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

Outline

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

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

---

### In tabular form

<table>
<thead>
<tr>
<th>Activity</th>
<th>Predecessor</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>n/a</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>n/a</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>n/a</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>B, D, E</td>
<td>2</td>
</tr>
</tbody>
</table>

### 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)** + t_F = 2 + 4 = 6

---

**Copyright 2008, Dr. Steven A. Gabriel**
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

\[ \begin{align*}
1 & \rightarrow 2 \\
3 & \rightarrow 4 \\
5 & \\
\end{align*} \]

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

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

Why do we use “max” of the predecessor EC’s in rule 2?

Rule 3: \( EC(i) = ES(i) + t_i \)
Rule 4: \( EC(\text{Project}) = EC(n) \)
Rule 5: \( LC(\text{Project}) = EC(\text{Project}) \) “zero project slack convention” (unless otherwise stated for example, see Rule 6)
Rule 6: \( LC(\text{Project}) = T_p \)
Rule 7: \( LC(j) = \min_i \{ S(j) \} \{ LS(i) \} \)
Rule 8: \( LS(j) = LC(j) - t_j \)

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.
  \[ TS(j) = LC(j) - EC(j) \text{ or } TS(j) = LS(j) - ES(j) \]

- 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.
  \[ FS(j) = \min_{i \in S(j)} \{ ES(i) - EC(j) \} \]

- Let’s consider the sample network relative to critical activities and slack times.

---

**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.
**CPM-Forward Pass Example AON**

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

**Sample Computations**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Predecessor</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>B,D,E</td>
<td>2</td>
</tr>
</tbody>
</table>

**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) + t_F = 2 + 4 = 6**

---

**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.
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.
  \[ TS(j) = LC(j) - EC(j) \]
  \[ TS(j) = LS(j) - ES(j) \]
- 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.
  \[ FS(j) = \min_{i \in S(j)} \{ES(i) - EC(j)\} \]
- 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**

<table>
<thead>
<tr>
<th>Activity</th>
<th>Duration (Days)</th>
<th>ES</th>
<th>EC</th>
<th>LS</th>
<th>LC</th>
<th>TS</th>
<th>FS</th>
<th>Critical Activity?</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>6-2=4 Min(2,2)=2=0 No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>6</td>
<td>0</td>
<td>6</td>
<td>3</td>
<td>9</td>
<td>9-6=3 Min(9,6)=6=3 No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>0</td>
<td>4</td>
<td>4-4=0 Min(4,4)=4=0 YES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>9-5=4 Min(9,5)=4=4 No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>5</td>
<td>4</td>
<td>9</td>
<td>4</td>
<td>9</td>
<td>9-9=0 Min(9,9)=9=0 YES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>7</td>
<td>11</td>
<td>11-6=5 Min(11,6)=5 No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>9</td>
<td>11</td>
<td>9</td>
<td>11</td>
<td>11=11=0 Min(11,11)=11=0 YES</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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)

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)
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 without delaying the completion of the project.
- Let $t_{ij}$ 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 without 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 without 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 (otherwise an activity would precede itself).

![AOA Network Structure Diagram]

- 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 no 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

<table>
<thead>
<tr>
<th>Activity</th>
<th>Predecessor</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>B,D,E</td>
<td>2</td>
</tr>
</tbody>
</table>

Small Sample Project AOA

<table>
<thead>
<tr>
<th>Activity</th>
<th>Predecessor</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>C</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>D</td>
<td>A</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>C</td>
<td>5</td>
</tr>
<tr>
<td>F</td>
<td>A</td>
<td>4</td>
</tr>
<tr>
<td>G</td>
<td>B,D,E</td>
<td>2</td>
</tr>
</tbody>
</table>
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

Min $x_5 - x_1$

s.t.

A) $x_2 \geq x_1 + 2$
B) $x_4 \geq x_1 + 6$
C) $x_3 \geq x_1 + 4$
D) $x_4 \geq x_2 + 3$
E) $x_4 \geq x_3 + 5$
F) $x_5 \geq x_2 + 4$
G) $x_5 \geq x_4 + 2$

Variables unrestricted in sign
Using Linear Programming to Find a Critical Path

Min \( x_5 - x_1 \)

s.t.

