ENCE 603
Management Science Applications in Project Management
Lectures 5-7

Project Management LP Models
in Scheduling, Integer Programming

Spring 2009

Instructor: Dr. Steven A. Gabriel
www.eng.umd.edu/~sgabriel

1

Outline

- **Project Scheduling**
- Critical Path Method (CPM)
- AON and AOA methods
- Project Crashing
- Precedence Diagramming Method (PDM)
- Gantt Charts

2

Project Networks

- Project activities described by a network
- Can use the activity-on-node (AON) model
- Nodes are activities, arrows (arcs) indicate the precedence relationships
- Could also consider the activity-on-arc (AOA) model which has arcs for activities with nodes being the starting and ending points
- AON used frequently in practical, non-optimization situations, AOA is used in optimization settings
- First AON, then AOA
- Main idea for both is to determine the critical path (e.g., tasks whose delay will cause a delay for the whole project)

Project Networks

- Sample project network (AON) (read left to right)
- Dashed lines indicate dummy activities
- Key: Activity, Duration (days)

Network Analysis

- **Network Scheduling:**
- Main purpose of CPM is to determine the "critical path"
- Critical path determines the minimum completion time for a project
- Use forward pass and backward pass routines to analyze the project network
- **Network Control:**
- Monitor progress of a project on the basis of the network schedule
- Take correction action when required
  - "Crashing" the project
  - Penalty/reward approach

Activity on Node (AON) Representation of Project Networks

Project Networks

A: Activity identification (node)
ES: Earliest starting time
EC: Earliest completion time
LS: Latest starting time
LC: Latest completion time
t: Activity duration
P(A): set of predecessor nodes to node A
S(A): set of successor nodes to node A

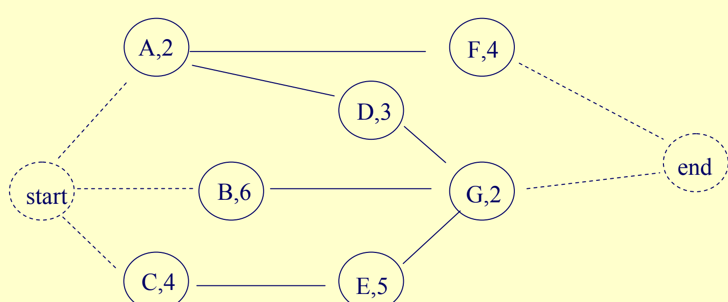
Project Networks

- In tabular form

Activity	Predecessor	Duration
A	n/a	2
B	n/a	6
C	n/a	4
D	A	3
E	C	5
F	A	4
G	B,D,E	2

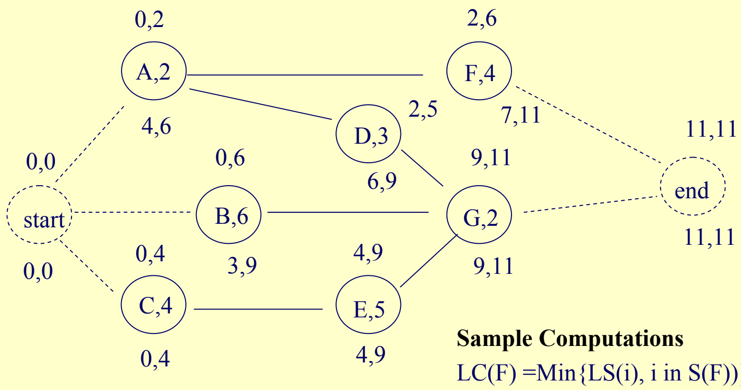
Sample Computations

ES(A) =Max{EC(j), j in P(A)}=EC(start)=0
EC(A)=ES(A)+t_A=0+2=2
ES(B)= EC(start)=0
EC(B)=ES(B)+ t_B=0+6=6
ES(F)= EC(A)=2
EC(F)= ES(F)+4=6, etc.



Project Networks

- **Notation:** Above node ES(i), EC(i), below node LS(i),LC(i)
- Zero project slack convention in force



Sample Computations

LC(F) =Min{LS(i), i in S(F))}=11

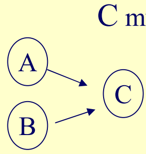
LS(F)=LC(F)-t_F=11-4=7

etc.

Project Networks

- During the **forward pass**, it is assumed that each activity will begin at its earliest starting time
- An activity can begin as soon as the last of its predecessors has finished

C must wait for both A and B to finish before it can start

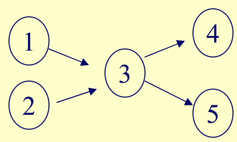


Completion of the forward pass determines the earliest completion time of the project

- During the **backward pass**, it is assumed that each activity begins at its latest completion time
- Each activity ends at the latest starting time of the first activity in the project network

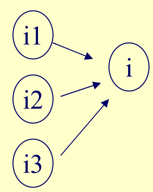
Project Networks

- **Note:**
  1=first node (activity),n=last node,i,j=arbitrary nodes, P(i)= immediate predecessors of node i, S(j)= immediate successors of node j, T_p=project deadline time



- P(3)= {1,2}
- S(3)= {4,5}

Rule 1: ES(1)=0 (unless otherwise stated)
Rule 2: ES(i)=Max j in P(i) {EC(j)}

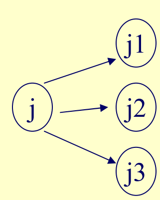


- **Why do we use "max" of the predecessor EC's in rule 2?**

Project Networks

Rule 3: EC(i)=ES(i)+t_i
Rule 4: EC(Project)=EC(n)
Rule 5: LC(Project)=EC(Project) "zero project slack convention" (unless otherwise stated for example, see Rule 6)
Rule 6: LC(Project)=Tp
Rule 7: LC(j) =Min i in S(j) LS(i)
Rule 8: LS(j)=LC(j)-tj

- **Why do we use "min" in the successor LS's in rule 7?**



Project Networks

- **Total Slack:** Amount of time an activity may be delayed from its earliest starting time without delaying the latest completion time of the project
  $TS(j)=LC(j)-EC(j)$ or $TS(j)=LS(j)-ES(j)$
- Those activities with the minimum total slack are called the critical activities (e.g., "kitchen cabinets")
- Examples of activities that might have slack
- **Free Slack:** Amount of time an activity may be delayed from its earliest starting time without delaying the starting time of any of its immediate successors.
  $FS(j)= \text{Min}_{i \text{ in } S(j)} \{ES(i)-EC(j)$
- Let's consider the sample network relative to critical activities and slack times

13

CPM-Determining the Critical Path AON

Step 1: Complete the forward pass

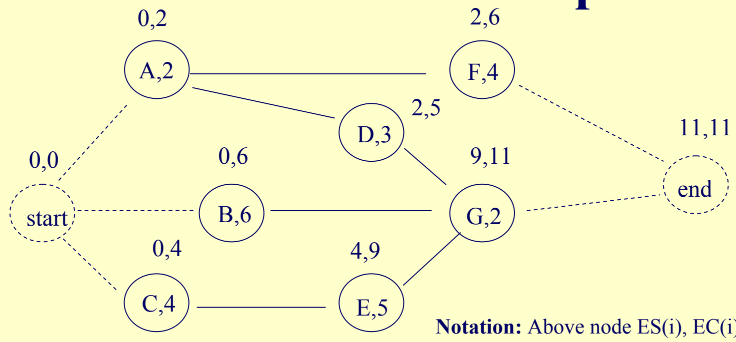
Step 2: Identify the last node in the network as a critical activity

Step 3: If activity i in P(j) and activity j is critical, check if EC(i)=ES(j). If yes → activity i is critical. When all i in P(j) done, mark j as completed

Step 4: Continue backtracking from each unmarked node until the start node is reached

14

CPM-Forward Pass Example AON



Notation: Above node ES(i), EC(i)

Activity	Predecessor	Duration
A	-	2
B	-	6
C	-	4
D	A	3
E	C	5
F	A	4
G	B,D,E	2

Sample Computations

ES(A) =Max{EC(j), j in P(A)}=EC(start)=0

EC(A)=ES(A)+t_A=0+2=2

ES(B)= EC(start)=0

EC(B)=ES(B)+ t_B=0+6=6

ES(F)= EC(A)=2

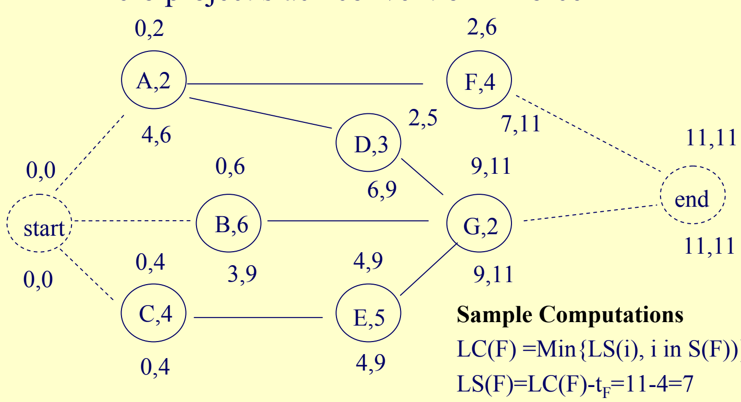
EC(F)= ES(F)+4=6, etc.

15

CPM-Backward Pass Example AON

Notation: Above node ES(i), EC(i), below node LS(i),LC(i)

- Zero project slack convention in force



Sample Computations

LC(F) =Min{LS(i), i in S(F))}=11

LS(F)=LC(F)-t_F=11-4=7

etc.

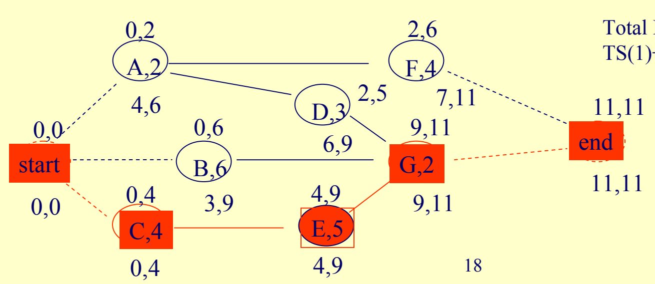
16

CPM-Slacks and the Critical Path AON

- **Total Slack:** Amount of time an activity may be delayed from its earliest starting time without delaying the latest completion time of the project
  TS(j)=LC(j)-EC(j) or TS(j)=LS(j)-ES(j)
- Those activities with the minimum total slack are called the critical activities.
- Examples of activities that might have slack
- **Free Slack:** Amount of time an activity may be delayed from its earliest starting time without delaying the starting time of any of its immediate successors.
  FS(j)= $\text{Min}_{i \text{ in } S(j)}$ {ES(i)-EC(j)}
- Other notions of slack time, see Badiru-Pulat
- Let's consider the sample network relative to critical activities and slack times

CPM Analysis for Sample Network AON

Activity	Duration (Days)	ES	EC	LS	LC	TS	FS	Critical Activity?
A	2	0	2	4	6	6-2=4	Min{2,2}-2=0	No
B	6	0	6	3	9	9-6=3	Min{9}-6=3	No
C	4	0	4	0	4	4-4=0	Min{4}-4=0	**YES**
D	3	2	5	6	9	9-5=4	Min{9}-5=4	No
E	5	4	9	4	9	9-9=0	Min{9}-9=0	**YES**
F	4	2	6	7	11	11-6=5	Min{11}-6=5	No
G	2	9	11	9	11	11-11=0	Min{11}-11=0	**YES**



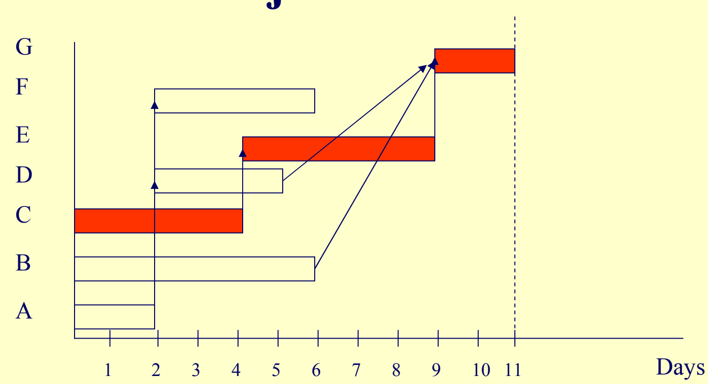
Total Project Slack
TS(1)+TS(C)+TS(E)+TS(G)+TS(n)=0

Project Networks

- When results of a CPM analysis are matched up with a calendar, then we obtain a project schedule
- Gantt chart is a popular way to present this schedule
- Using the ES times from the sample AON project network, we have the following Gantt chart
  (could also use latest completion times as well, extreme case when all slack times are fully used)

19

Project Networks

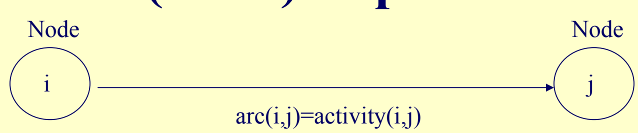


- Note, Gantt chart shows for example:
- Starting time of F can be delayed until day 7 (TS=5) w/o delaying overall project
- Also, A, D, or both may be delayed by a combined total of four days (TS=4) w/o delaying the overall project
- B may be delayed up to 3 days without affecting the overall project completion time
- Can ignore precedence arrows (better for large networks)

