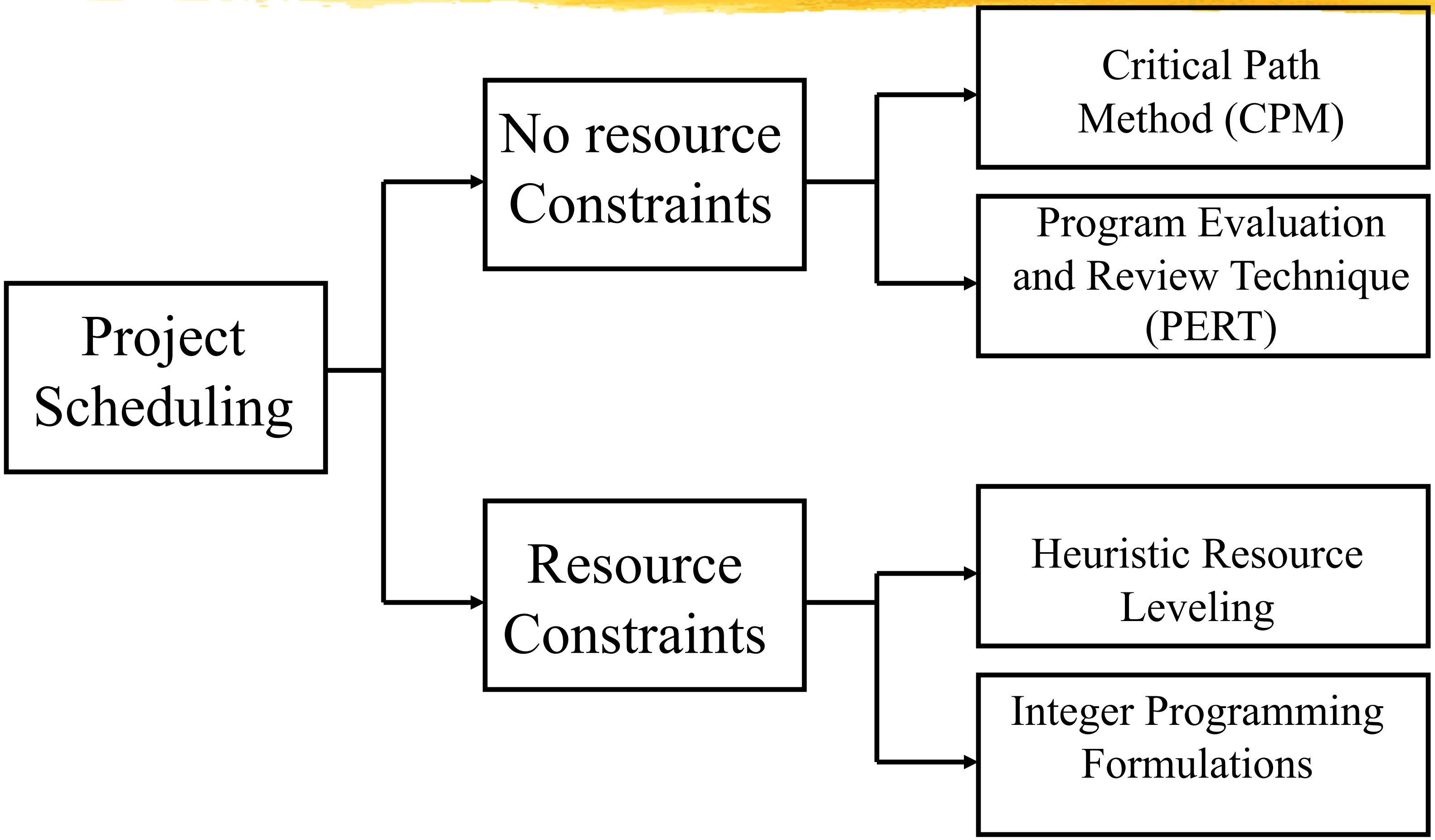
Basic Project Scheduling

Overview


Project Scheduling
No resource Constraints
Resource Constraints
Critical Path Method (CPM)
Program Evaluation and Review Technique (PERT)
Heuristic Resource Leveling
Integer Programming Formulations

Planning a Concert

Task		Predecessors
A	Plan concert	-
B	Advertise	A
C	Sell tickets	A
D	Hold concert	B, C

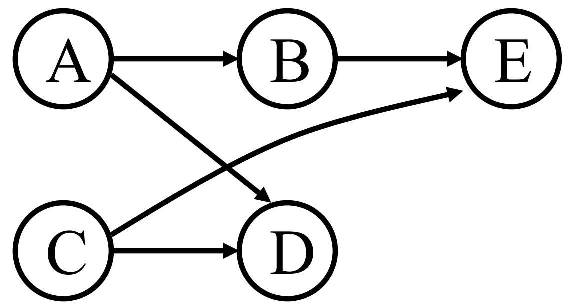
Changing a Tire

Task		Predecessors
A	Remove flat tire from wheel	-
B	Repair puncture on flat tire	A
C	Remove spare from trunk	-
D	Put spare on wheel	A, C
E	Place repaired tire in trunk	B, C

Job on Node Network

- Concert planning

Changing a tire



Critical Path Method (CPM)

- Think of unlimited machines in parallel
- … and n jobs with precedence constraints
- Processing times p_j as before

~~= average load~~

= longest path

- Objective to minimize makespan

Critical Path Method

- Forward procedure:
  - Starting at time zero, calculate the **earliest** each job can be started
  - The completion time of the last job is the makespan
- Backward procedure
  - Starting at time equal to the makespan, calculate the **latest** each job can be started so that this makespan is realized

Forward Procedure

Step 1:

Set at time $t = 0$ for all jobs j with no predecessors,

$S_j' = 0$ and set $C_j' = p_j$.

Step 2:

Compute for each job j

$$S_j' = \max_{\text{all } k \to j} C_k',$$

$$C_j' = S_j' + p_j.$$

Step 3:

The optimal makespan is $C_{\max} = \max\{C_1', C_2', \ldots, C_n'\}$

STOP

Backward Procedure

Step 1:

Set at time $t = C_{max}$ for all jobs j with no successors, $C_j'' = C_{max}$ and set $S_j'' = C_{max} - p_j$.

Step 2:

Compute for each job j

$$C_j'' = \min_{k \to \text{all } j} S_k'',$$

$S_j'' = C_j'' - p_j$.

Step 3:

Verify that $$0 = \min\{S_1'', \ldots, S_n''\}.$$

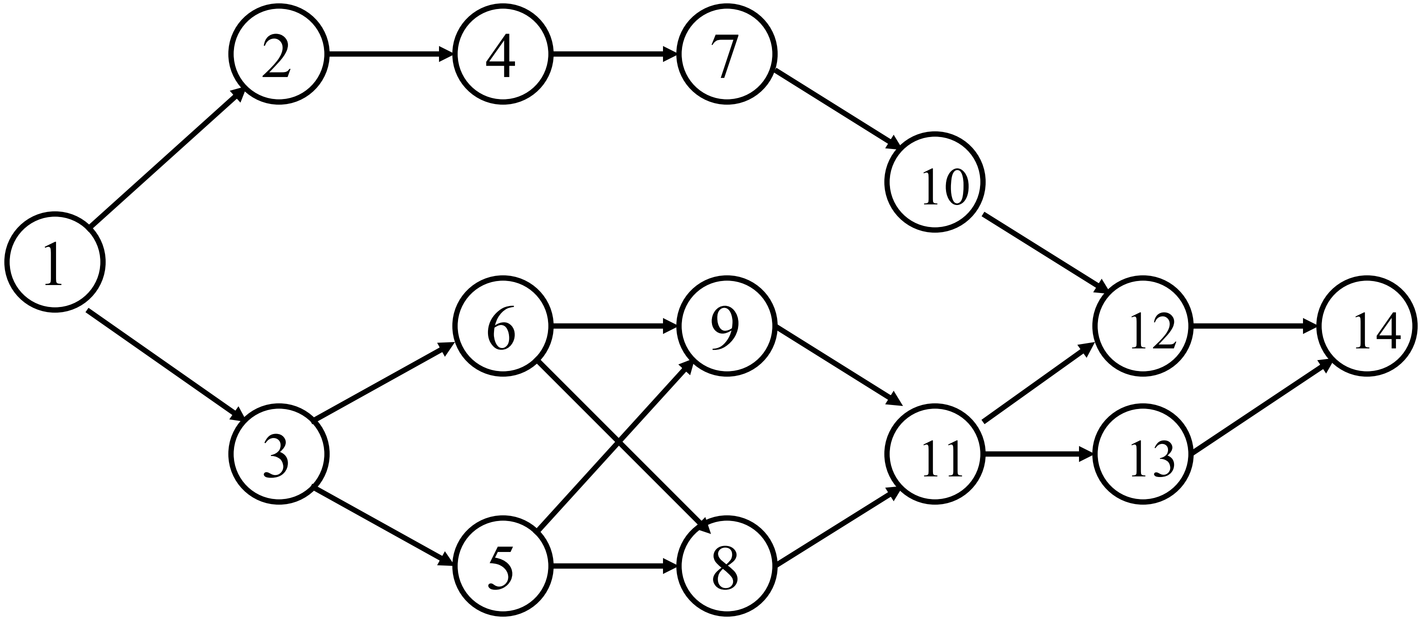
STOP

Comments

- The forward procedure gives the earliest possible completion time for each job
- The backwards procedures gives the latest possible completion time for each job
- If these are equal the job is a **critical job**.
- If these are different the job is a **slack job**, and the difference is the **float**.
- A **critical path** is a chain of jobs starting at time 0 and ending at C_{max}.