A) \( x_2 - x_1 \geq 2 \)
B) \( x_4 - x_1 \geq 6 \)
C) \( x_3 - x_1 \geq 4 \)
D) \( x_4 - x_2 \geq 3 \)
E) \( x_4 - x_3 \geq 5 \)
F) \( x_5 - x_2 \geq 4 \)
G) \( x_5 - x_4 \geq 2 \)

end

free \( x_1 \)
free \( x_2 \)
free \( x_3 \)
free \( x_4 \)
free \( x_5 \)

OBJECTIVE FUNCTION VALUE 11.00000

VARIABLE VALUE REDUCED COST
\( X_5 \) 11.000000 0.000000
\( X_1 \) 0.000000 0.000000
\( X_2 \) 6.000000 0.000000
\( X_4 \) 9.000000 0.000000
\( X_3 \) 4.000000 0.000000

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

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
<th>Immediate Predecessors</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Dad, Mom, and son arrive home in the same car</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>Dad and Mom change clothes</td>
<td>a</td>
</tr>
<tr>
<td>c</td>
<td>Start watching TV</td>
<td>a</td>
</tr>
<tr>
<td>d</td>
<td>Mom washes the food*</td>
<td>b</td>
</tr>
<tr>
<td>e</td>
<td>Dad sets the table</td>
<td>b</td>
</tr>
<tr>
<td>f</td>
<td>Son does homework**</td>
<td>c</td>
</tr>
<tr>
<td>g</td>
<td>Mom fixes salad</td>
<td>d</td>
</tr>
<tr>
<td>h</td>
<td>The family eats dinner</td>
<td>c,g</td>
</tr>
<tr>
<td>i</td>
<td>Dad loads the dishwasher</td>
<td>b</td>
</tr>
<tr>
<td>j</td>
<td>Mom checks son’s homework</td>
<td>d,b</td>
</tr>
<tr>
<td>k</td>
<td>Son practices (insert music of choice here)</td>
<td>b</td>
</tr>
<tr>
<td>l</td>
<td>All go to son’s basketball game</td>
<td>d,k</td>
</tr>
<tr>
<td>m</td>
<td>All wash up and go to bed</td>
<td>j,m</td>
</tr>
</tbody>
</table>

*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.

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

<table>
<thead>
<tr>
<th>Activity</th>
<th>Description</th>
<th>Immediate Predecessors</th>
<th>Duration (Days)</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Train workers</td>
<td>-</td>
<td>6</td>
</tr>
<tr>
<td>b</td>
<td>Purchase raw materials</td>
<td>-</td>
<td>9</td>
</tr>
<tr>
<td>c</td>
<td>Produce product 1</td>
<td>a,b</td>
<td>8</td>
</tr>
<tr>
<td>d</td>
<td>Produce product 2</td>
<td>a,b</td>
<td>7</td>
</tr>
<tr>
<td>e</td>
<td>Test product 2</td>
<td>d</td>
<td>10</td>
</tr>
<tr>
<td>f</td>
<td>Assemble products 1 and 2 into new product 3</td>
<td>c,e</td>
<td>12</td>
</tr>
</tbody>
</table>
Finding the Critical Path in an AOA Project Network for Introducing a New Product

\[ \begin{align*}
\text{min } & x_6 - x_1 \\
\text{s.t. } & x_3 - x_1 \geq 6 \quad \text{! arc (1,3)} \\
& x_2 - x_1 \geq 9 \quad \text{! arc (1,2)} \\
& x_5 - x_3 \geq 8 \quad \text{! arc (3,5)} \\
& x_4 - x_3 \geq 7 \quad \text{! arc (3,4)} \\
& x_5 - x_4 \geq 10 \quad \text{! arc (4,5)} \\
& x_6 - x_5 \geq 12 \quad \text{! arc (5,6)} \\
& x_3 - x_2 \geq 0 \quad \text{! arc (2,3)} \\
\end{align*} \]

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

Excel version of this LP?

Project completed in 38 days

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 \( LT(i) \).

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...)

Copyright 2008, Dr. Steven A. Gabriel
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

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

If c “crashable” activities, there are $2^c$ possible crash strategies, why?

Suppose we can crash 6 of the 7 activities $\Rightarrow 2^6=64$ possible crash strategies.

There are 12 of the 64 strategies shown here.