20

Activity on Arc (AOA) Representation of Project Networks

Project Networks: Activity on Arc (AOA) Representation



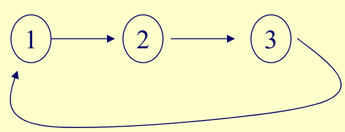
- Nodes represent the realizations of some milestones (events) of the project
- Arcs represent the activities
- Node i, the immediate predecessor node of arc(i,j) is the start node for the activity
- Node j, the immediate successor node of arc(i,j) is the end node for the activity
- Want to determine the critical path of activities, i.e., those with the least slack

Activity on Arc (AOA) Representation

- The early event time for node i, **ET(i)**, is the earliest time at which the event corresponding to node i can occur
- The late event time for node i, **LT(i)**, is the latest time at which the event corresponding to node i can occur w/o delaying the completion of the project
- Let $\mathbf{t_{ij}}$ be the duration of activity (i,j)
- The total float (slack) **TF(i,j)** of activity (i,j) is the amount by which the starting time of (i,j) could be delayed beyond its earliest possible starting time w/o delaying the completion of the project (assuming no other activities are delayed)
- $TF(i,j)=LT(j)-ET(i)-t_{ij}$
- The free float of (i,j), **FF(i,j)** is the amount by which the starting time of activity (i,j) can be delayed w/o delaying the start of any later activity beyond its earliest possible starting time
- $FF(i,j) = ET(j)-ET(i)-t_{ij}$

AOA Network Structure

- The network is *acyclic* (o/w an activity would precede itself)



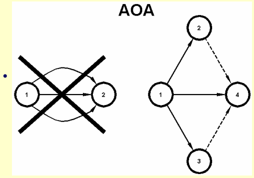
- Each node should have at least one arc directed into the node and one arc directed out of the node (with the exception of the start and end nodes), **why?**
- Start node has does not have any arc into it and the end node has no arc out of it
- All of the nodes and arcs of the network have to be visited (that is realized) in order to complete the project, **why?**

AOA Network Structure

- If a cycle exists (due perhaps to an error in the network construction), this will lead to *cycling* in the procedures
- More specifically, critical path calculations will not terminate
- Need a procedure to detect cycles in the project network (e.g., Depth-First Search method)

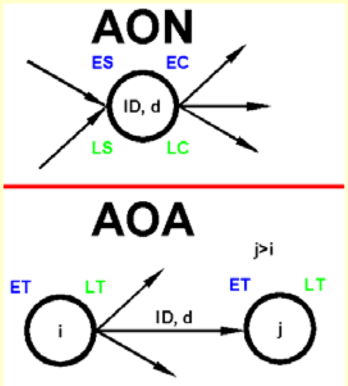
Rules in AOA Networks

1. Node 1 represents the start of the project. An arc should lead from node 1 to represent each activity that has no predecessors.
2. A node (called the finish or end node) representing completion of the project should be included in the network.
3. Number the nodes in the network so that the node representing the completion of an activity always has a larger number than the node for the start of an activity (more than 1 way to do this).
4. An activity should not be represented by more than one arc in the network.
5. Two nodes can be connected by at most one arc.

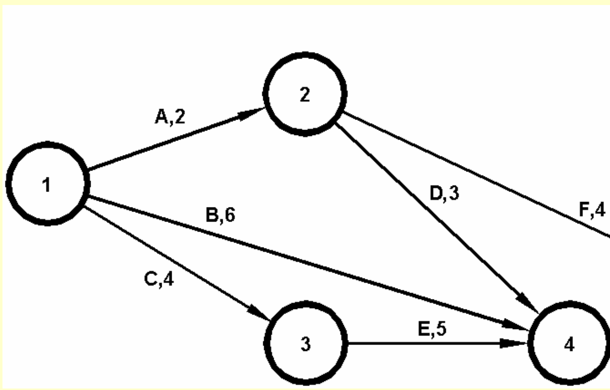


Small Sample Project

Activity	Predecessor	Duration
A	-	2
B	-	6
C	-	4
D	A	3
E	C	5
F	A	4
G	B,D,E	2



Small Sample Project AOA



Activity	Predecessor	Duration
A	-	2
B	-	6
C	-	4
D	A	3
E	C	5
F	A	4
G	B,D,E	2

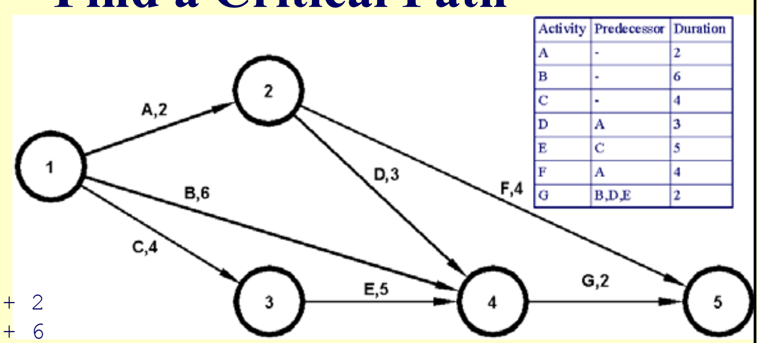
Using Linear Programming to Find a Critical Path

- Let x_j= the time that the event corresponding to node j occurs
- Let t_{ij}=the time to complete activity (i,j)
- For each activity (i,j), we know that before node j occurs, node i must occur and activity (i,j) must be completed

$$\Rightarrow x_j \geq x_i + t_{ij}, \forall (i,j)$$

- Let 1 be the index of the start node
- Let F be the index of the finish node (i.e., when the project is completed)
- LP objective function is to minimize x_F-x_1, i.e., the total project time

Using Linear Programming to Find a Critical Path

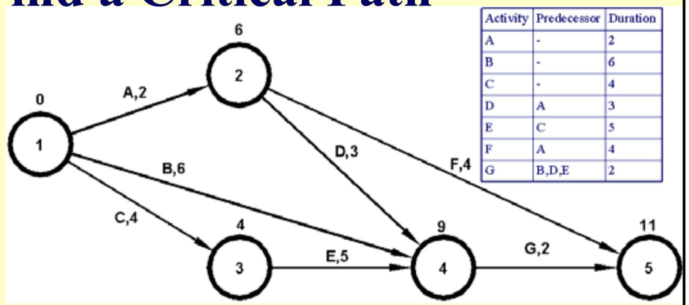


Activity	Predecessor	Duration
A	-	2
B	-	6
C	-	4
D	A	3
E	C	5
F	A	4
G	B,D,E	2

```
Min x5 - x1
s.t.
A) x2 >= x1 + 2
B) x4 >= x1 + 6
C) x3 >= x1 + 4
D) x4 >= x2 + 3
E) x4 >= x3 + 5
F) x5 >= x2 + 4
G) x5 >= x4 + 2
Variables unrestricted in sign
```

Using Linear Programming to Find a Critical Path

```
Min x5 - x1
s.t.
A) x2 - x1 >= 2
B) x4 - x1 >= 6
C) x3 - x1 >= 4
D) x4 - x2 >= 3
E) x4 - x3 >= 5
F) x5 - x2 >= 4
G) x5 - x4 >= 2
end
free x1
free x2
free x3
free x4
free x5
```



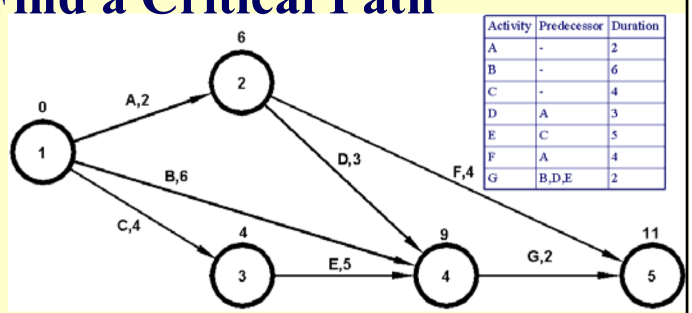
Activity	Predecessor	Duration
A	-	2
B	-	6
C	-	4
D	A	3
E	C	5
F	A	4
G	B,D,E	2

```
OBJECTIVE FUNCTION VALUE      11.00000

VARIABLE          VALUE          REDUCED COST
      X5       11.000000            0.000000
      X1        0.000000            0.000000
      X2        6.000000            0.000000
      X4        9.000000            0.000000
      X3        4.000000            0.000000
```


Using Linear Programming to Find a Critical Path

```
Min x5 - x1
s.t.
A) x2 - x1 >= 2
B) x4 - x1 >= 6
C) x3 - x1 >= 4
D) x4 - x2 >= 3
E) x4 - x3 >= 5
F) x5 - x2 >= 4
G) x5 - x4 >= 2
end
free x1
free x2
free x3
free x4
free x5
```



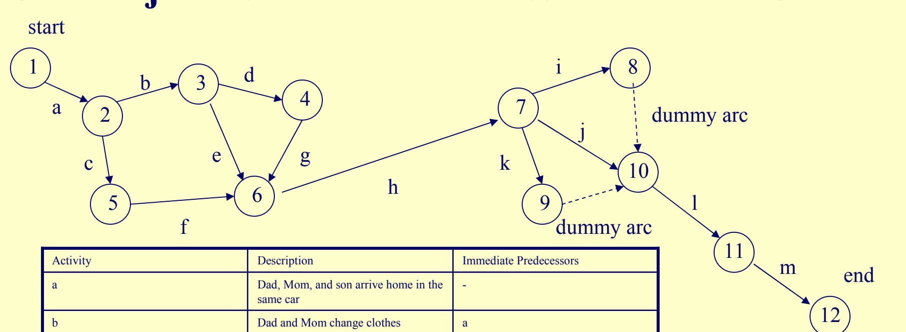
Activity	Predecessor	Duration
A	-	2
B	-	6
C	-	4
D	A	3
E	C	5
F	A	4
G	B,D,E	2

```
ROW    SLACK OR SURPLUS     DUAL PRICES
      A)        4.000000        0.000000
      B)        3.000000        0.000000
      C)        0.000000       -1.000000
      D)        0.000000        0.000000
      E)        0.000000       -1.000000
      F)        1.000000        0.000000
      G)        0.000000       -1.000000
```

Using Linear Programming to Find a Critical Path

- For each variable with zero value and zero reduced cost there is an alternative optimal solution.
- For each constraint with zero slack and zero dual variable there is an alternative optimal solution.
- For each constraint with a dual price of −1, increasing the duration of the activity corresponding to that constraint by one day will increase the duration of the project by one day. Those constraints identify the critical activities.

AOA Project Network: After-Work-Hours Chores



Activity	Description	Immediate Predecessors
a	Dad, Mom, and son arrive home in the same car	-
b	Dad and Mom change clothes	a
c	Son watches TV	a
d	Mom warns up the food*	b
e	Dad sets the table	b
f	Son does homework**	c
g	Mom fixes salad	d
h	The family eats dinner	e,f,g
i	Dad loads the dishwasher	h
j	Mom checks son's homework	h
k	Son practices *(insert musical instrument name here)*	h
l	All go to son's basketball game	i,j,k
m	All wash up and go to bed	l

*For politically correct project networks, "Mom" and "Dad" are interchangeable.
** In a perfect world, activity f precedes activity c!

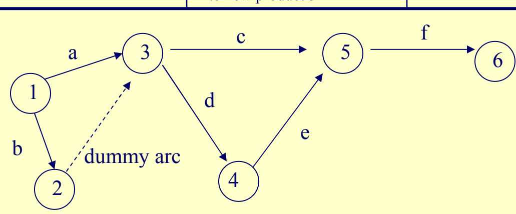
Dummy Arcs in AOA Networks

- Since activities:
  i (Dad loads dishwasher),j (Mom checks son's homework), and k (son practices musical instrument)
  all have the same predecessor activity h (family eats dinner) and the same immediate successor, activity l (go to basketball game), this would mean 3 parallel arcs between nodes 7 and 10
- An activity network allows only one arc between any two nodes so nodes 8 and 9 are drawn and connected to node 10 via dummy arcs

35

Finding the Critical Path in an AOA Project Network for Introducing a New Product

Activity	Description	Immediate Predecessors	Duration (Days)
a	Train workers	-	6
b	Purchase raw materials	-	9
c	Produce product 1	a,b	8
d	Produce product 2	a,b	7
e	Test product 2	d	10
f	Assemble products 1 and 2 into new product 3	c,e	12



36

Finding the Critical Path in an AOA Project Network for Introducing a New Product

```
min x6-x1
s.t.
x3-x1>=6   ! arc (1,3)
x2-x1>=9   ! arc (1,2)
x5-x3>=8   ! arc (3,5)
x4-x3>=7   ! arc (3,4)
x5-x4>=10  ! arc (4,5)
x6-x5>=12  ! arc (5,6)
x3-x2>=0   ! arc (2,3)
end
! could have variables free or not
!free x1 x2 x3 x4 x5 x6
```

Why variables free (i.e., not necessarily nonnegative)? When ok, when not?

Excel version of this LP?

Finding the Critical Path in an AOA Project Network for Introducing a New Product

LP OPTIMUM FOUND AT STEP 1

OBJECTIVE FUNCTION VALUE

1) 38.00000 ← Project completed in 38 days

VARIABLE	VALUE	REDUCED COST
X6	38.000000	0.000000
X1	0.000000	0.000000
X3	9.000000	0.000000
X2	9.000000	0.000000
X5	26.000000	0.000000
X4	16.000000	0.000000

← LP will have many alternate optima all with 38 days. In general, the value of x_i in an optimal solution may assume any value between ET(i) and LT(i).