Example

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

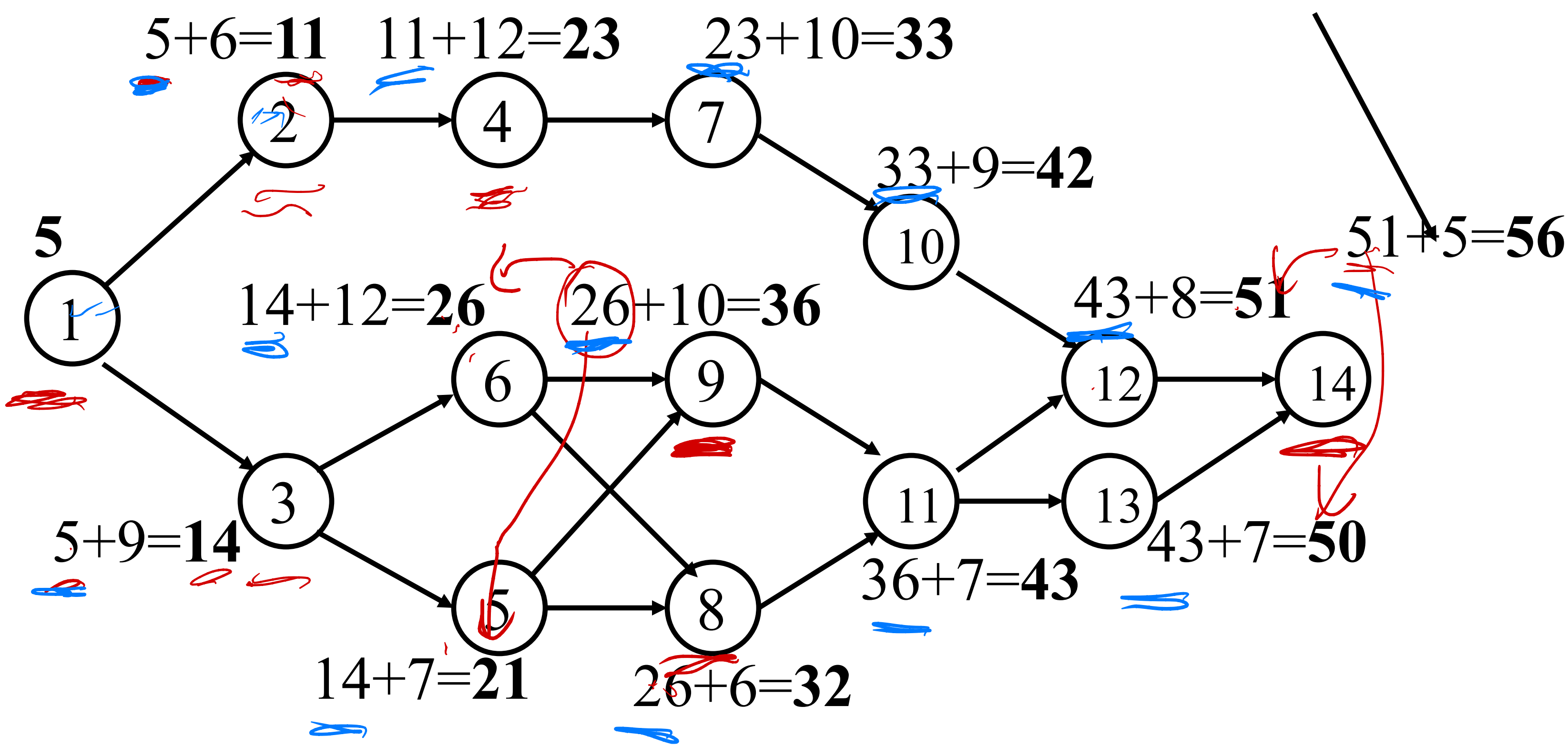


Forward Procedure

earliest Completion time

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

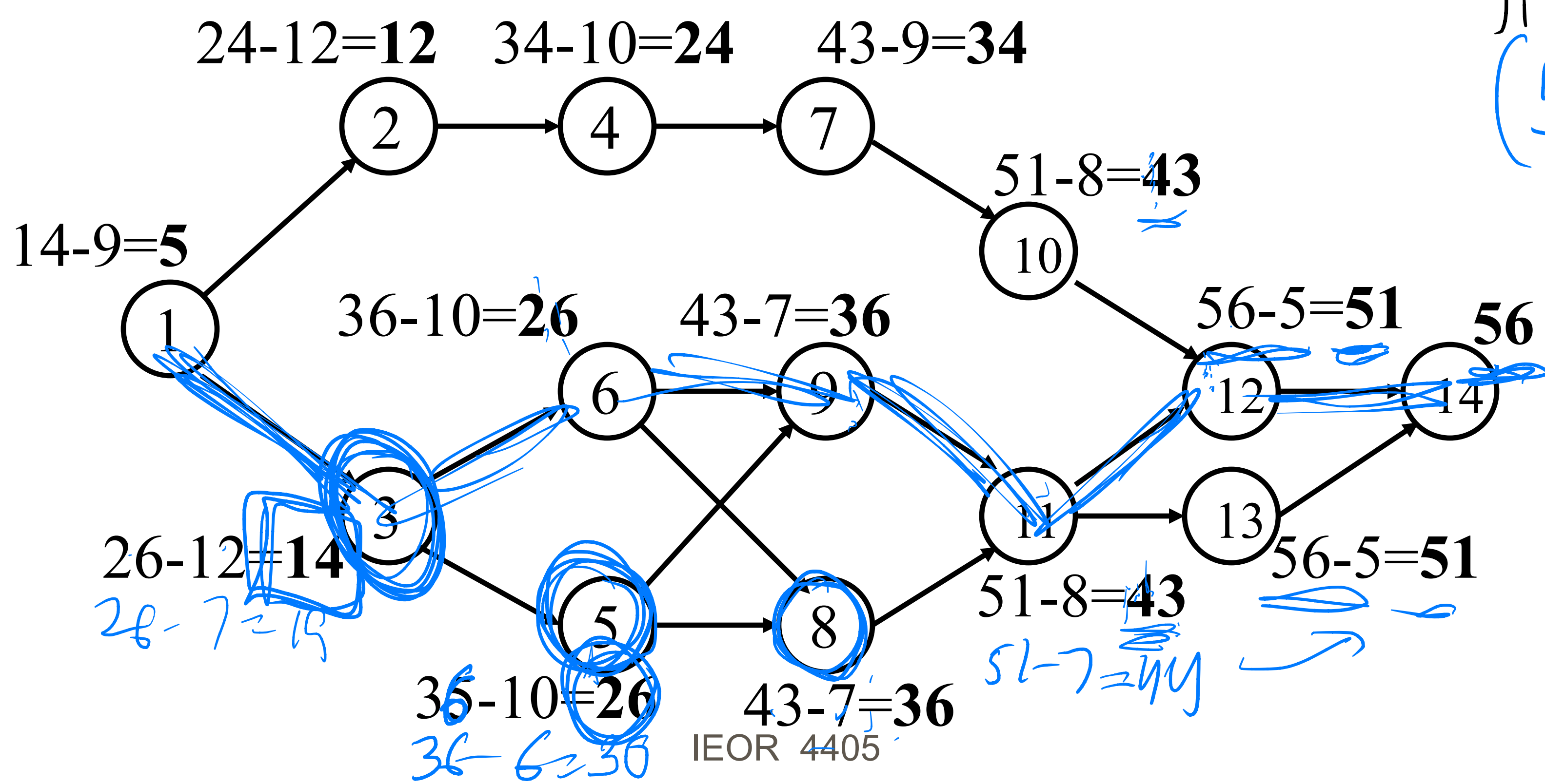
$C_{\max} = 56$



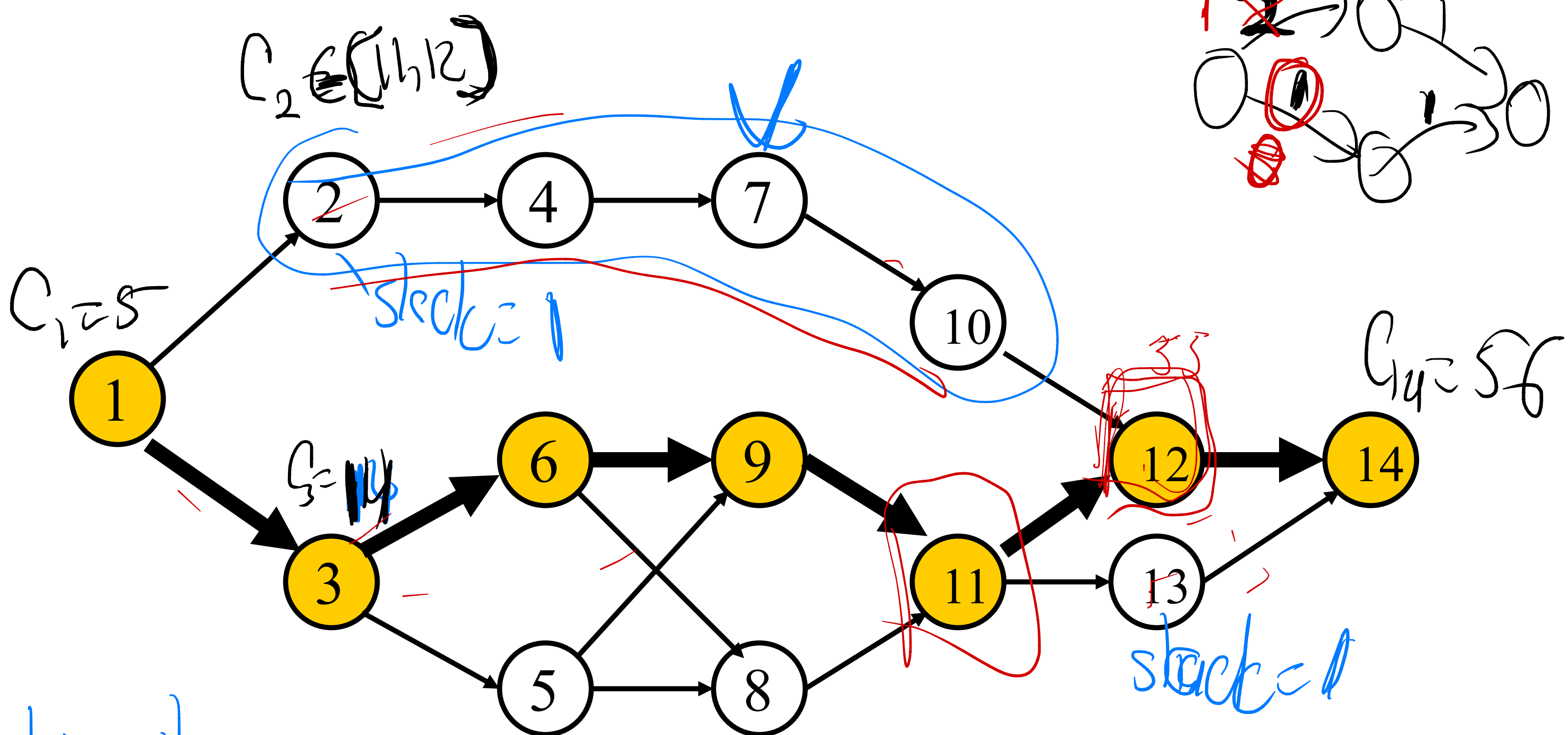
Backwards Procedure

What is the latest a job can complete, with the project still finish on time (56)