---

### Project Crashing and Time-Cost Analysis

**Time-Cost Tradeoff Curve**

- Cost ($): 2500, 3000, 3500, 4000, 4500
- Duration (days): 6, 7, 8, 9, 10, 11, 12

Copyright 2008, Dr. Steven A. Gabriel
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

\[
\begin{align*}
\text{min } & \quad 10A + 20B + 3C + 30D + 40E + 50F \\
\text{s.t. } & \quad A \leq 5 \\
& \quad B \leq 5 \\
& \quad C \leq 5 \\
& \quad D \leq 5 \\
& \quad E \leq 5 \\
& \quad F \leq 5 \\
& \quad x_3 - x_1 + A \geq 6 \quad ! \text{arc (1,3)} \\
& \quad x_2 - x_1 + B \geq 9 \quad ! \text{arc (1,2)} \\
& \quad x_5 - x_3 + C \geq 8 \quad ! \text{arc (3,5)} \\
& \quad x_4 - x_3 + D \geq 7 \quad ! \text{arc (3,4)} \\
& \quad x_5 - x_4 + E \geq 10 \quad ! \text{arc (4,5)} \\
& \quad x_6 - x_5 + F \geq 12 \quad ! \text{arc (5,6)} \\
& \quad x_3 - x_2 \geq 0 \quad ! \text{arc (3,4)} \\
& \quad x_6 - x_1 \leq 25 \quad ! \text{at most 25 days} \\
\end{align*}
\]

End

! could have variables free or not
! free x_1 x_2 x_3 x_4 x_5 x_6

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

Copyright 2008, Dr. Steven A. Gabriel
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).

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?
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?**

```
A
```

```
B
```

**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?**

```
A
```

```
B
```

Copyright 2008, Dr. Steven A. Gabriel
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

![Diagram of A, B, C activities with lead-lag constraints and starting and finishing times]

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

  ![Diagram of A, B, C activities with lead-lag constraints and starting and finishing times]

- 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!
Before B’s time is reduced

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

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 120 and 125 months 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 skyrocket 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**

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

**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 $x_i$ be the time for node $i$. The associated LP model is thus:

$$\text{min } x_{12} - x_1$$

s.t.

A) $x_2 - x_1 \geq 5$ ! A
B) $x_3 - x_2 \geq 12$ ! B
C) $x_4 - x_3 \geq 12$ ! C
D) $x_5 - x_3 \geq 6$ ! D
E) $x_5 - x_4 \geq 12$ ! E
F) $x_6 - x_5 \geq 12$ ! F
G) $x_7 - x_6 \geq 48$ ! G
H) $x_8 - x_6 \geq 24$ ! H
DU1) $x_9 - x_8 \geq 0$ ! dummy arc
DU2) $x_9 - x_7 \geq 0$ ! dummy arc
I) $x_{10} - x_9 \geq 12$ ! I
J) $x_{11} - x_{10} \geq 36$ ! J
K) $x_{12} - x_{11} \geq 1$ ! K
Objective function value

1) 150.0000

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>VALUE</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X12</td>
<td>150.0000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X1</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X2</td>
<td>5.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X3</td>
<td>17.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X4</td>
<td>29.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X5</td>
<td>41.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X6</td>
<td>53.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X7</td>
<td>101.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X8</td>
<td>77.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X9</td>
<td>101.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X10</td>
<td>113.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X11</td>
<td>149.000000</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

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 uncashed 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.
New model is thus:
\[
\text{min } x_{12} - x_{A1} \\
\text{s.t.}
\]
A) \( x_{B2} - x_{B1} \geq 5 \uparrow A \)
B) \( x_{C2} - x_{C1} \geq 12 \uparrow B \)
C) \( x_{D2} - x_{D1} \geq 6 \uparrow D \)
SS1) \( x_{B1} - x_{A1} \geq 1 \uparrow SS(A,B)=1 \)
SS2) \( x_{C1} - x_{B1} \geq 1 \uparrow SS(B,C)=1 \)
SS3) \( x_{D1} - x_{C1} \geq 1 \uparrow SS(C,D)=1 \)
A2) \( x_{B2} - x_{A2} \geq 0 \) \uparrow A2 before B2
B2) \( x_{C2} - x_{B2} \geq 0 \) \uparrow B2 before C2
C2) \( x_{4} - x_{C2} \geq 0 \) \uparrow C2 before 4
D2) \( x_{5} - x_{D2} \geq 0 \) \uparrow D2 before 5
E) \( x_{5} - x_{4} \geq 12 \uparrow E \)
F) \( x_{6} - x_{5} \geq 12 \uparrow F \)
G) \( x_{7} - x_{6} \geq 48 \uparrow G \)
H) \( x_{8} - x_{6} \geq 24 \uparrow H \)
DU1) \( x_{9} - x_{8} \geq 0 \) \uparrow dummy arc
DU2) \( x_{9} - x_{7} \geq 0 \) \uparrow dummy arc
I) \( x_{10} - x_{9} \geq 12 \uparrow I \)
J) \( x_{11} - x_{10} \geq 36 \uparrow J \)
K) \( x_{12} - x_{11} \geq 1 \uparrow K \)