ROW	SLACK OR SURPLUS	DUAL PRICES
2)	3.000000	0.000000
3)	0.000000	-1.000000
4)	9.000000	0.000000
5)	0.000000	-1.000000
6)	0.000000	-1.000000
7)	0.000000	-1.000000
8)	0.000000	-1.000000

← Critical path goes from start to finish node in which each arc corresponds to a constraint with "dual price"=-1, i.e., 1-2-3-4-5-6 is a CP (more on dual prices later...)

Finding the Critical Path in an AOA Project Network for Introducing a New Product

- For each constraint with a "dual price" of –1, increasing the duration of the activity corresponding to that constraint by delta days will increase the duration of the project by delta days
- This assumes that the current vertex remains optimal
- Now we consider a time-cost tradeoff approach to scheduling

Project Crashing in Activity on Arc (AOA) Project Networks

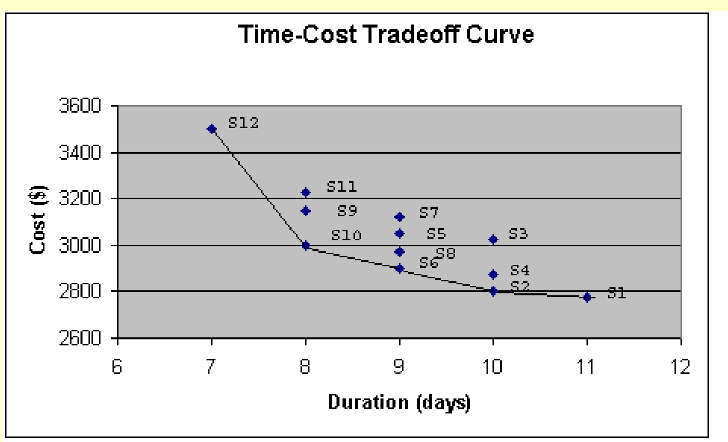
Project Crashing and Time-Cost Analysis, Sample Data

Project Duration	Crashing Strategy	Description of Crashing	Total Cost
T=11	S1	Activities at normal duration	$2,775
T=10	S2	Crash F by 1 unit	$2,800
T=10	S3	Crash C by 1 unit	$3,025
T=10	S4	Crash E by 1 unit	$2,875
T=9	S5	Crash F and C by 1 unit	$3,050
T=9	S6	Crash F and E by 1 unit	$2,900
T=9	S7	Crash C and E by 1 unit	$3,125
T=9	S8	Crash E by 2 units	$2,975
T=8	S9	Crash F, C, and E by 1 unit	$3,150
T=8	S10	Crash F by 1 unit, E by 2 units	$3,000
T=8	S11	Crash C by 1 unit, E by 2 units	$3,225
T=7	S12	Crash F and C by 1 unit, and E by 2 units	$3,500

- **If c "crashable" activities, there are 2^c possible crash strategies, why?**
- Suppose we can crash 6 of the 7 activities➔ 2^6=64 possible crash strategies
- There are 12 of the 64 strategies shown here

41

Project Crashing and Time-Cost Analysis



42

Project Crashing and Time-Cost Analysis –A Specific Example

- Define the variables:
  - A= # of days by which activity a is reduced (unit cost =$10)
  - B= # of days by which activity b is reduced (unit cost =$20)
  - C= # of days by which activity c is reduced (unit cost =$3)
  - D= # of days by which activity d is reduced (unit cost =$30)
  - E= # of days by which activity e is reduced (unit cost =$40)
  - F= # of days by which activity f is reduced (unit cost =$50)

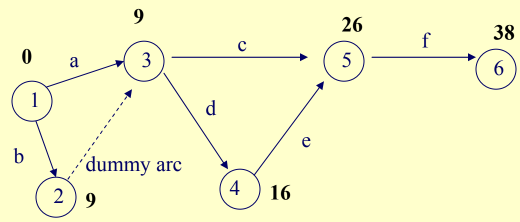
- We have the following LP

Project Crashing and Time-Cost Analysis -An Example

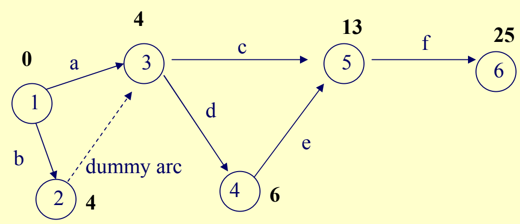
```
min 10A+20B+3C+30D+40E+50F
s.t.
A<=5
B<=5
C<=5
D<=5
E<=5
F<=5
x3-x1+A>=6    ! arc (1,3)
x2-x1+B>=9    ! arc (1,2)
x5-x3+C>=8    ! arc (3,5)
x4-x3+D>=7    ! arc (3,4)
x5-x4+E>=10  ! arc (4,5)
x6-x5+F>=12  ! arc (5,6)
x3-x2>=0       ! arc (2,3)
x6-x1<=25      ! at most 25 days
end
! could have variables free or not
!free x1 x2 x3 x4 x5 x6
```

Excel version of this LP?

Project Crashing and Time-Cost Analysis -An Example



Normal AOA network
Total project time=38 days
CP=1-2-3-4-5-6
Numbers by nodes are solution



Crashed AOA network
Total cost = $390
Total project time=25 days
CPs=1-2-3-4-5-6, 1-3-4-5-6
Numbers by nodes are solution
Crash variables:
A=2,B=5,C=0,D=5,E=3,F=0

Precedence Diagramming Method in Activity on Arc (AOA) Project Networks

Precedence Diagramming Method (PDM)

- Normal CPM assumptions are that a task B cannot start until its predecessor task A is completely finished
- PDM allows activities that are mutually dependent to be performed partially in parallel instead of serially
- The usual finish-to-start dependencies are “relaxed” so that the performance of the activities can be overlapped
- The result is that the project schedule can be compressed (like project crashing in that sense)

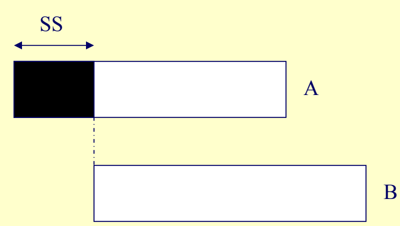
47

Precedence Diagramming Method (PDM)

- The time between the finishing or starting time of the 1st activity and the finishing or starting time of the 2nd activity is called the *lead-lag* requirement between the two activities
- Four basic lead-lag relationships to consider:

1. **Start-to-Start Lead** (SS_{AB}) This specifies that activity B cannot start until activity A has been in progress for at least SS time units

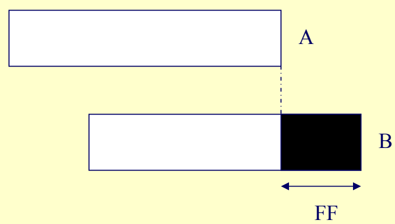
Example?



48

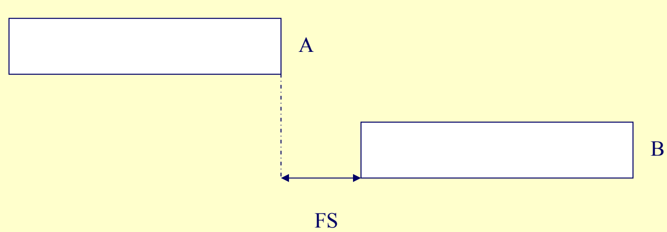
Precedence Diagramming Method (PDM)

2. **Finish-to-Finish Lead** (FF_{AB}) This specifies that activity B cannot finish until at least FF time units after the completion of activity A
Example?



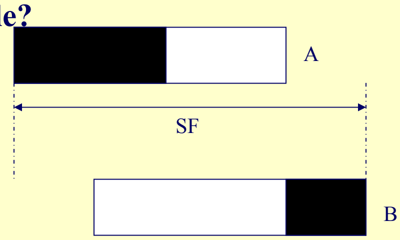
Precedence Diagramming Method (PDM)

3. **Finish-to-Start Lead** (FS_{AB}) This specifies that activity B cannot start until at least FS time units after the completion of activity A (CPM takes FS_{AB}=0)
Example?



Precedence Diagramming Method (PDM)

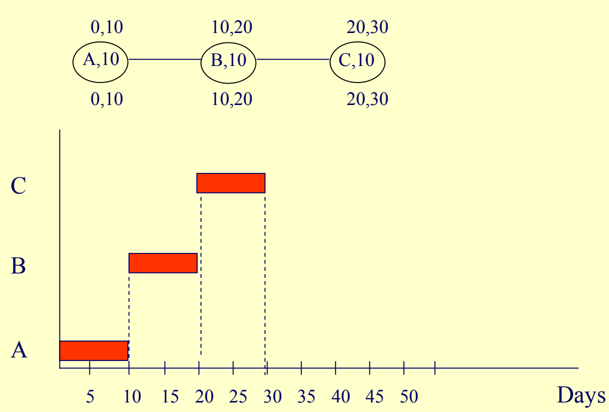
4. **Start-to-Finish Lead** (SF_{AB}) This specifies that there must be at least SF time units between the start of activity A and the completion of activity B
Example?



- Can also express the leads or lags in percentages (instead of time units)
- Can also use "at most" relationships as well as the "at least" ones shown above

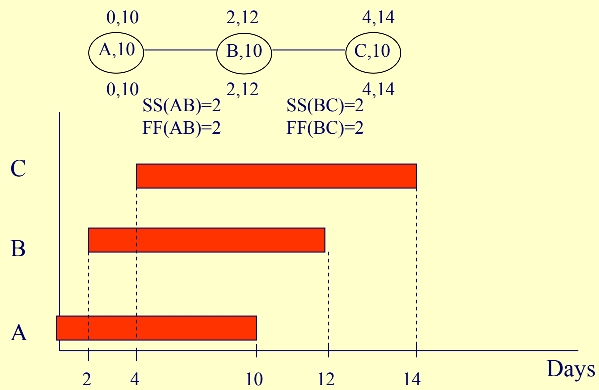
Precedence Diagramming Method (PDM)

- An example: 3 activities done in series
➔ project duration of 30 days using conventional CPM method



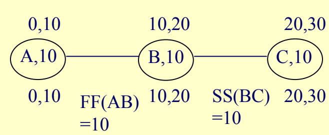
Precedence Diagramming Method (PDM)

- The same 3 activities done in series but with lead-lag constraints ➔ project duration of 14 days, a 16 day speedup over the conventional CPM method



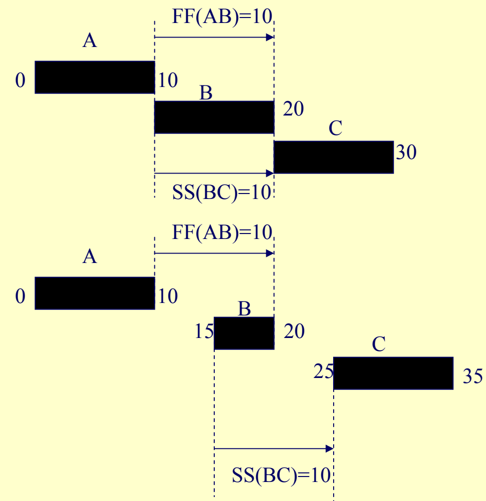
Precedence Diagramming Method (PDM)

- Must be careful about possible anomalies in PDM
- Example:



- Now crash B and reduce the duration of task B from 10 days to 5 days
- You would think that the total projection duration would decrease from 30 to something lower
- However, the SS(BC) constraint forces the starting time of C to be shifted forward by 5 days➔ project duration actually *increases* even though B's duration has *decreased*!

Precedence Diagramming Method (PDM)



Before B's time is reduced

After B's time is reduced

As a safeguard, may want to perform one activity change at a time and record the result

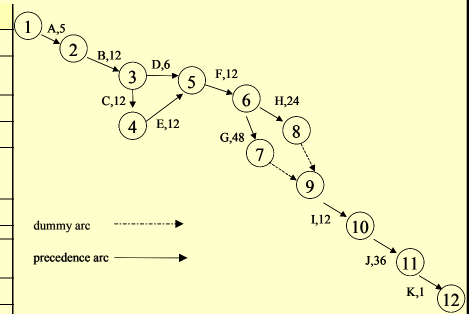
Precedence Diagramming Method Example

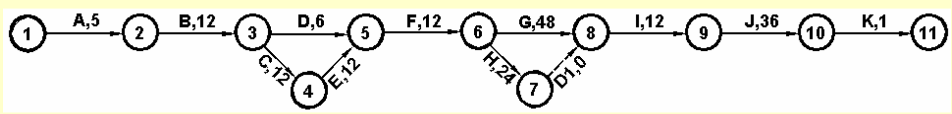
You are a planner at the National Aeronautics and Space Administration (NASA) planning the next major rocket development, production, and launching to the planet Neptune. Due to the particular positioning of the planet Neptune relative to Earth and the other planets in between, the rocket must be within 100,000 kilometers of the planet Saturn somewhere between <u>120 and 125 months</u> from today in order to make it to Neptune in a reasonable amount of time.

If this time window is not satisfied, the cost of reaching Neptune skyrockets dramatically (no pun intended). For example, if the time is greater than 125 months, it is estimated that $100 million more are needed to reach Neptune due to additional engineering considerations. Consider the following set of activities related to this project shown in the following table.

