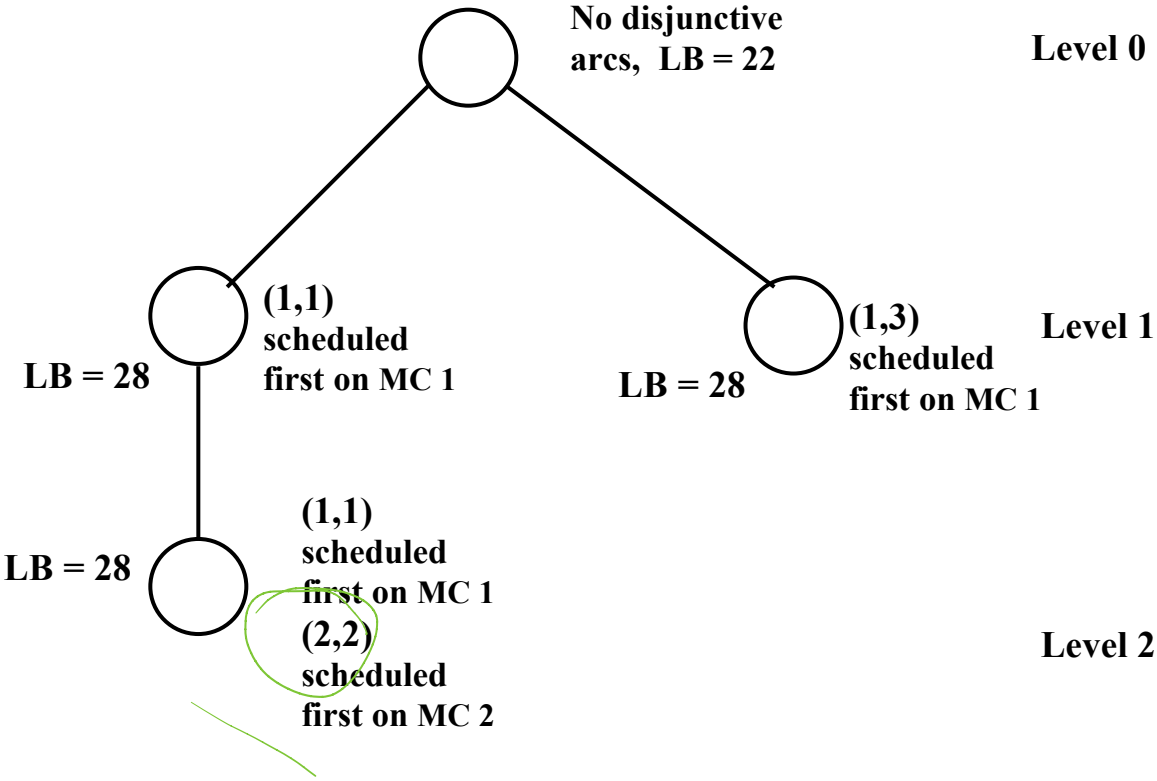
- **Branch from node (1,1) at Level 1; apply algorithm OFA**

AO = {(2,1), (2,2), (1,3)}
$r_{22} = 0$, $r_{21} = 10$, $r_{13} = 10$

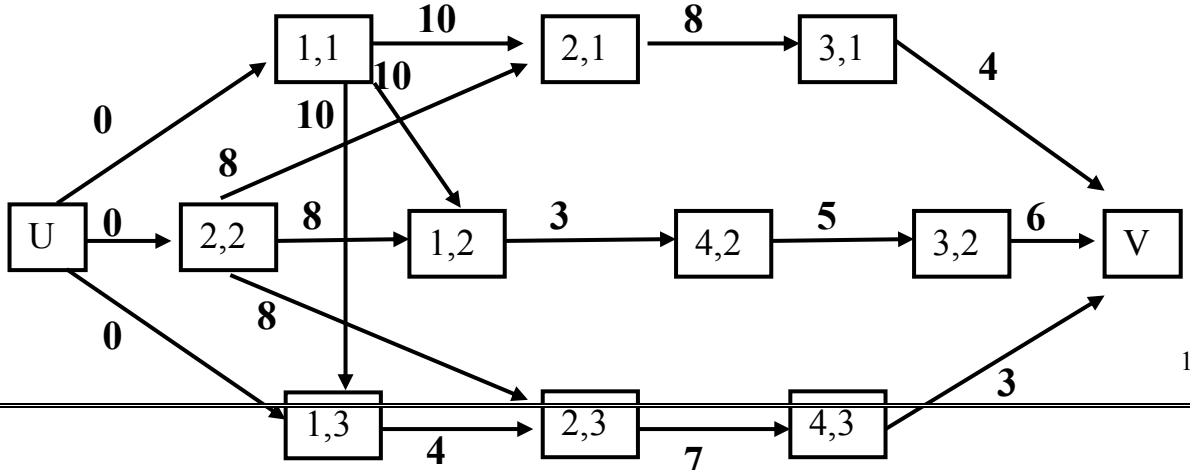
Calculate $t(AO) = \min_{(i,j) \text{ in } AO}\{ r_{ij} + p_{ij}\}$

$t(AO) = \min \{ 0+8, 10+8, 10+4\} = 8$ and $i^* = 2$

Branch on operation (2,2)



- **Process the currently only node at level 2**
- **The four disjunctive *arcs going out of* (1,1) and (2,2) are added to the original graph**



LB_1 =

L_{max} = 0 for all single machine problems, so LB = 28

If all the unprocessed nodes in the branch and bound tree are processed, the following optimal solution with makespan 28 is obtained:

Machine	Job Sequence
1	1,3,2 or (1,2,3)
2	2,1,3
3	1,2
4	2,3