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

OBJECTIVE FUNCTION VALUE

<table>
<thead>
<tr>
<th>VARIABLE</th>
<th>VALUE</th>
<th>REDUCED COST</th>
</tr>
</thead>
<tbody>
<tr>
<td>X12</td>
<td>135.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>XA1</td>
<td>0.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>XA2</td>
<td>14.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>XB2</td>
<td>14.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>XB1</td>
<td>1.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>XC2</td>
<td>14.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>XC1</td>
<td>2.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>XD2</td>
<td>26.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>XD1</td>
<td>3.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X4</td>
<td>14.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X5</td>
<td>26.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X6</td>
<td>38.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X7</td>
<td>86.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X8</td>
<td>86.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X9</td>
<td>86.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X10</td>
<td>98.000000</td>
<td>0.000000</td>
</tr>
<tr>
<td>X11</td>
<td>134.000000</td>
<td>0.000000</td>
</tr>
</tbody>
</table>

Copyright 2008, Dr. Steven A. Gabriel

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

Copyright 2008, Dr. Steven A. Gabriel
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
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)
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
tinte x  or
tinte 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

Copyright 2008, Dr. Steven A. Gabriel
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

Definitions: For \( j = 1, 2, \ldots, n \) 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

\[
\begin{align*}
\text{max} & \quad \sum_{j=1}^{n} v_j x_j \\
\text{s.t.} & \quad \sum_{j=1}^{n} w_j x_j \leq W, x_j \geq 0, \text{integer, } j = 1, \ldots, n
\end{align*}
\]
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

\[
\begin{align*}
\text{max} & \quad \sum_j v_j x_j \\
\text{s.t.} & \quad \sum_j c_j x_j \leq B \\
& \quad x_j \in \{0,1\}, j = 1,2,...,n
\end{align*}
\]

The Knapsack Problem: Project Selection Example

- 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

---

**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

\[
\begin{align*}
\text{max} \quad & \sum_{j=1}^{n} c_j x_j \\
\text{s.t.} \quad & \sum_{j=1}^{n} w_j x_j \leq W, \quad x_j \geq 0, \text{integer}, \quad j = 1, \ldots, n
\end{align*}
\]
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

\[
\begin{align*}
&\text{max } z_t = \sum_{j \in S_t} p_j x_{jt} \\
&s.t. \sum_{j \in S_t} r_{ij} x_{jt} \leq R_i, i = 1, \ldots, k, x_{jt} \in \{0,1\}, j \in S_t \text{ for all } t
\end{align*}
\]

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 $j$
- $R_i =$ units of resource type $i$ available at time $t$

\[
\begin{align*}
&\text{max } z_t = \sum_{j \in S_t} p_j x_{jt} \\
&s.t. \sum_{j \in S_t} r_{ij} x_{jt} \leq R_i, i = 1, \ldots, k, x_{jt} \in \{0,1\}, j \in S_t \text{ for all } t
\end{align*}
\]
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
Example from Badiru-Pulat