PDM-AOA Project Network for Example

Activity	Activity Description	Immediate Predecessors	Duration (months)
A	Develop trajectory plan for rocket	-	5
B	Generate specifications for rocket design	A	12
C	Request funding from Congress	B	12
D	Begin initial search for contractors	B	6
E	Prepare modified budget (using suggestions from Congress)	C	12
F	Select rocket contractor	D,E	12
G	Build and test rocket	F	48
H	Create simulation model for rocket trajectory	F	24
I	Prepare rocket launch & launch rocket	G,H	12
J	Proceed towards Jupiter and then perform gravitational "slingshot" maneuver around Jupiter	I	36
K	Achieve Saturn orbit (within 100,000 kilometers)	J	1

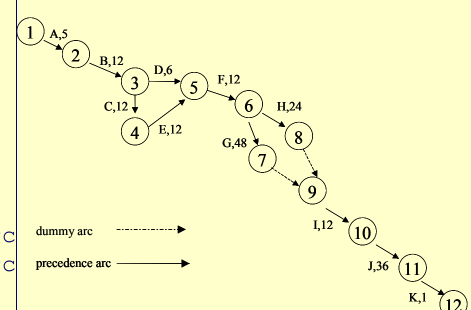




PDM-Example

•Compute the critical paths and project duration by formulating and solving an appropriate LP model to capture the precedence relationships between the activities. Let xi be the time for node i. The associated LP model is thus:

```
min x12-x1
s.t.
A)   x2-x1   >=  5  ! A
B)   x3-x2   >= 12  ! B
C)   x4-x3   >= 12  ! C
D)   x5-x3   >=  6  ! D
E)   x5-x4   >= 12  ! E
F)   x6-x5   >= 12  ! F
G)   x7-x6   >= 48  ! G
H)   x8-x6   >= 24  ! H
DU1) x9-x8   >=  0  ! dummy arc
DU2) x9-x7   >=  0  ! dummy arc
I)   x10-x9  >= 12  ! I
J)   x11-x10 >= 36  ! J
K)   x12-x11 >=  1  ! K
```



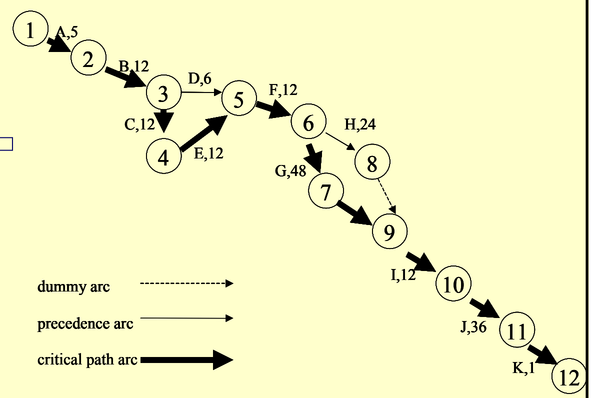
PDM-Example

```
OBJECTIVE FUNCTION VALUE

       1)      150.0000

  VARIABLE        VALUE          REDUCED COST
       X12       150.000000          0.000000
        X1         0.000000          0.000000
        X2         5.000000          0.000000
        X3        17.000000          0.000000
        X4        29.000000          0.000000
        X5        41.000000          0.000000
        X6        53.000000          0.000000
        X7       101.000000          0.000000
        X8        77.000000          0.000000
        X9       101.000000          0.000000
       X10       113.000000          0.000000
       X11       149.000000          0.000000

       ROW   SLACK OR SURPLUS     DUAL PRICES
        A)         0.000000         -1.000000
        B)         0.000000         -1.000000
        C)         0.000000         -1.000000
        D)        18.000000          0.000000
        E)         0.000000         -1.000000
        F)         0.000000         -1.000000
        G)         0.000000         -1.000000
        H)         0.000000          0.000000
      DU1)        24.000000          0.000000
      DU2)         0.000000         -1.000000
        I)         0.000000         -1.000000
        J)         0.000000         -1.000000
        K)         0.000000         -1.000000
```



We see that the project time is 150 months which is too high (greater than 125 months).

PDM-Example

Question:
Due to various reasons, it is believed that activities A, B, C, and D can be sped up as follows. Activity B can start as soon as 1 month after A starts. Activity C can start as soon as 1 month after activity B starts. Activity D can start as soon as one month after activity C starts. The total cost for this acceleration is $5,000,000. Modify the project network from part a (i.e., the uncrashed one) to allow for these possibilities. What is the total project time allowing for these changes?

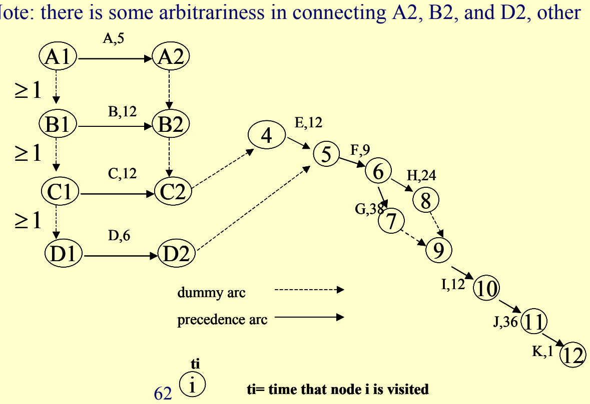
Note: Could also try project crashing to speed things up, not considered here.

PDM Combined with LP

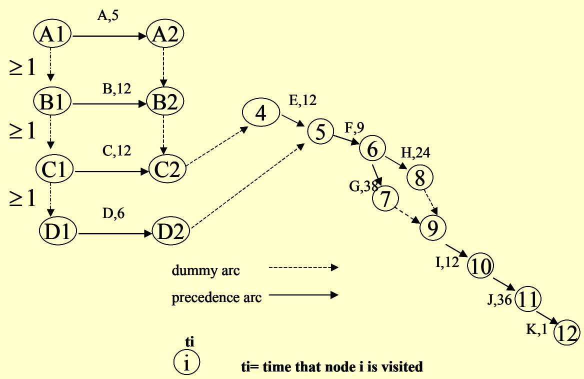
- Create one start and one end node for each activity that has a PDM rule.
- Insert arrows to enforce the new relationships
- Solve as previous cases

PDM-Example

Answer:
We can modify the AOA network to include two nodes for A, namely A1 and A2 when activity A starts and when it finishes, respectively. The same modification can be applied to activities B, C, and D. We need to make sure that the earliest that B1 can start is 1 month after A1, the earliest that C1 can start is 1 month after B1, and the earliest that D1 can start is 1 month after C1. The resulting new project network is as follows. Note: there is some arbitrariness in connecting A2, B2, and D2, other slight variations are possible.



PDM-Example




```
New model is thus:
min x12-xa1
s.t.
A)   xa2-xa1>=  5  ! A
B)   xb2-xb1>= 12  ! B
C)   xc2-xc1>= 12  ! C
D)   xd2-xd1>=  6  ! D

SS1)xb1-xa1>=  1  ! SS(A,B)=1
SS2)xc1-xb1>=  1  ! SS(B,C)=1
SS3)xd1-xc1>=  1  ! SS(C,D)=1

A2) xb2-xa2>=  0  ! A2 before B2
B2) xc2-xb2>=  0  ! B2 before C2
C2) x4 -xc2>=  0  ! C2 before 4
D2) x5 -xd2>=  0  ! D2 before 5

E)   x5 -x4 >= 12  ! E
F)   x6 -x5 >= 12  ! F
G)   x7 -x6 >= 48  ! G
H)   x8 -x6 >= 24  ! H
DU1) x9 -x8>=  0  ! dummy arc
DU2) x9 -x7>=  0  ! dummy arc
I)   x10-x9 >= 12  ! I
J)   x11-x10>= 36  ! J
K)   x12-x11>=  1  ! K
```

63

PDM-Example

OBJECTIVE FUNCTION VALUE

1) 135.0000

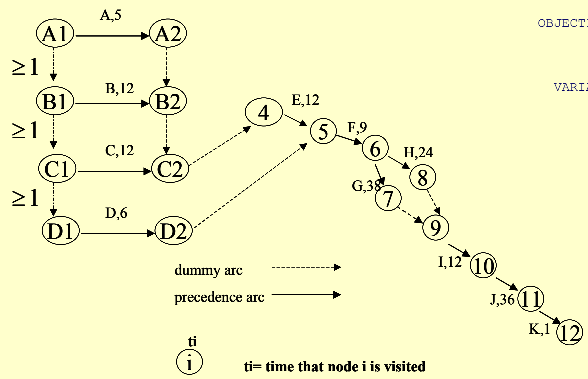
VARIABLE	VALUE	REDUCED COST
X12	135.000000	0.000000
XA1	0.000000	0.000000
XA2	14.000000	0.000000
XB2	14.000000	0.000000
XB1	1.000000	0.000000
XC2	14.000000	0.000000
XC1	2.000000	0.000000
XD2	26.000000	0.000000
XD1	3.000000	0.000000
X4	14.000000	0.000000
X5	26.000000	0.000000
X6	38.000000	0.000000
X7	86.000000	0.000000
X8	86.000000	0.000000
X9	86.000000	0.000000
X10	98.000000	0.000000
X11	134.000000	0.000000

ROW	SLACK OR SURPLUS	DUAL PRICES
A)	9.000000	0.000000
B)	1.000000	0.000000
C)	0.000000	-1.000000
D)	17.000000	0.000000
SS1)	0.000000	-1.000000
SS2)	0.000000	-1.000000
SS3)	0.000000	0.000000
A2)	0.000000	0.000000
B2)	0.000000	0.000000
C2)	0.000000	-1.000000
D2)	0.000000	0.000000
E)	0.000000	-1.000000
F)	0.000000	-1.000000
G)	0.000000	-1.000000
H)	24.000000	0.000000
DU1)	0.000000	0.000000
DU2)	0.000000	-1.000000
I)	0.000000	-1.000000
J)	0.000000	-1.000000
K)	0.000000	-1.000000

Total project time is 135 months, still too big, will need to consider crashing the project.

64

PDM Example



OBJECTIVE FUNCTION VALUE

1) 135.0000

VARIABLE	VALUE	REDUCED COST
X12	135.000000	0.000000
XA1	0.000000	0.000000
XA2	14.000000	0.000000
XB2	14.000000	0.000000
XB1	1.000000	0.000000
XC2	14.000000	0.000000
XC1	2.000000	0.000000
XD2	26.000000	0.000000
XD1	3.000000	0.000000
X4	14.000000	0.000000
X5	26.000000	0.000000
X6	38.000000	0.000000
X7	86.000000	0.000000
X8	86.000000	0.000000
X9	86.000000	0.000000
X10	98.000000	0.000000
X11	134.000000	0.000000

Integer Programming

- All linear programming problems so far assumed that fractional answers were acceptable
  - In practice not always ok, **why**?
  - Certain classes of LPs we studied will have integer solutions, which ones and why?
- Want to explore modeling aspects when we specify certain variables must be integer-valued
  - Why this is a MUCH harder problem to solve in general
  - Interesting applications of binary variables for encoding logic in mathematical programs

The Toy Problem Revisited

67

The Toy Problem Revisited

Recall the toy production problem from before

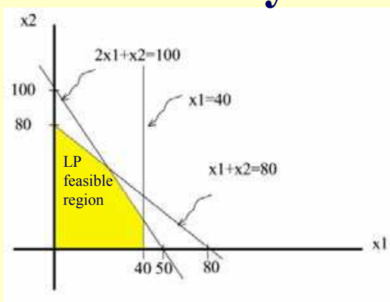
- Complete LP

Max 3x1+2x2	(Objective function)
s.t.	
2x1+x2 <=100	(Finishing constraint)
x1+ x2 <=80	(Carpentry constraint)
x1 <=40	(Limited demand constraint on soldiers)
x1 >=0	(Nonnegativity constraint on soldiers)
x2 >=0	(Nonnegativity constraint on cars)

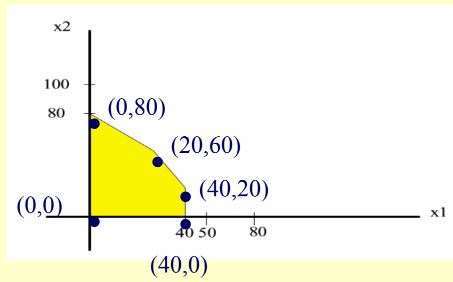
Optimal solution: x1=20, x2=60 , with an optimal objective function value of z=$180

68

The Toy Problem Revisited



- According to LP theory, a solution (if it exists) must be at one of the vertices (also called extreme points)



- In this case, all vertices are integer-valued (i.e., whole numbers)
- This is fortunate since we want to produce a whole number of toys and soldiers
- What if this were not the case? That is, what if the the solution were not integer-valued?

The Toy Problem Revisited

- Modified Complete LP

Max 3x1+2x2	(Objective function)
s.t.	
2x1+x2 <=99.7	(Finishing constraint)
x1+ x2 <=83.5	(Carpentry constraint)
x1 <=40	(Limited demand constraint on soldiers)
x1 >=0	(Nonnegativity constraint on soldiers)
x2 >=0	(Nonnegativity constraint on cars)

New optimal solution: x1=16.199997, x2=67.300003 , with an optimal objective function value of z=$183.2

- **How has the feasible region changed**?
- But this fractional answer really doesn't really make sense, we don't want to produce a fractional number of toy soldiers or cars (no one would buy them).