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



Critical Path



what is the min amount spent to reduce C_{max} to 55

Variable Processing Times

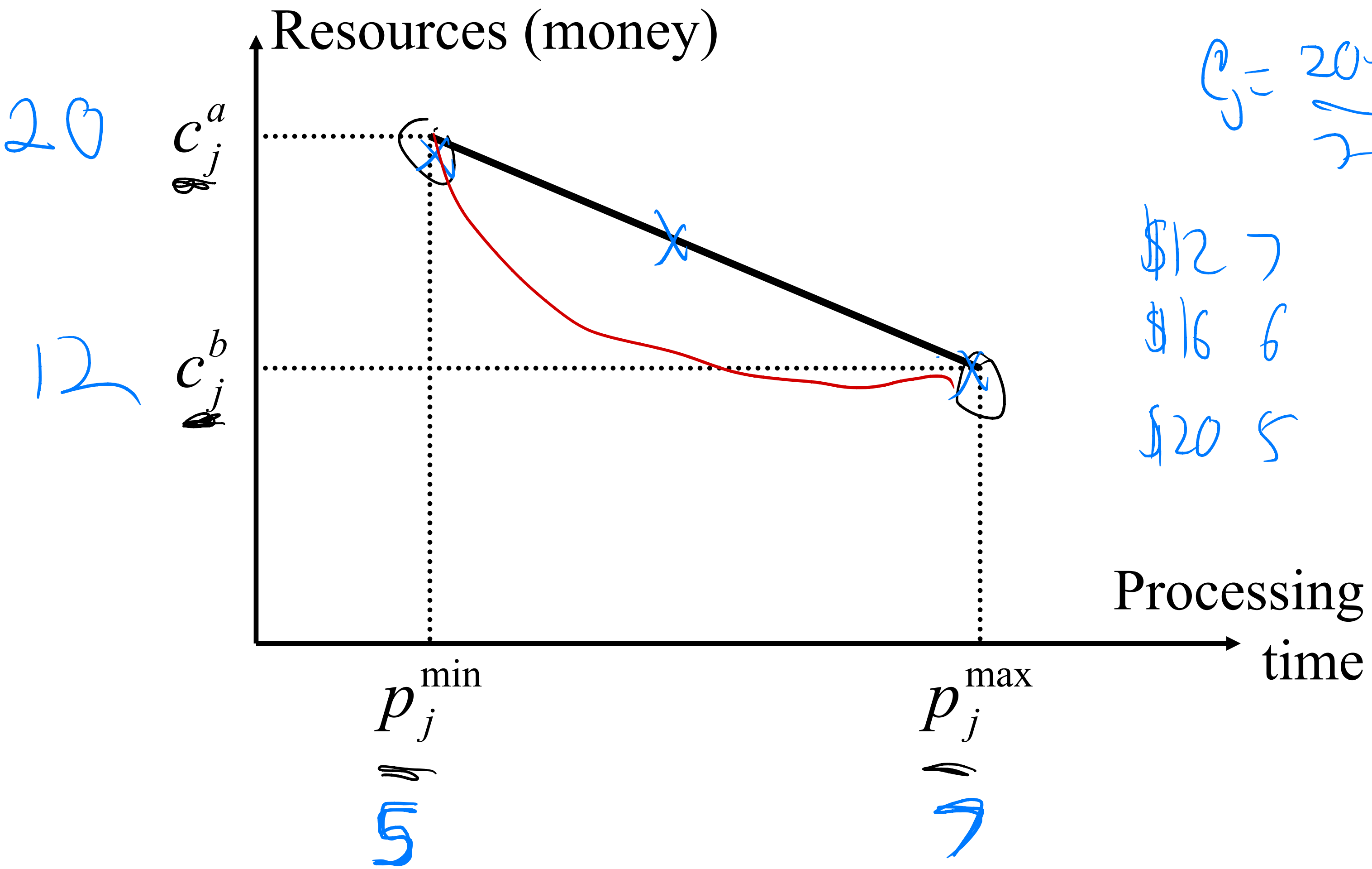
Time/Cost Trade-Offs

General

- Assumed the processing times were fixed
- More money ⇒ shorter processing time
- Start with linear costs specific model
- Processing time $p_j^{\min} \leq p_j \leq p_j^{\max}$
- Marginal cost

$$c_j = \frac{c_j^a - c_j^b}{p_j^{\max} - p_j^{\min}}$$

Linear Costs



$c_j = \frac{20-12}{7-5} = 4$

\$12 7

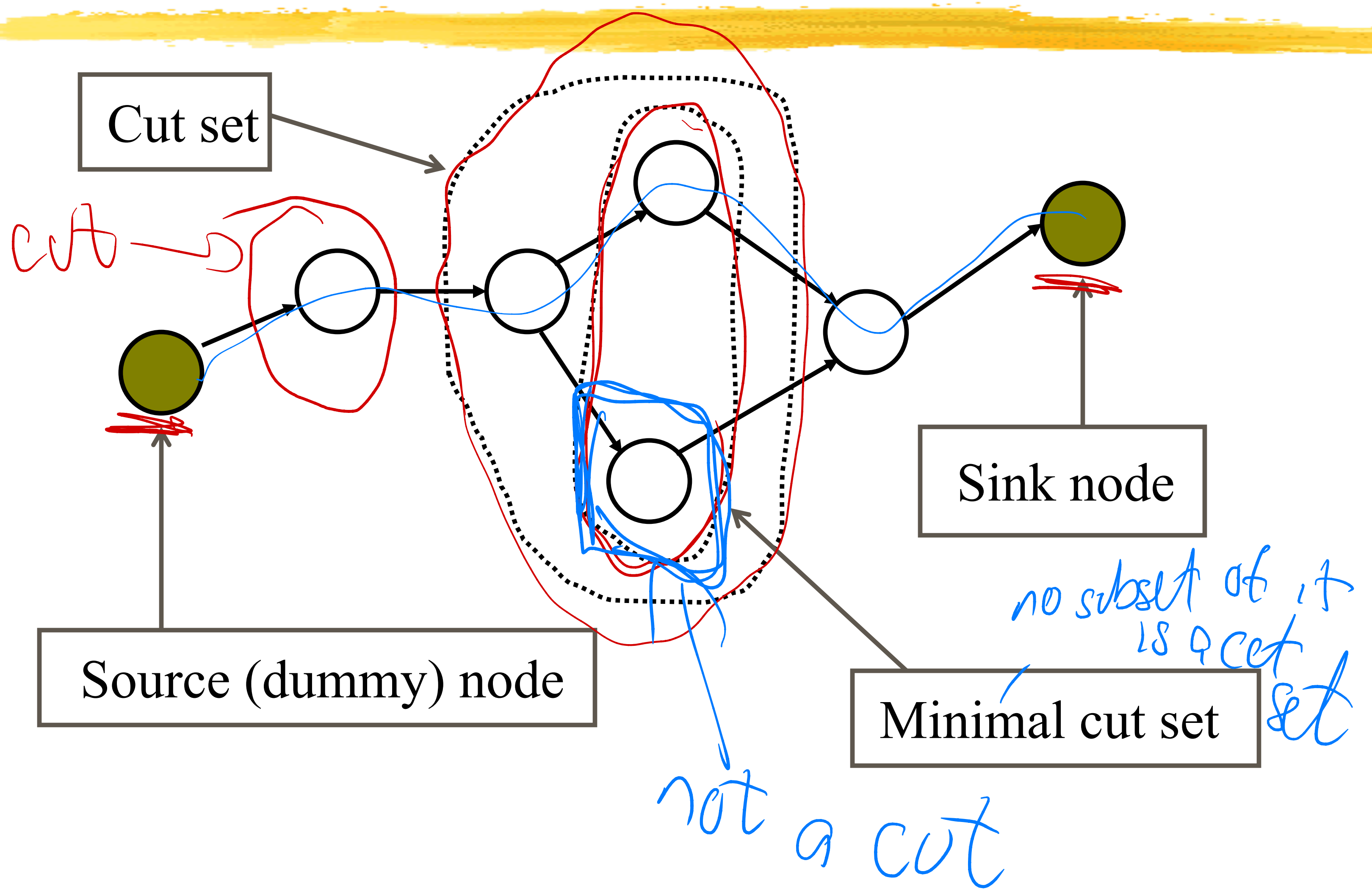
\$16 6

\$20 5

Solution Methods

- Objective: minimum cost of project
- Time/Cost Trade-Off Heuristic
  - Good schedules
  - Works also for non-linear costs
- Linear programming formulation
  - Optimal schedules
  - Non-linear version not easily solved

Sources, Sinks, and Cuts



Time/Cost Trade-Off Heuristic

Step 1:

Set all processing times at their maximum

(spend min amt of money)

$$p_j = p_j^{\max}$$

Determine all critical paths with these processing times

Construct the graph G_{cp} of critical paths

Continue to Step 2

Time/Cost Trade-Off Heuristic

Step 2:

Minimal cut set in G_{cp}

Determine all minimum cut sets in G_{cp}

IS

Consider those sets where all processing times are larger than their minimum

a set of nodes that we can reduce to reduce C_{max}

$$p_j > p_j^{\min}, \forall j \in G_{cp}$$

If no such set STOP; otherwise continue to Step 3

Time/Cost Trade-Off Heuristic

Step 3:

For each minimum cut set:

Compute the cost of reducing all processing times by one time unit.