The Toy Problem Revisited

- We add integer constraints
- Complete Integer Program (IP)

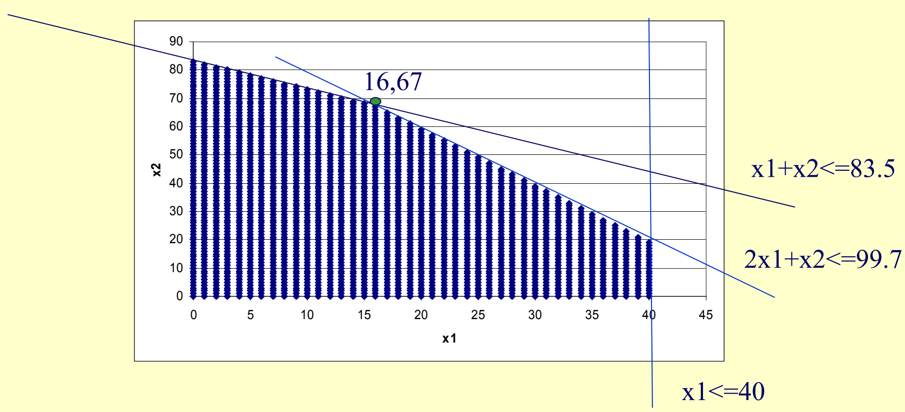
Max 3x1+2x2 (Objective function)
s.t.
2x1+x2 <=99.7 (Finishing constraint)
x1+ x2 <=83.5 (Carpentry constraint)
x1 <=40 (Limited demand constraint on soldiers)
x1 >=0 (Nonnegativity constraint on soldiers)
x2 >=0 (Nonnegativity constraint on cars)

x1, x2 integer-valued

New optimal solution: x1=16, x2=67 , with an optimal objective function value of z=$182

- **Note:** Just rounding to the nearest integer worked in this case but in general, it won't even produce a feasible solution. Not a good way to solve a MIP
- **What does the feasible region look like in this case**?

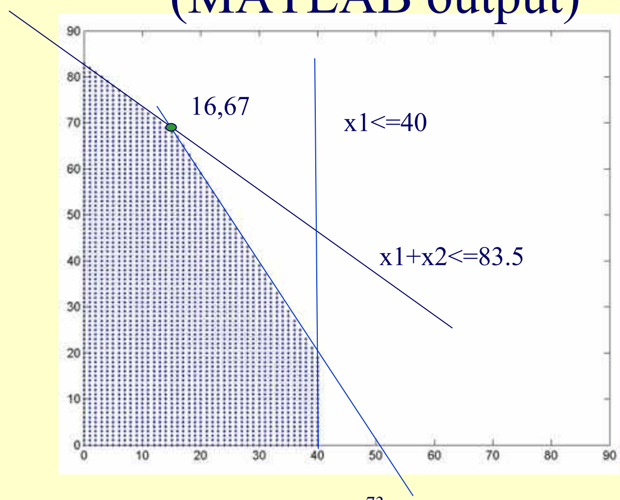
The Geometry of the Toy Problem with Integer Constraints (Excel output)



72

The Geometry of the Toy Problem with Integer Constraints

(MATLAB output)



The Toy Problem Revisited

LINDO Formulation

```
Max 3x1+2x2        (Objective function)
s.t.
2x1+x2 <=99.7      (Finishing constraint)
 x1+ x2 <=83.5     (Carpentry constraint)
 x1     <=40       (Limited demand constraint on soldiers)
 x1     >=0        (Nonnegativity constraint on soldiers)
 x2     >=0        (Nonnegativity constraint on cars)
end
gin x1             ("gin" stands for general integer variable)
gin x2
```

Special case of binary variables (=0 or 1) to be used later, the command to make the variable x a binary variable is

inte x or

inte x

The Toy Problem Revisited

```
LP OPTIMUM FOUND AT STEP      3
OBJECTIVE VALUE =   183.199997

NEW INTEGER SOLUTION OF    182.000000     AT BRANCH      0 PIVOT       5
BOUND ON OPTIMUM:  182.0000
ENUMERATION COMPLETE. BRANCHES=     0 PIVOTS=       5

LAST INTEGER SOLUTION IS THE BEST FOUND
RE-INSTALLING BEST SOLUTION...

      OBJECTIVE FUNCTION VALUE

      1)      182.0000
  VARIABLE        VALUE          REDUCED COST
        X1        16.000000         -3.000000
        X2        67.000000         -2.000000
ROW   SLACK OR SURPLUS     DUAL PRICES
        2)         0.699997          0.000000
        3)         0.500000          0.000000
        4)        24.000000          0.000000
        5)        16.000000          0.000000
        6)        67.000000          0.000000

NO. ITERATIONS=       5
BRANCHES=     0 DETERM.=  1.000E    0
```

- **Why is the objective function worse?**

The Knapsack Problem

The Knapsack Problem

- Suppose there are n items to be considered for inclusion in a "knapsack"
- Each item has a certain per unit value to the traveler who is packing the knapsack
- Each item has a per unit weight that contributes to the overall weight of the knapsack
- There is a limitation on the total weight that can be carried
- **Objective:** Maximize the total value of what is packed into the knapsack subject to the total weight limitation
- We can use IP to solve this problem
- Why a "trivial solution" is not apparent
- Generalizations of this problem beyond a knapsack

77

The Knapsack Problem

- Definitions: For j=1,2,...,n let
- $v_j>0$ be the value per unit for item j
- $w_j>0$ be the weight per unit of item j
- W be the total weight limitation
- x_j is the number of units of item j included in the knapsack

$$max \sum_{j=1}^{n} v_j x_j$$

$$s.t.$$

$$\sum_{j=1}^{n} w_j x_j \le W, x_j \ge 0, \text{integer}, \ j = 1,\ldots,n$$

78

The Knapsack Problem: Project Selection Example

- Definitions: For j=1,2,...,n let
- v_j>0 be the value if project j is selected
- c_j>0 be the cost of selecting project j
- B be the total budget available limitation
- x_j =1 if project is selected, 0 otherwise

$$\max \sum_j v_j x_j$$

$$s.t.$$

$$\sum_j c_j x_j \le B$$

$$x_j \in \{0,1\},\ j = 1,2,\ldots,n$$

The Knapsack Problem: Project Selection Example

Project	Value	Cost
1	25.99	13.69
2	17.56	12.31
3	21.33	15
4	14.34	12.73
5	24.37	13.69
6	24.37	12.31
7	21.33	15
8	11.65	12.73
9	25.27	13.69
10	21.33	12.31
11	18.46	15
12	11.65	12.73
13	25.27	13.69
14	17.56	12.31
15	21.33	15

- 15 projects, total budget of 100
- Why not just fund all 15?
- Total cost is 202.2, therefore, need the right subset
- Don’t just pick the least costly ones, want high value ones too
- “Cherry-picking” solution is not always the best
- Use Excel to solve this integer program (IP)

Scheduling Under Limited Resources Using Integer Programming

The Knapsack Problem

- Suppose there are n items to be considered for inclusion in a knapsack
- Each item has a certain per unit value to the traveler who is packing the knapsack
- Each item has a per unit weight that contributes to the overall weight of the knapsack
- There is a limitation on the total weight that can be carried
- **Objective:** Maximize the total value of what is packed into the knapsack subject to the total weight limitation
- We can use IP to solve this problem

Scheduling Under Limited Resources Using Integer Programming

The Knapsack Problem

- Definitions: For j=1,2,...,n let
- $c_j>0$ be the value per unit for item j
- $w_j>0$ be the weight per unit of item j
- W be the total weight limitation
- x_j is the number of units of item j included in the knapsack

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

$$s.t.$$

$$\sum_{j=1}^{n} w_j x_j \le W, x_j \ge 0, \text{integer}, \ j = 1,\ldots,n$$

Scheduling Under Limited Resources Using Integer Programming

The Knapsack Problem for scheduling

- Each activity to be scheduled at a specific instant is modeled as an item to be included in the knapsack
- The composition of the activities in a scheduling window (certain amount of time) is viewed as the knapsack
- Note: for activity scheduling, only one unit of each activity (item) can be included in the schedule at any given scheduling time; in general, can't schedule the activity twice at the same time!
- The knapsack problem for activity scheduling is done at each and every scheduling time t
- The objective is to schedule as many activities of high priority as possible while satisfying precedence relationships w/o exceeding the resources

83

Scheduling Under Limited Resources Using Integer Programming

The Knapsack Problem for scheduling

- Definitions:
- z_t =overall performance of the schedule generated at time t
- p_j= the priority value for activity j
- t= current time of scheduling
- x_{jt}= binary variable, =1 if activity j is scheduled at time t, =0 o/w
- S_t=set of activities eligible for scheduling at time t
- k= number of different resource types
- r_{ij}=units of resource type i required by activity j
- R_{it}=units of resource type i available at time t

$$\max z_t = \sum_{j \in S_t} p_j x_{jt}$$

$$s.t. \sum_{j \in S_t}^{n} r_{ij} x_{jt} \le R_{it}, i = 1, \ldots, k, x_{jt} \in \{0,1\}, j \in S_t \text{ for all } t$$

84

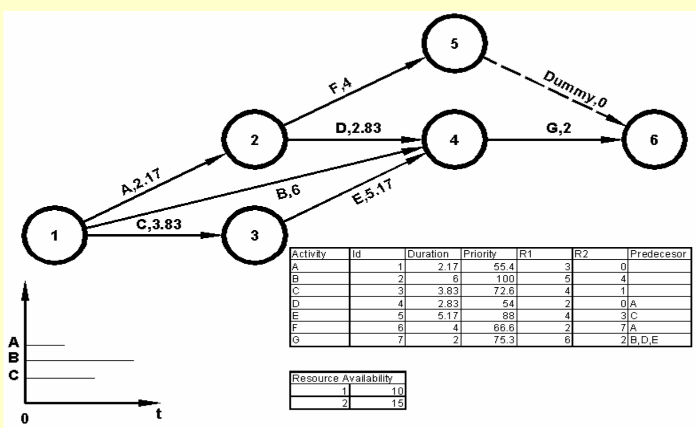
Scheduling Under Limited Resources Using Integer Programming

The Knapsack Problem for scheduling

- Note that the next scheduling time, t, for the knapsack problem is given as the minimum of {the finishing times of the scheduled and unfinished activities}
- Now let's consider a specific example (see page 303)
- Note that the priority values do not change from time one time period to the next (*fixed prioritization* scheme)
- Can also consider knapsack problem with changing priority values (*variable prioritization* scheme)
- Can use specialized methods to solve this problem

85

Scheduling Under Limited Resources Using Integer Programming



Activity	Id	Duration	Priority	R1	R2	Predecesor
A	1	2.17	55.4	3	0	
B	2	6	100	5	4	
C	3	3.83	72.6	4	1	
D	4	2.83	54	2	0	A
E	5	5.17	88	4	3	C
F	6	4	66.6	2	7	A
G	7	2	75.3	6	2	B,D,E

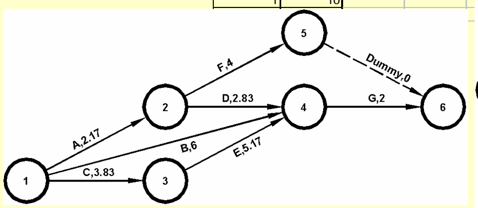
Resource Availability	
1	10
2	15

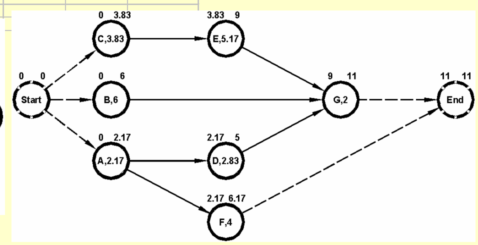
86

Example from Badiru-Pulat

Activity	Id	Duration	Priority	R1	R2	Predecesor
A	1	2.17	55.4	3	0	
B	2	6	100	5	4	
C	3	3.83	72.6	4	1	
D	4	2.83	54	2	0	A
E	5	5.17	88	4	3	C
F	6	4	66.6	2	7	A
G	7	2	75.3	6	2	B,D,E

Resource Availability	
1	10



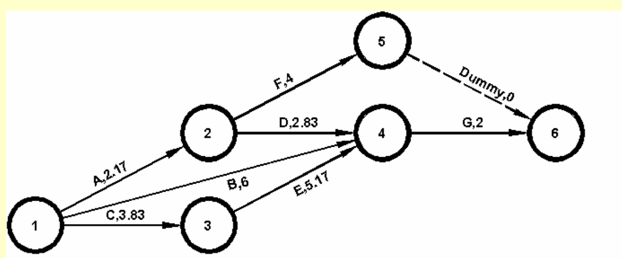


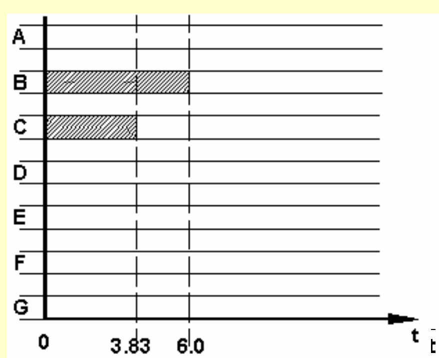
t=0

$S_0=\{A,B,C\}$

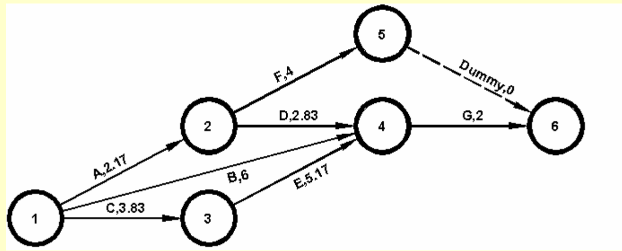
```
max 55.4xa+100xb+72.6xc
s.t.
R1)3xa+5xb+4xc<=10
R2)0xa+4xb+1xc<=15
end
int xa
int xb
int xc

OBJECTIVE FUNCTION VALUE
172.6000
XA = 0.000000
XB = 1.000000
XC = 1.000000
```





t=3.83

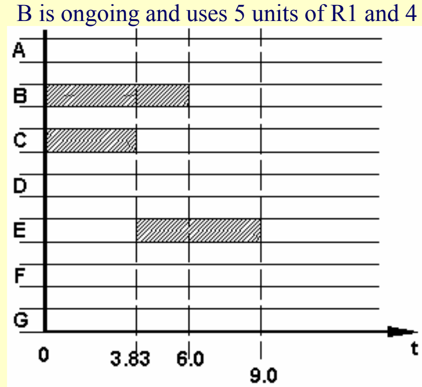


$S_{3.83}$={A,E}

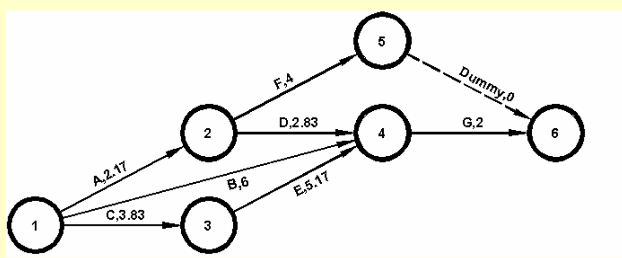
```
max 55.4xa+88xe
s.t.
R1)3xa+4xe<= 5
R2)0xa+3xe<=11
end
int xa
int xe
```

```
 OBJECTIVE FUNCTION VALUE
88.00000
XA = 0.000000
XE = 1.000000
```

B is ongoing and uses 5 units of R1 and 4 of R2



t=6.0

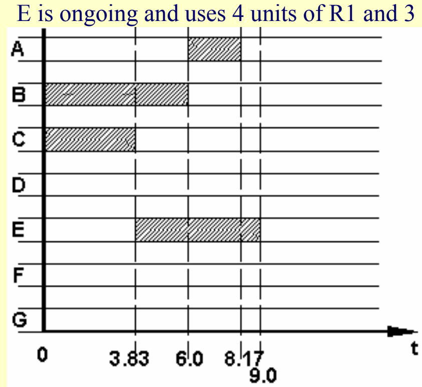


S_6={A}

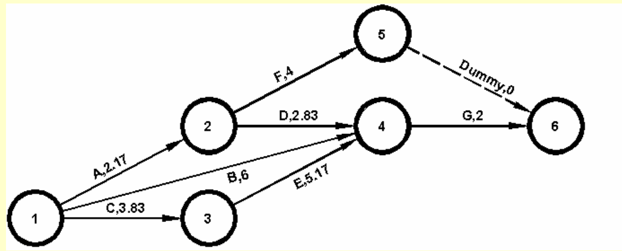
```
max 55.4xa
s.t.
R1)3xa<= 6
end
int xa
```

```
 OBJECTIVE FUNCTION VALUE
55.00000
XA = 1.000000
```

E is ongoing and uses 4 units of R1 and 3 of R2



t=8.17

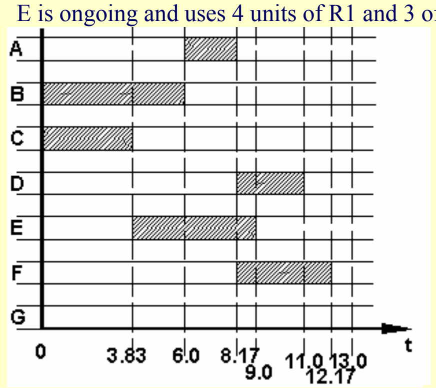


$S_{8.17}$={D,F}

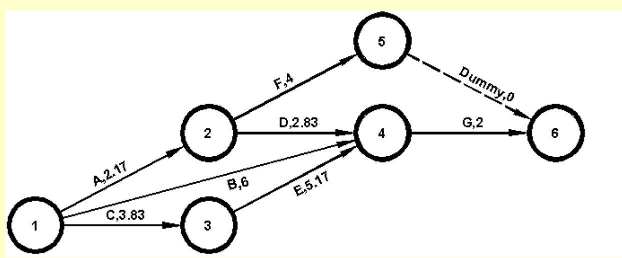
```
max 54xd+66.6xf
s.t.
R1)2xd+2xf<= 6
R1)0xd+7xf<=12
end
int xd
int xf
```

```
 OBJECTIVE FUNCTION VALUE
120.00000
Xd = 1.000000
Xf = 1.000000
```

E is ongoing and uses 4 units of R1 and 3 of R2



t=9

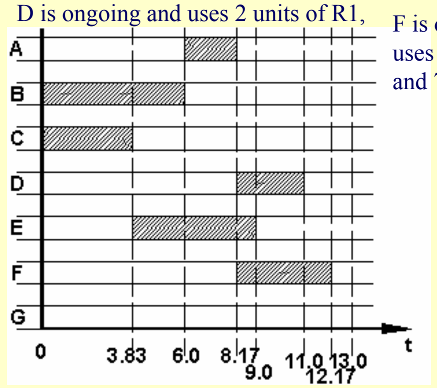


S_9={}

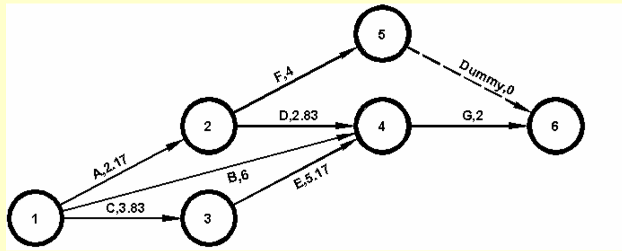
Can't do anything!

D is ongoing and uses 2 units of R1,

F is ongoing and uses 2 units of R1, and 7 units of R2



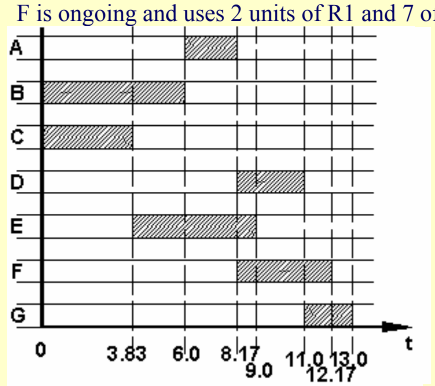
t=11



S_{11}={G}

```
max 75.3xg
s.t.
R1)6xg<= 8
R1)2xg<= 8
end
int xg
```

F is ongoing and uses 2 units of R1 and 7 of R2




```
 OBJECTIVE FUNCTION VALUE
73.00000
Xg = 1.000000
```


Some Other Uses of Integer (Binary) Programming

- Fixed charge problems
- Either-or constraints
- If-then constraints

Fixed Charge Problems: An Example

- Manufacturing project involving 3 products (1, 2, 3)
- Each product requires that an appropriate type of machinery be available
- Rental rates for machines:
- Product 1 machine: $200/week
- Product 2 machine: $150/week
- Product 3 machine: $100/week
- Also raw materials and labor required for each product

	Labor (hours)	Raw Materials (lbs)
Product 1	3	4
Product 2	2	3
Product 3	6	4

Fixed Charge Problems

- Each week 150 hours of labor and 160 lbs of raw materials are available
- Also need to consider the variable unit cost and selling price for each product
- Want an IP whose solution will maximize the weekly net profits
- **Variables?**

	Sales price	Variable Cost
Product 1	$12	$6
Product 2	$8	$4
Product 3	$15	$8

Fixed Charge Problems

- Variables
- Let x_i= number of units of product i manufactured in a given week (i=1,2,3)

  Let y_i= 1 if any product i is manufactured, =0 o/w (i=1,2,3)

  (note: if $x_i>0$ ➔ $y_i=1$ and if $x_i=0$ ➔ $y_i=0$)
- **Constraints?**
- **Objective function?**

Fixed Charge Problems

- Let x_i= number of units of product i manufactured in a given week (i=1,2,3)
- Let y_i= 1 if any product i is manufactured, =0 o/w (i=1,2,3)

  (note: if $x_i>0$ ➔ $y_i=1$ and if $x_i=0$ ➔ $y_i=0$)
- Let M_1, M_2, M_3 be 3 large positive numbers

$$\max\ 6x_1+4x_2+7x_3-200y_1-150y_2-100y_3$$ ← Net Revenues – rental costs

$s.t.$

$$3x_1+2x_2+6x_3\le 150$$ ← Labor constraint

$$4x_1+3x_2+4x_3\le 160$$ ← Raw materials constraint

$$\left.\begin{aligned} x_1 &\le M_1y_1\\ x_2 &\le M_2y_2\\ x_3 &\le M_3y_3 \end{aligned}\right\}$$ ← Constraints that ensure if $x_i>0$ ➔ $y_i=1$

note that if $x_i=0$ ➔ $y_i=0$ or =1 but we get $y_i=0$ at an optimal solution (for cost reasons)

max x values give M values: 40, 53, 25

$$x_1,x_2,x_3\ge 0,\text{integer}$$

$$y_1,y_2,y_3\in\{0,1\}$$

Fixed Charge Problems

max 6x1+4x2+7x3-200y1-150y2-100y3

s.t.

3x1+2x2+6x3<=150

4x1+3x2+4x3<=160

x1-40y1 <=0

x2-53y2<=0

x3-25y3<=0

end

gin x1

gin x2

gin x3

int y1

int y2

int y3

Solution:
Optimal net profit of $75
x3=25
y3=1
all other variables =0

Either-Or Formulation

- Let's modify the manufacturing problem from before
- If any of product 1 produced, then it must be at least 25 units, i.e., if $x_1>0$ ➔ $x_1>=25$ or equivalently either $x_1<=0$ or $x_1>=25$
- If any of product 2 produced, then it must be at least 26 units i.e., if $x_2>0$ ➔ $x_2>=26$ or equivalently either $x_2<=0$ or $x_2>=26$
- If any of product 3 produced, then it must be at least 27 units i.e., if $x_3>0$ ➔ $x_3>=27$ or equivalently either $x_3<=0$ or $x_3>=27$
- More general setting, we have two constraints of the form:
- $f(x_1,x_2,..., x_n)<=0$ and $g(x_1,x_2,..., x_n)<=0$
- We want to ensure that <u>at least one</u> of these constraints is satisfied
- For N a large enough positive number and z a binary variable, this is ensured with the following two constraints

$$f(x_1, x_2, \dots, x_n) \le Nz$$
$$g(x_1, x_2, \dots, x_n) \le N(1-z)$$

$$\max\ 6x_1 + 4x_2 + 7x_3 - 200y_1 - 150y_2 - 100y_3$$ ← Net Revenues – rental costs

$s.t.$

$3x_1 + 2x_2 + 6x_3 \le 150$ ← Labor constraint

$4x_1 + 3x_2 + 4x_3 \le 160$ ← Raw materials constraint

$$\left.\begin{array}{l} x_1 \le M_1 y_1 \\ x_2 \le M_2 y_2 \\ x_3 \le M_3 y_3 \end{array}\right\}$$

← Constraints that ensure if $x_i>0$ → $y_i=1$

note that if $x_i=0$ → $y_i=0$ or $=1$ but we get $y_i=0$ at an optimal solution (for cost reasons)

max x values give M values: 40, 53, 25

$$\left.\begin{array}{l} x_1 \le N_1 z_1 \\ 25 - x_1 \le N_1(1-z_1) \\ x_2 \le N_2 z_2 \\ 26 - x_2 \le N_2(1-z_2) \\ x_3 \le N_3 z_3 \\ 27 - x_3 \le N_3(1-z_3) \end{array}\right\}$$

← Either-or constraints, N values need to be chosen suitably large, for example take all N's to be equal to 100

$x_1, x_2, x_3 \ge 0$, integer

$y_1, y_2, y_3, z_1, z_2, z_3 \in \{0,1\}$

101

LINDO Formulation and Solution to Either-Or Problem

```
max 6x1+4x2+7x3-200y1-
150y2-100y3
s.t.
3x1+2x2+6x3<=150
4x1+3x2+4x3<=160
x1-40y1 <=0
x2-53y2<=0
x3-25y3<=0
x1-100z1<=0
-x1+100z1<=75
x2-100z2<=0
-x2+100z2<=74
x3-100z3<=0
-x3+100z3<=73
end
gin x1
gin x2
gin x3
int y1
int y2
int y3
int z1
int z2
int z3
```

Solution:
Optimal profit of $62
x2=53
y2=1
z2=1
all other variables =0

102

If-Then Formulation

- Let's modify the original manufacturing problem one more time (taking away the modifications from before)
- We want the restriction: if the sum of products 2 and 3 exceed 24 units (true for both cases considered so far), then at least 30 of product 1 must be manufactured (union rules?)
- More generally, we will be considering

$$\text{if } f(x_1, x_{2,} \ldots, x_n) > 0 \text{ then } g(x_1, x_{2,} \ldots, x_n) \geq 0$$

$$\text{if } f(x_1, x_{2,} \ldots, x_n) \leq 0 \text{ then } g(x_1, x_{2,} \ldots, x_n) \geq 0 \text{ or } < 0$$