Take the minimum cut set with the lowest cost

If this is less than the overhead per time unit go on to Step 4; otherwise STOP

Time/Cost Trade-Off Heuristic

Step 4:

Reduce all processing times in the minimum cut set by one time units

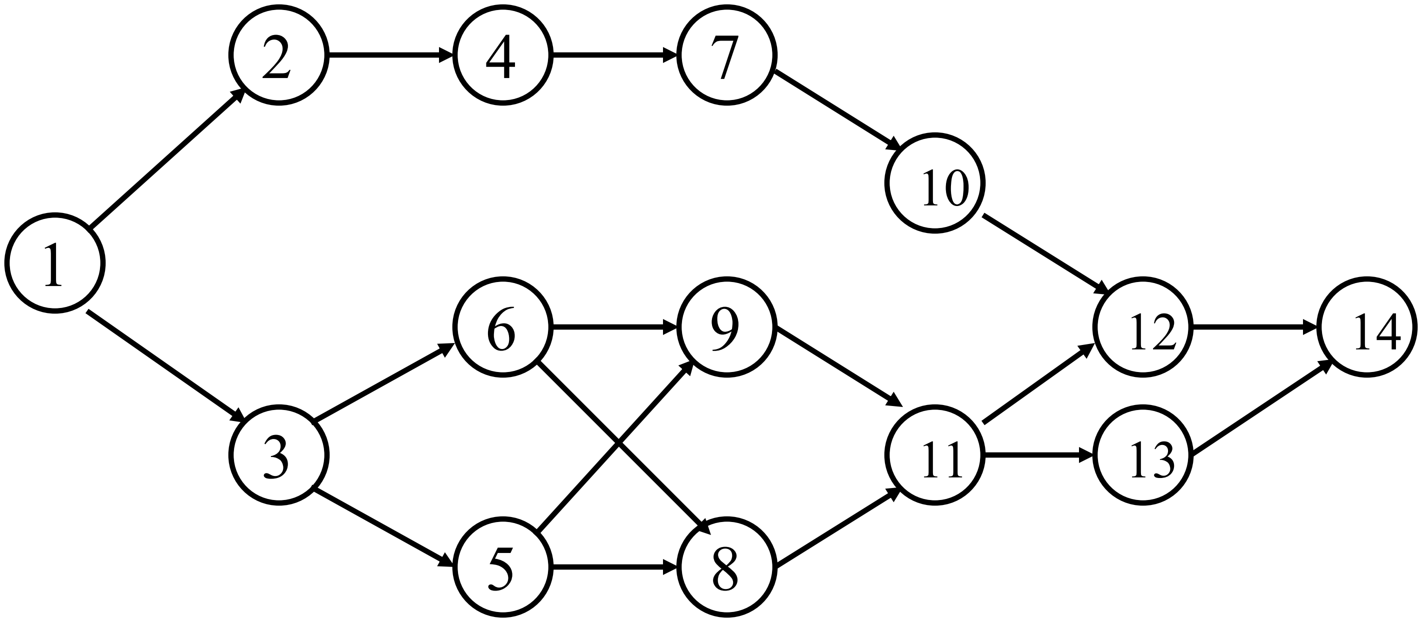
Determine the new set of critical paths

Revise graph G_{cp} and go back to Step 2

Example

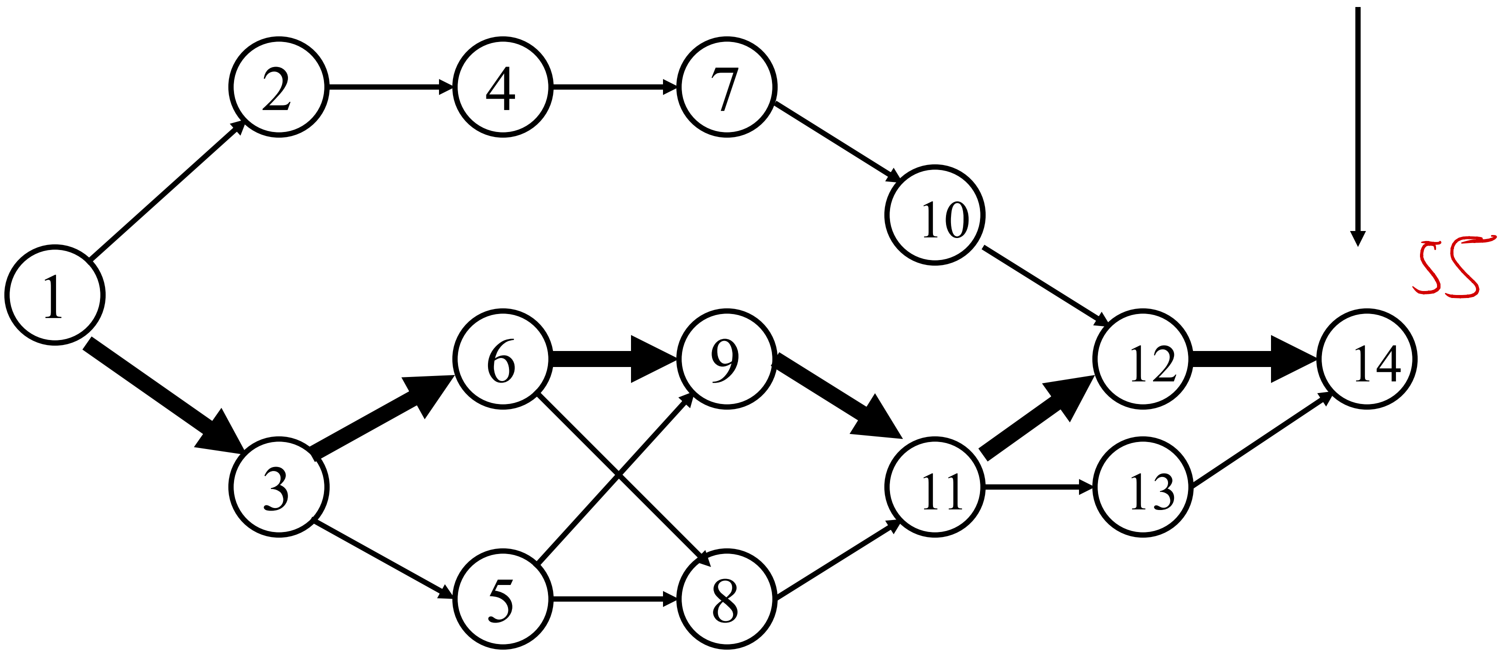
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**
Pj max	5	6	9	12	7	12	10	6	10	9	7	8	7	5
Pj min	3	5	7	9	5	9	8	3	7	5	6	5	5	2
c_j^a	20	25	20	15	30	40	35	25	30	20	25	35	20	10
c_j	7	2	4	3	4	3	4	4	4	5	2	2	4	8