If-Then Formulation

- We can use the following constraints where N is a suitably large positive value and z is a binary variable

$$-g(x_1, x_{2,} \ldots, x_n) \leq Nz$$

$$f(x_1, x_{2,} \ldots, x_n) \leq N(1-z)$$

- In our example we can take

$$x_1 - 30 = g(x_1, x_{2,} \ldots, x_n)$$

$$x_2 + x_3 - 24 = f(x_1, x_{2,} \ldots, x_n)$$

LINDO Formulation and Solution to If-Then Problem

```
max 6x1+4x2+7x3-200y1-150y2-
100y3
s.t.
3x1+2x2+6x3<=150
4x1+3x2+4x3<=160
x1-40y1 <=0
x2-53y2<=0
x3-25y3<=0

-x1-100z<=-30
x2+x3+100z<=124

end
gin x1
gin x2
gin x3
int y1
int y2
int y3
int z
```

Solution:
Optimal profit of $68
x3=24
y3=1
z=1
all other variables =0

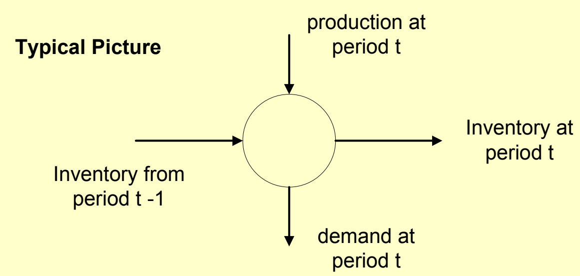
Multiperiod Production-Inventory Problem

- A Production -Inventory Problem (Periodic Review Model):
  - Time is broken up into periods:
    present period ---- period 1
    next period ---- period 2
    ….
    last period ---- period T
  - At start of each period, firm must determine how much should be produced (production capacity at each period is limited)
  - Each period's demand must be met on time from inventory or current production

Multiperiod Production-Inventory Problem

- During any period in which production occurs, a fixed cost as well as a variable per unit cost is incurred
- There is limited storage capacity
  - Limit on end-of-period inventory
- A per unit holding cost is incurred on each period's ending inventory
- The firm's goal is to minimize the total cost of meeting on time, the demands for periods 1,2,…,T

Multiperiod Production-Inventory Problem



Multiperiod Production-Inventory Problem

- Demand schedule

Month	Number of units (tons)
1	1
2	3
3	2
4	4

At start of each month, how much to produce during the current month (questions for company)

- Production (if production occurs): Max of 5 units/period to be produced
  Set up cost \$3
  Variable cost \$1/unit produced.
- Inventory : Holding Costs (at the end of the month): \$0.50/unit
  Capacity limitations: max of 4 units, initial inventory=0

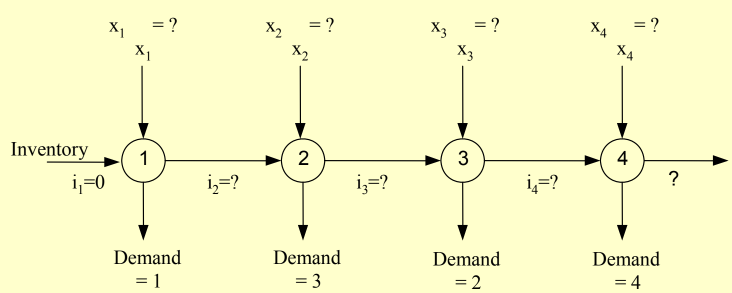
Want: Production schedule that will meet all demands on time and will minimize the sum of production and holding costs for the 4 months. (Assume inventory at start of month 1 is 0 units)

Variables, constraints, objective function?

109

Multiperiod Production-Inventory Problem

Production schedule (try without LP/IP first)



Can relate to a shortest path problem in a network as follows.

110

Multiperiod Production-Inventory Problem

Production schedule IP

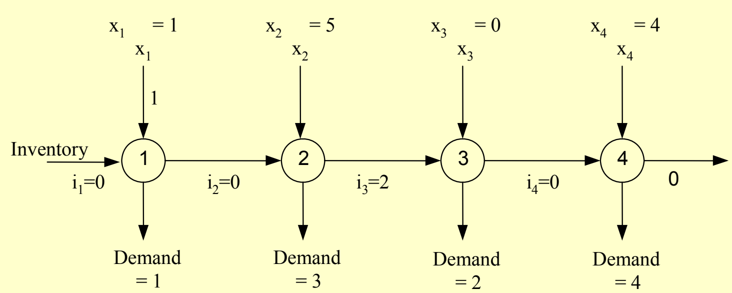
```
Min 1x1+1x2+1x3+1x4      ! Production variable costs
+ 3y1+3y2+3y3+3y4        ! Production fixed costs
+0.5i1+0.5i2+0.5i3+0.5i4 ! Inventory costs
s.t.
d1=1                     ! Demand for period 1
d2=3                     ! Demand for period 2
d3=2                     ! Demand for period 3
d4=4                     ! Demand for period 4
i1=0                     ! Initial inventory
i2<=4 i3<=4 i4<=4        ! Inventory capacity
x1<=5 x2<=5 x3<=5 x4<=5  ! Production capacity
i1+x1-d1 -i2=0           ! Period 1 material balance
i2+x2-d2 -i3=0           ! Period 1 material balance
i3+x3-d3 -i4=0           ! Period 1 material balance
i4+x4-d4    =0           ! Period 1 material balance

x1-100000y1 <=0
     ! Consistency between production and set-up cost varibles
x2-100000y2 <=0
     ! Consistency between production and set-up cost varibles
x3-100000y3 <=0
     ! Consistency between production and set-up cost varibles
x4-100000y4 <=0
     ! Consistency between production and set-up cost varibles
end              ! Nonnegativity implied by LINDO
inte y1
inte y2
inte y3
inte y4
```

111

Multiperiod Production-Inventory Problem

⇒20 is the minimum cost for the 4 months optimal schedule.

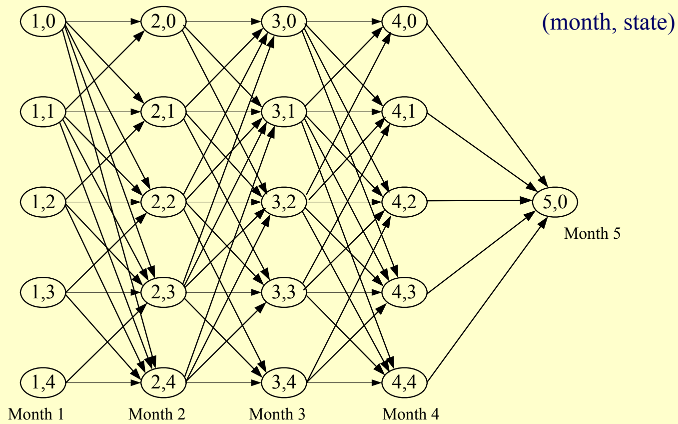


Can relate to a shortest path problem in a network as follows.

112

Multiperiod Production-Inventory Problem

- Representation of Inventory Example



113

Assignment Problems (You are now a Project Manager)

- Want to optimally assign "workers" to "tasks"
- Suppose we have n tasks to be performed by n workers
- The cost of worker i performing task j is given as c_{ij}
- Remarks:
  1. If the number of tasks to be done is greater than the number of workers, we add *dummy workers* to balance the problem
  2. If the number of workers is greater than the number of tasks, we add *dummy tasks* to balance the problem
  3. If no problem of overlapping a worker's time, the time can be split between projects and each segment can be modeled as a separate resource (can consider partial allocation of resource units to multiple tasks)

114

The Assignment Problem

- CPM can be used to control the project duration
- Such methods do not however, assign resources to project tasks
- Now we consider the assignment problem , a formulation to optimally assign workers to tasks
- Suppose we have n tasks to be performed by n workers
- The cost of worker i performing task j is given as c_{ij}
- Remarks:
  1. If the number of tasks to be done is greater than the number of works, we add *dummy workers* to balance the problem
  2. If the number of workers is greater than the number of tasks, we add *dummy tasks* to balance the problem
  3. If no problem of overlapping a worker's time, the time can be split between projects and each segment can be modeled as a separate resource (can consider partial allocation of resource units to multiple tasks)

The Assignment Problem

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij}$$ ←minimize total costs

$s.t.$

$$\sum_{j=1}^{n} x_{ij} = 1, i = 1,2,\ldots,n$$ ←ith worker must get assigned to exactly 1 task

$$\sum_{i=1}^{n} x_{ij} = 1, j = 1,2,\ldots,n$$ ←jth task must get exactly 1 worker

$$x_{ij} \geq 0, i, j = 1,2,\ldots,n$$ ←nonnegative variables, binary restriction satisfied indirectly

The Assignment Problem

- Consider the following cost matrix for an assignment problem with n=5
- Select the cheapest workers by task first, **will this work?**

	Task 1	Task 2	Task 3	Task 4	Task5
Worker 1	$200	$400	$500	$100	$400
Worker 2	$400	$700	$800	$1,100	$500
Worker 3	$300	$900	$800	$1,000	$500
Worker 4	$100	$300	$500	$100	$400
Worker 5	$700	$100	$200	$100	$200

The Assignment Problem

min 200x11 + 400x12 + 500x13 + 100x14 + 400x15+ 400x21 + 700x22 + 800x23 + 1100x24 + 500x25+ 300x31 + 900x32 + 800x33 + 1000x34 + 500x35+ 100x41 + 300x42 + 500x43 + 100x44 + 400x45+700x51 + 100x52 + 200x53 + 100x54 + 200x55

s.t.

x11+x12+x13+x14+x15=1 ! worker 1

x21+x22+x23+x24+x25=1 ! worker 2

x31+x32+x33+x34+x35=1 ! worker 3

x41+x42+x43+x44+x45=1 ! worker 4

x51+x52+x53+x54+x55=1 ! worker 5

x11+x21+x31+x41+x51=1 ! task 1

x12+x22+x32+x42+x52=1 ! task 2

x13+x23+x33+x43+x53=1 ! task 3

x14+x24+x34+x44+x54=1 ! task 4

x15+x25+x35+x45+x55=1 ! task 5

! and all variables nonnegative

The Assignment Problem

- Optimal solution at a cost of $1,400 is as follows

	Task 1	Task 2	Task 3	Task 4	Task5
Worker 1					
Worker 2					
Worker 3					
Worker 4					
Worker 5					

Solving Integer Programs

- Certain classes of LPs we studied will have integer solutions so don't need to enforce integrality restrictions
- Otherwise, how can we solve integer-constrained problems?
- Many approaches, will give just two mentioned here
  - Enumeration
  - Branch-and-Bound (pure IP example)

Solving Integer Programs Using Enumeration

- For small enough problems, can just enumerate all feasible solutions
- Then pick the one(s) with the best objective function value
- When this method will work, when it won't

121

Solving Integer Programs Using Branch-and-Bound for Pure IPs

- Basic Idea: Solve a sequence of linear programming relaxations (in the form of a "tree structure") to solve original IP
- Elementary observation: if you solve the LP relaxation of a pure IP and get a solution which has just integer answers, then LP optimal is also IP optimal
- Example (from Winston)
  max z=3x1+2x2
  s.t. 3x1+x2<=6
  x1,x2 >=0, x1,x2 integer
- Feasible region? LP and IP solutions?

122

Solving Integer Programs Using Branch-and-Bound for Pure IPs

- Sample Problem:
- Production of tables and chairs
- 1 table needs 1 hour of labor & 9 square board feet of wood, $8 in profit
- 1 chair needs 1 hour of labor & 5 square board feet of wood, $5 in profit
- Currently: 6 hours of labor, 45 square board feet available
- IP to maximize profit?

123

Solving Integer Programs Using Branch-and-Bound for Pure IPs

let

x1= number of tables produced

x2= number of chairs produced

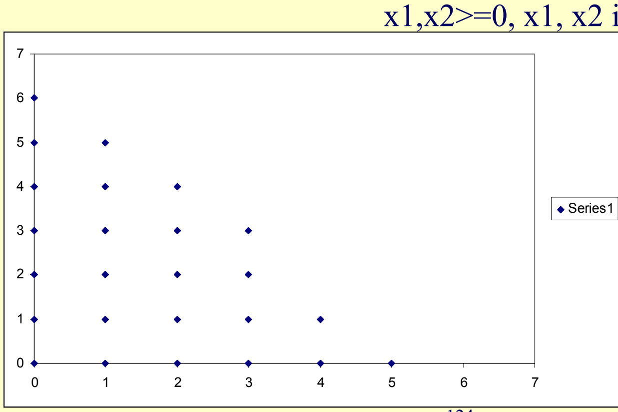
max $8x1+5x2$

s.t.