Maximum Processing Times

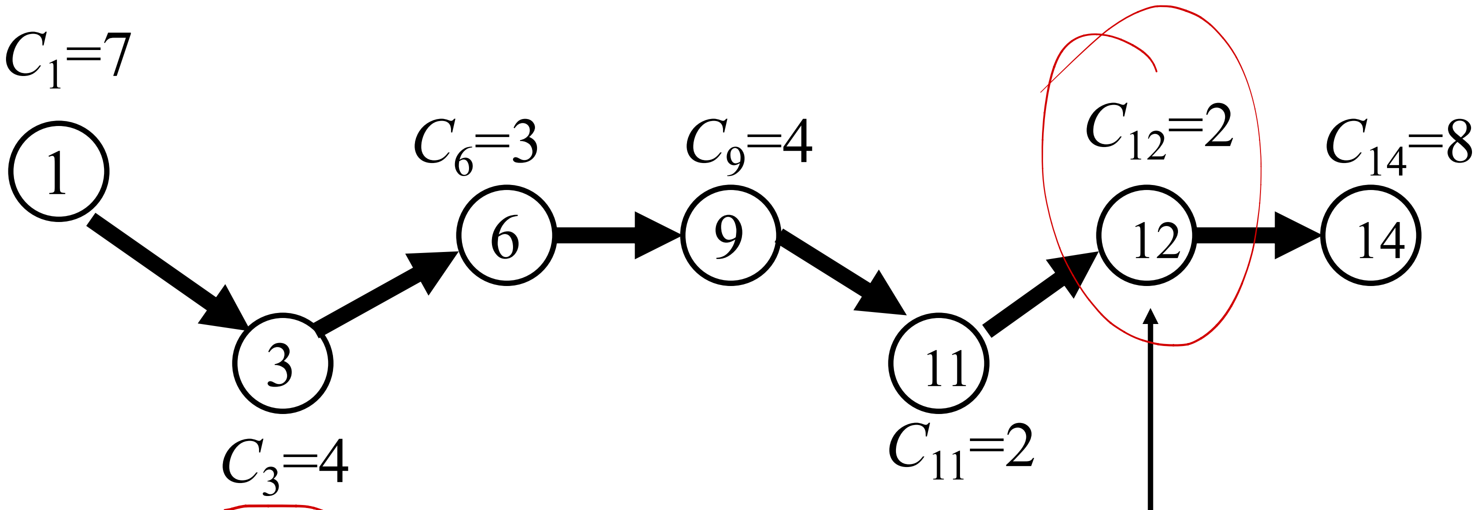


Maximum Processing Times

$C_{\max} = 56$



Critical Path Subgraph (G_{cp})

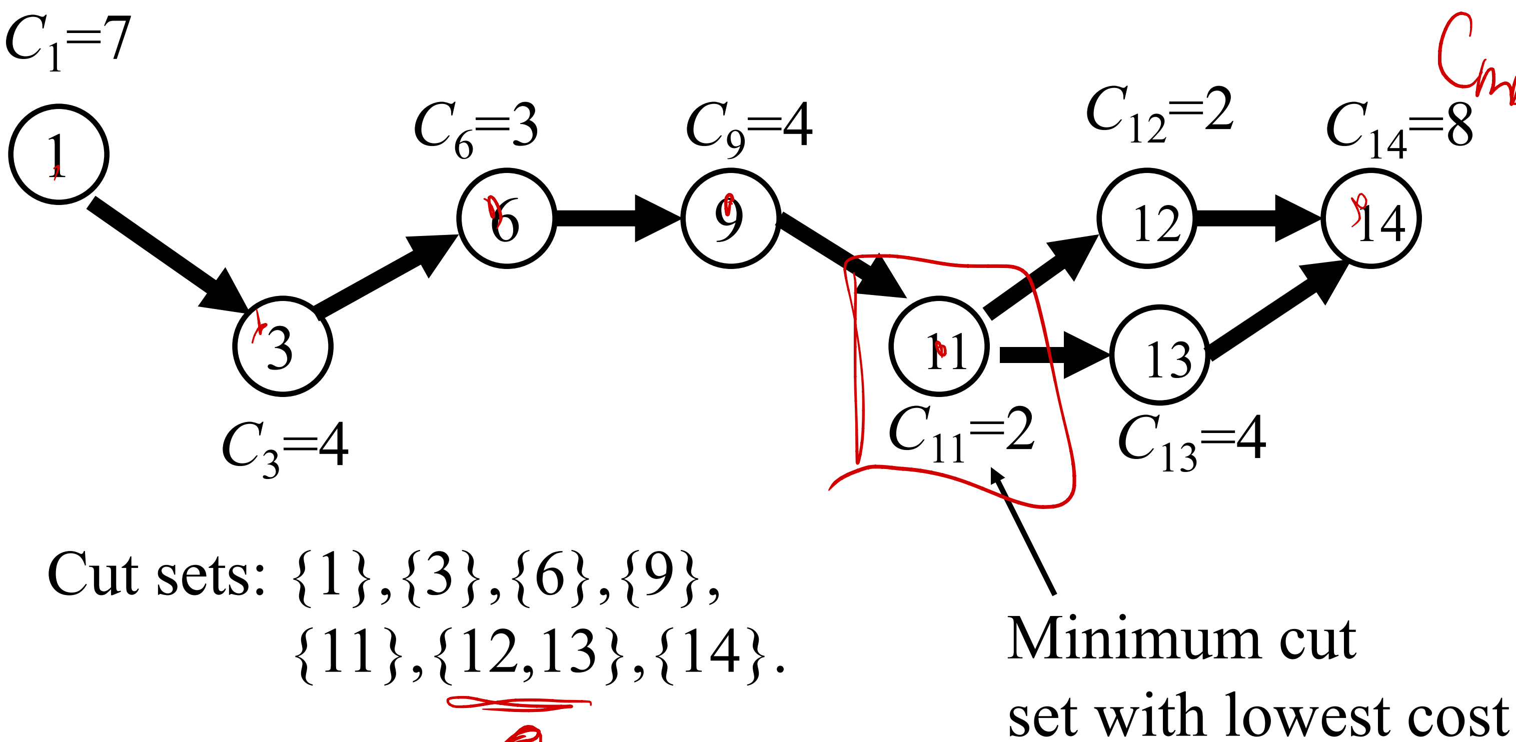


Cut sets: {1},{3},{6},{9}, {11},{12},{14}.

Minimum cut set with lowest cost

Critical Path Subgraph (G_{cp})

$12
$C_{max}=58$
$2
$C_{max}=54$



Cut sets: {1},{3},{6},{9},
{11},{12,13},{14}.

6

It may not compute the min makespan for $X

Heuristic.

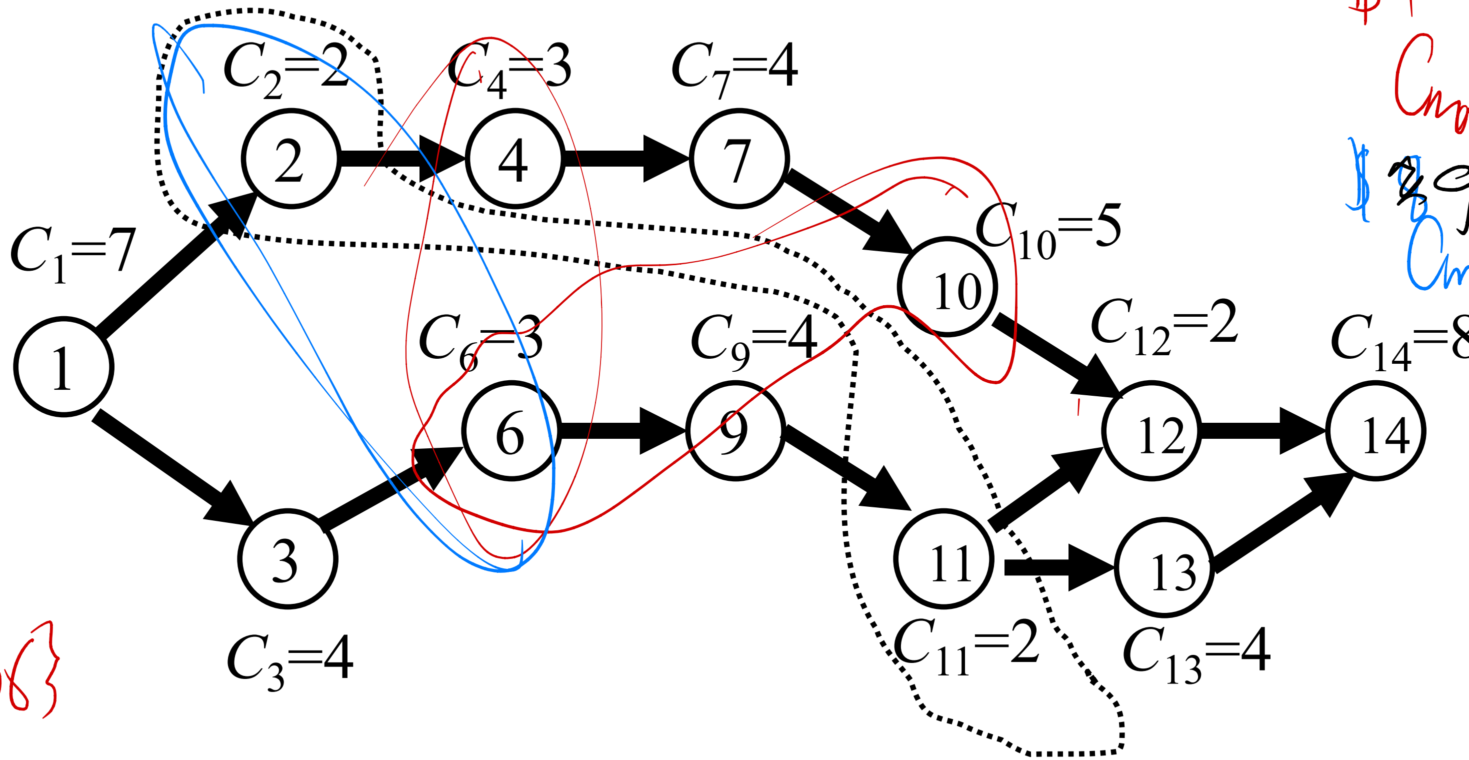
Critical Path Subgraph (G_{cp})