$x1+x2<=6$

$9x1+5x2<=45$

$x1,x2>=0$, x1, x2 integer



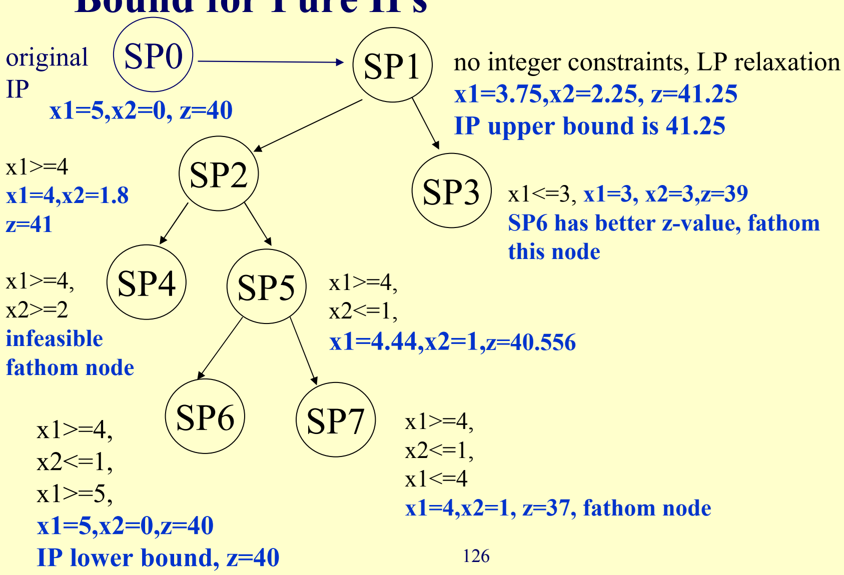
x1	x2	z
0	0	0
0	1	5
0	2	10
0	3	15
0	4	20
0	5	25
0	6	30
1	0	8
1	1	13
1	2	18
1	3	23
1	4	28
1	5	33
2	0	16
2	1	21
2	2	26
2	3	31
2	4	36
3	0	24
3	1	29
3	2	34
3	3	39
4	0	32
4	1	37
5	0	40

124

Solving Integer Programs Using Branch-and-Bound for Pure IPs

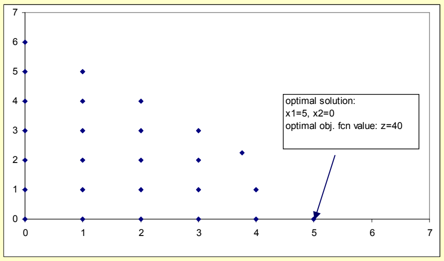
- Let's see how to get this solution with the Branch-and-Bound Technique
- 7 LP subproblems to solve

Solving Integer Programs Using Branch-and-Bound for Pure IPs



- Original problem

max 8x1+5x2
s.t.
x1+x2<=6
9x1+5x2<=45
end
gin x1
gin x2



- Subproblem 1
- remove integer constraints

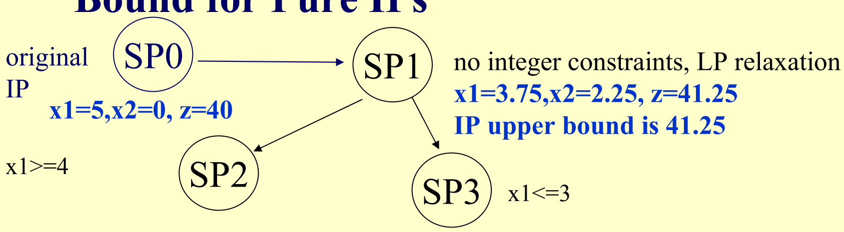
max 8x1+5x2
s.t.
x1+x2<=6
9x1+5x2<=45
end

x1=3.75,x2=2.25, z=41.25
IP upper bound is 41.25

- Conclusion: optimal z-value for IP<=optimal z-value for LP relaxation
- Upper bound for IP is 41.25
- Next step, arbitrarily choose a fractional variable (say x1) and try 2 LPs with the rounded values
- Subproblem 2: x1>=4, subproblem 3, x1<=3 (branch on x1)
- Why can't a feasible solution to the IP have 3<x1<4? The point x1=3.75 will be avoided this way (can't return to this solution), why?

127

Solving Integer Programs Using Branch-and-Bound for Pure IPs



128

- Original problem

  max 8x1+5x2

  s.t.

  x1+x2<=6

  9x1+5x2<=45

  x1,x2>=0, x1, x2 integer

- Subproblem 2
- add new constraint x1>=4

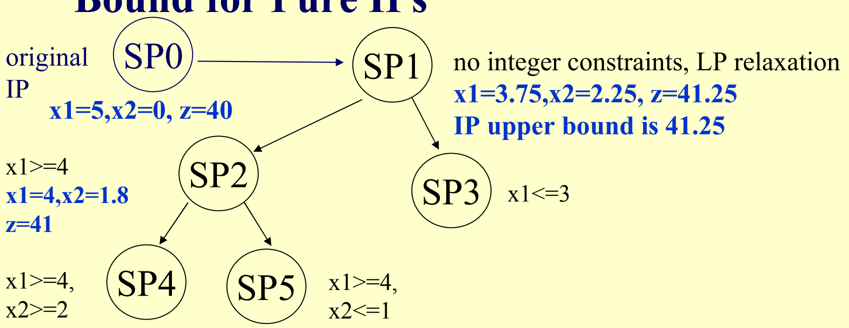
  max 8x1+5x2
  s.t.
  x1+x2<=6
  9x1+5x2<=45
  x1,x2>=0, **x1>=4**

  x1=4,x2=1.8,z=41

- Conclusion: integer solution not obtained
- Next step: arbitrarily choose a fractional variable (branch on x2) and try 2 LPs with rounded values
- Subproblem #4: x2>=2, subproblem #5: x2<=1

129

Solving Integer Programs Using Branch-and-Bound for Pure IPs



130

- Original problem

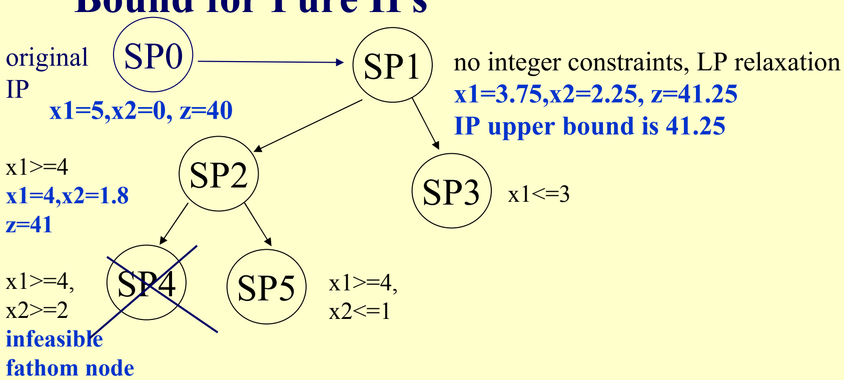
max 8x1+5x2
s.t.
x1+x2<=6
9x1+5x2<=45
x1,x2>=0, x1, x2 integer

- Subproblem 4
- add new constraint x2>=2

max 8x1+5x2
s.t.
x1+x2<=6
9x1+5x2<=45
x1>=4, x2>=2
x1,x2>=0

- Conclusion: Subproblem 4 is **infeasible**, fathom this node
- Next step: try subproblem 5

Solving Integer Programs Using Branch-and-Bound for Pure IPs



- Original problem

max 8x1+5x2
s.t.
x1+x2<=6
9x1+5x2<=45
x1,x2>=0, x1, x2 integer

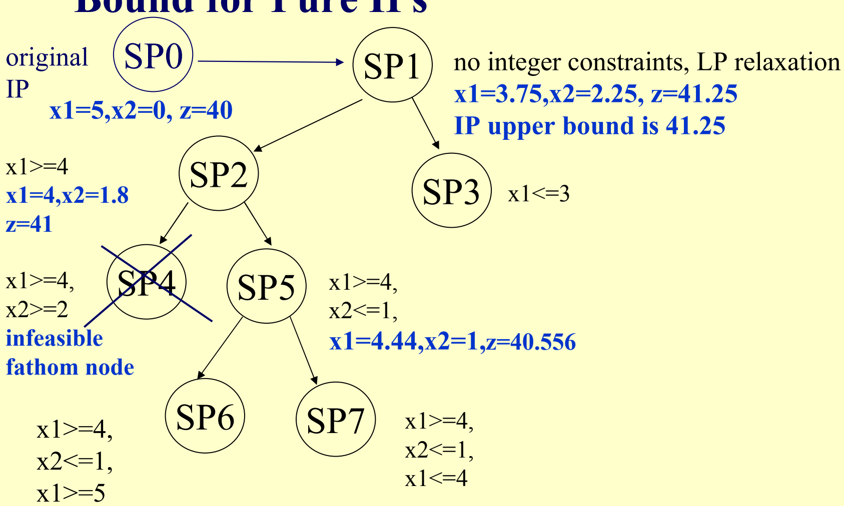
- Subproblem 5
- add new constraint x2<=1

max 8x1+5x2
s.t.
x1+x2<=6
9x1+5x2<=45
x1,x2>=0 **,x1>=4, x2<=1**

x1=4.44, x2=1,z=40.556

- Conclusion: feasible solution, still fractional though so need to branch again
- Next step: branch on x1, Subproblem 6: add x1>=5, subproblem 7: add x1<=4
- Could also try subproblem 3 but we are using a LIFO rule (LIFO=last in first out)

Solving Integer Programs Using Branch-and-Bound for Pure IPs



- Original problem

  max 8x1+5x2
  s.t.
  x1+x2<=6
  9x1+5x2<=45
  x1,x2>=0, x1, x2 integer

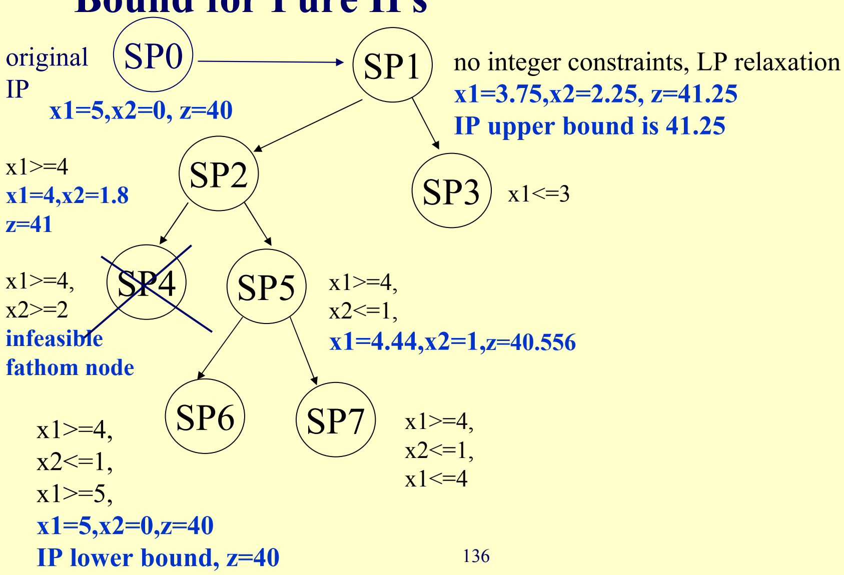
- Subproblem 6
- add new constraint x1>=5

  max 8x1+5x2
  s.t.
  x1+x2<=6
  9x1+5x2<=45
  x1,x2>=0, **x1>=4, x2<=1, x1>=5**

x1=5,x2=0,z=40
IP lower bound, z=40

- Conclusion: candidate solution
- IP lower bound is now 40
- Next step: try remaining node relating to subproblem 7

Solving Integer Programs Using Branch-and-Bound for Pure IPs



- Original problem

max 8x1+5x2
s.t.
x1+x2<=6
9x1+5x2<=45
x1,x2>=0, x1, x2 integer

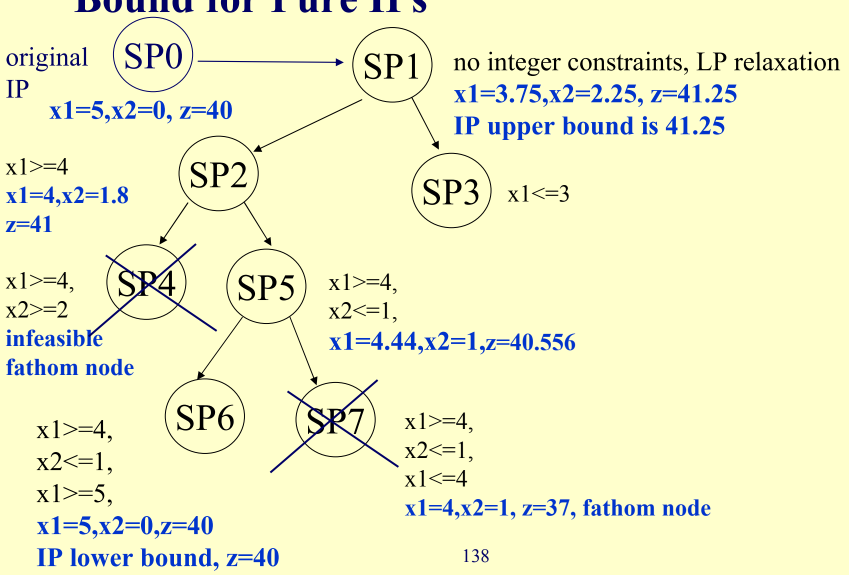
- Subproblem 7
- add new constraint x1<=4

max 8x1+5x2
s.t.
x1+x2<=6
9x1+5x2<=45
x1,x2>=0, **x1>=4, x2<=1, x1<=4**

x1=4,x2=1,z=37
IP lower bound, z=37

- Conclusion: further branching on subproblem7 cannot yield a feasible integer solution>37, why?
- Next step: fathom this node and try subproblem 3

Solving Integer Programs Using Branch-and-Bound for Pure IPs



- Original problem

  max 8x1+5x2
  s.t.
  x1+x2<=6
  9x1+5x2<=45
  x1,x2>=0, x1, x2 integer

- Subproblem 3
- add new constraint x1<=3

  max 8x1+5x2
  s.t.
  x1+x2<=6
  9x1+5x2<=45
  x1,x2>=0, **x1<=3**

  x1=3,x2=3, z=39

- Conclusion: Not better than the current lower bound of 40 from subproblem 6
- Fathom this node
- No nodes left to try- done!

Solving Integer Programs Using Branch-and-Bound for Pure IPs