$54

$C_{max}=54$

$29

$C_{max}=53$



{4, 6}

{5, 10} {12, 13} {1}

This set cannot be decreased,
Choose 2 and 6 instead

Linear Programming Formulation

cost
time
C_{max}
dec var p_j

Objective is weighted avg. of makespan and cost

- Here total cost is linear

\$/min

time

$$c_0 C_{\max} + \overbrace{\sum_{j=1}^{n} c_j^b + c_j\left(p_j^{\max} - p_j\right)}^{\text{cost}}$$

min cost

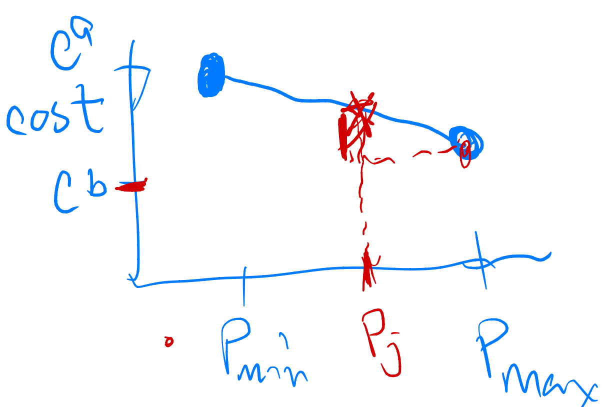
max cost

- Want to minimize

$$c_0 C_{\max} - \sum_{j=1}^{n} c_j p_j .$$

$$\cancel{\sum_{j=1}^{n} c_j^b}$$

$$\cancel{\sum_{j=1}^{n} c_j p_j^{max}}$$



min $x + 300$ $(x=12)$

312 s.t.

$x \geq 12$

$\parallel$

12

min x $(x=12)$

s.t.

$x \geq 12$

Linear Program

dec vers.
p_j, C_{max}

Minimize

$$c_0 C_{\max} - \sum_{j=1}^{n} c_j p_j .$$

x_j = start time of job j

subject to

$$x_k - p_j - x_j \geq 0, \forall j \rightarrow k \in A$$

prec.

$x_k \geq x_j + p_j$
$j \rightarrow k$

$$p_j \leq p_j^{\max}, \forall j$$

$$p_j \geq p_j^{\min}, \forall j$$

p_j is in correct range

$$x_j \geq 0, \forall j$$

$$C_{\max} - x_j - p_j \geq 0, \forall j$$

$C_{max} \geq x_j + p_j$

Cost-Time Tradeoff Heur

vs.

LP

- LP is probably slower
- LP requires that you know c_0 ($/min)
- LP require that cost/time linear tradeoff
- LP actually computes an opt sol'n if you meet the criteria above.

CPM - critical path method.

PERT

Program Evaluation and Review Technique (PERT)

- Assumed processing times deterministic
- Processing time of j random with mean μ_j and variance σ_j^2.
- Want to determine the **expected makespan**
- Assume we have

p_j^a = most optimistic processing time

p_j^m = most likely processing time (mode)

p_j^b = most pessimistic processing time

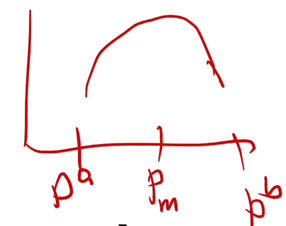
Expected Makespan

$P_a = 4$ $\quad \frac{4+24+10}{6}$
$P_m = 6$
$P_b = 10$ $\quad = 6\frac{2}{3}$

- Estimate expected processing time Assumption

assumptions ~

$$\mu_j = \frac{p_j^a + 4p_j^m + p_j^b}{6}$$



- Apply CPM with expected processing times
- Let J_{cp} be a critical path
- Estimate expected makespan

$$\hat{E}(C_{\max}) = \sum_{j \in J_{cp}} \mu_j$$

Distribution of Makespan

- Estimate the variance of processing times

$$\sigma_j^2 = \frac{p_j^b - p_j^a}{6}$$

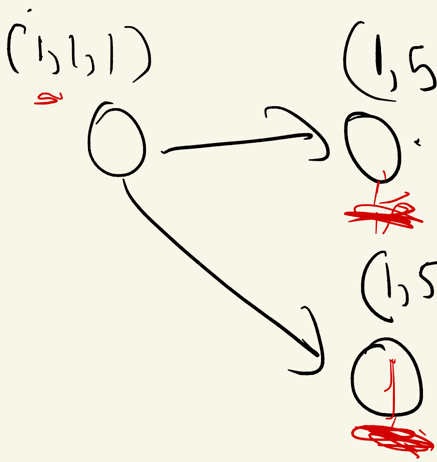
- and the variance of the makespan

$$\hat{V}(C_{\max}) = \sum_{j \in J_{cp}} \sigma_j^2$$

- Assume it is normally distributed

- take distribution for each job
- compute expected time of each job
- pretend those are deterministic time
- run CPM

(min, mode, max)



$\mu = \frac{1+4(5)+20}{6}$

$= 41/6 \approx 7$

≈ 7

CP = 8

PERT is bogus!

2 r.v. X_1, X_2

linearity of expectation

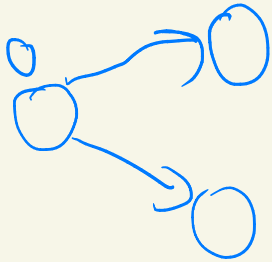
$$E[X_1 + X_2] = E(X_1) + E(X_2)$$

<u>@max</u>

$$E[\max(X_1, X_2)] \neq \max(E(X_1), E(X_2))$$

PERT assumes

$$\boxed{E(\max(X_1, X_2)) = \max(E(X_1), E(X_2))}$$



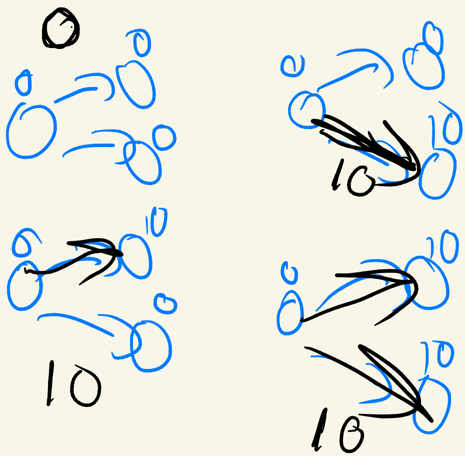
$p_j = 10$ w/ $Pr = \frac{1}{2}$
$p_j = 0$ w/ $Pr = \frac{1}{2}$

$p_j = 10$ w/ $Pr = \frac{1}{2}$
$p_j = 0$ w/ $Pr = \frac{1}{2}$

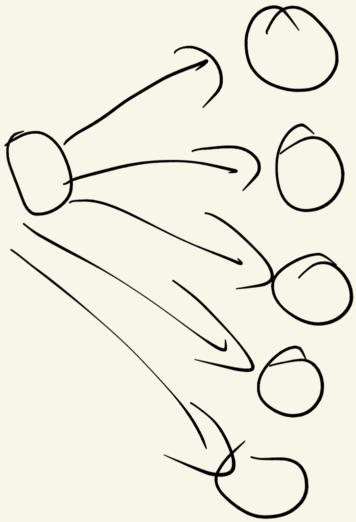
PERT

$C_{max} = 5$

What is $E[C_{max}]$ for



$E[C_{max}]$
$= \frac{1}{4}(0) + \frac{1}{4}(10) + \frac{1}{4}(10) + \frac{1}{4}(10)$
$= 7.5$



$C_{max} = 100 \quad P_r = \frac{1}{100}$

$C_{max} = 0 \quad P_r = 99/100$

$E = 1$

PERT $E(C_{max}) = 1$

Reality $E[C_{max}] \approx 100$

Discussion

- Potential problems with PERT:
  - Always underestimates project duration
    - other paths may delay the project
  - Non-critical paths ignored
    - critical path probability
    - critical activity probability
  - Activities are not always independent
    - same raw material, weather conditions, etc.
  - Estimates by be inaccurate

Discussion

- No resource constraints:
  - Critical Path Method (CPM)
  - Simple deterministic
  - Time/cost trade-offs
    - Linear cost (heuristic or exact)
    - Non-linear cost (heuristic)
  - Accounting for randomness (PERT)

expensive machines
e.g. bulldozer

renewable - reusable
non-renewable - dynamite

Adding Resource Constraints

Resource Constraints

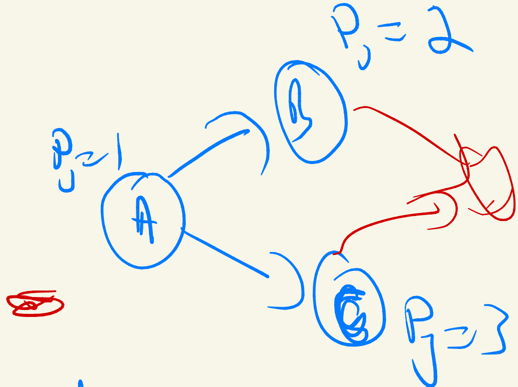
input: DAG, p_j

r resources: R_i is the amount of resource i

R_{ij} = the amount of resource i that job j needs

- Renewable resources
- Very hard problem
- No LP
- Can develop an IP
- Let job n+1 be dummy job (sink) and

$$x_{jt} = \begin{cases} 1 & \text{if job } j \text{ is completed at time } t \\ 0 & \text{otherwise.} \end{cases}$$



$C_{max}=4$

add resources

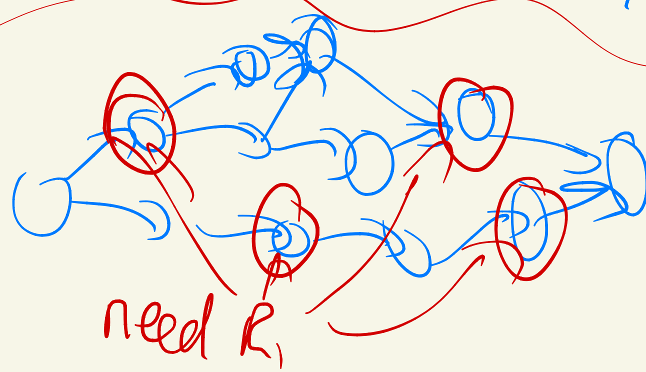
$R_1=1$

$R_{1B}=1$

$R_{1C}=1$

B+C cannot run simultaneously

$C_{max}=6$



Makespan

- Let H bound the makespan, e.g.

x_{jt}

$$H = \sum_{j=1}^{n} p_j \qquad C_j = \sum_{t=1}^{H} t \cdot x_{jt}$$

- Completion time of job j is $\sum_{t=1}^{H} t \cdot x_{jt}$ and the makespan

$$C_{n+1} = \sum_{t=1}^{H} t \cdot x_{n+1,t}$$

IP Formulation

at time t, for each i

Minimize $$\sum_{t=1}^{H} t \cdot x_{n+1,t}$$

Subject to

$\sum_j R_{ij}$ (j running at time t) $\le R_i$

C C_k

$$\sum_{t=1}^{H} t \cdot x_{jt} + p_k - \sum_{t=1}^{H} t x_{kt} \le 0$$ If j prec k

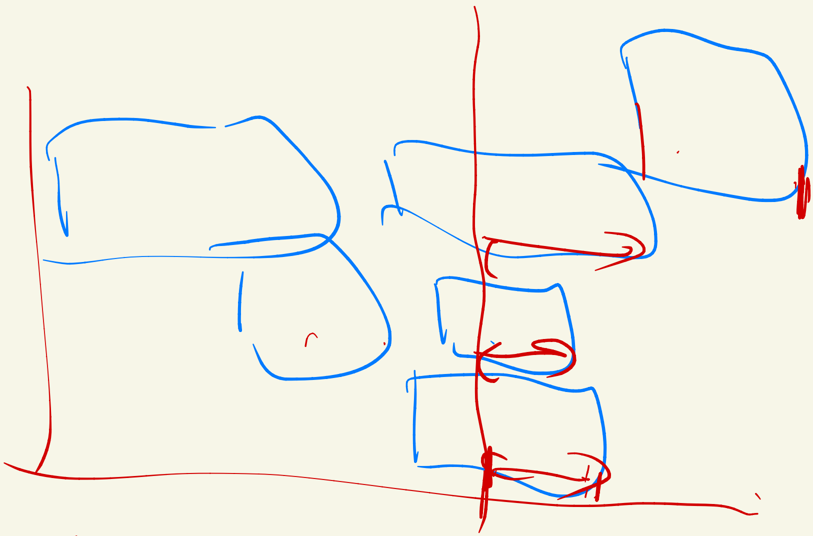
$\to C_j$

$C_k \ge C_j + P_k$

$j \to k$

$$\sum_{i=1}^{n}\left(R_{ij} \sum_{u=t}^{t+p_j-1} x_{ju} \right) \le R_j$$ Each resource Used at each time t

$$\sum_{t=1}^{H} x_{jt} = 1$$ For each job



which jobs are running at time t

any job j whose completion time is in the interval

$[t, t+p_j]$

In Practice

- The IP cannot be solved
- (Almost) always resource constraints
- $\Rightarrow$Heuristic:

  Resource constraint $\rightarrow$ Precedence constraint
- Example: pouring foundation and sidewalk
  - both require same cement mixer
  - delaying foundation delays building
  - precedence constraint: pour foundation first
  - not a logical constraint

Optimality of Heuristic

- Say *n* jobs need the same resource
- Could otherwise all be done simultaneously
- Add (artificial) precedence constraints
- Have *n*! possibilities

2	3	4	5	6
2	6	24	120	720

- Will we select the optimal sequence?

Decision Support

- Resource leveling
  - Solve with no resource constraints
  - Plot the resource use as a function of time
  - If infeasible suggest precedence constraints
    - Longest job
    - Minimum slack
  - User adds constraints
  - Start over

Example: One Resource