<table>
<thead>
<tr>
<th>ACTIVITY</th>
<th>ID</th>
<th>DURATION</th>
<th>PRIORITY</th>
<th>R1</th>
<th>R2</th>
<th>PREDECESSOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>2.17</td>
<td>55.4</td>
<td>3</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>B2</td>
<td>2</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>3.83</td>
<td>72.6</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4</td>
<td>2.83</td>
<td>54</td>
<td>2</td>
<td>0</td>
<td>A</td>
</tr>
<tr>
<td>E</td>
<td>5</td>
<td>5.17</td>
<td>88</td>
<td>4</td>
<td>3</td>
<td>C</td>
</tr>
<tr>
<td>F6</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>G7</td>
<td>7</td>
<td>2</td>
<td>7</td>
<td>5</td>
<td>3</td>
<td>B, D, E</td>
</tr>
</tbody>
</table>

Resource Availability

\[ t=0 \]

\[ S_0 = \{A,B,C\} \]

\[ \text{max } 55.4xa + 100xb + 72.6xc \]

\[ \text{s.t.} \]

\[ R1) 3xa + 5xb + 4xc \leq 10 \]

\[ R2) 0xa + 4xb + 1xc \leq 15 \]

\[ \text{end} \]

\[ \text{int } xa \]

\[ \text{int } xb \]

\[ \text{int } xc \]

OBJECTIVE FUNCTION VALUE

\[ 172.6000 \]

XA = 0.000000

XB = 1.000000

XC = 1.000000
**t=3.83**

\[ S_{t=3.83} = \{A, E\} \]

\[
\begin{align*}
\text{max} & \quad 55.4x_a + 88x_e \\
\text{s.t.} & \quad R1) 3x_a + 4x_e \leq 5 \\
& \quad R2) 0x_a + 3x_e \leq 11 \\
\end{align*}
\]

**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**

\[ S_{t=6.0} = \{A\} \]

\[
\begin{align*}
\text{max} & \quad 55.4x_a \\
\text{s.t.} & \quad R1) 3x_a \leq 6 \\
\end{align*}
\]

**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.

$R1) 2xd + 2xf \leq 6$

$R1) 0xd + 7xf \leq 12$

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 $R2$

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

Can’t do anything!

$t=9$

$S_9 = \{\}$

Copyright 2008, Dr. Steven A. Gabriel
$S_1 = \{G\}$

max $75.3x_g$

s.t.

$R1)6x_g \leq 8$

$R1)2x_g \leq 8$

end

int $x_g$

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

<table>
<thead>
<tr>
<th></th>
<th>Labor (hours)</th>
<th>Raw Materials (lbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Product 2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Product 3</td>
<td>6</td>
<td>4</td>
</tr>
</tbody>
</table>

**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?**

<table>
<thead>
<tr>
<th></th>
<th>Sales price</th>
<th>Variable Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product 1</td>
<td>$12</td>
<td>$6</td>
</tr>
<tr>
<td>Product 2</td>
<td>$8</td>
<td>$4</td>
</tr>
<tr>
<td>Product 3</td>
<td>$15</td>
<td>$8</td>
</tr>
</tbody>
</table>
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 $x_i > 0 \Rightarrow y_i = 1$ and if $x_i = 0 \Rightarrow y_i = 0$)

• Constraints?
• Objective function?

Let $M_1, M_2, M_3$ be 3 large positive numbers

$$\max 6x_1 + 4x_2 + 7x_3 - 200y_1 - 150y_2 - 100y_3 \quad \text{Net Revenues - rental costs}$$

s.t.

$$3x_1 + 2x_2 + 6x_3 \leq 150 \quad \text{Labor constraint}$$
$$4x_1 + 3x_2 + 4x_3 \leq 160 \quad \text{Raw materials constraint}$$

$$x_1 \leq M_1 y_1$$
$$x_2 \leq M_2 y_2$$
$$x_3 \leq M_3 y_3$$

Constraints that ensure if $x_i > 0 \Rightarrow y_i = 1$

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

max $x$ values give $M$ values: 40, 53, 25
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 x1>0 \implies x1>=25 or equivalently either x1<=0 or x1>=25
• If any of product 2 produced, then it must be at least 26 units i.e.,
  if x2>0 \implies x2>=26 or equivalently either x2<=0 or x2>=26
• If any of product 3 produced, then it must be at least 27 units i.e.,
  if x3>0 \implies x3>=27 or equivalently either x3<=0 or x3>=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 at least one 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
\[
f(x_1,x_2,...,x_n) \leq Nz \\
g(x_1,x_2,...,x_n) \leq N(1−z)
\]
max \(6x_1 + 4x_2 + 7x_3 - 200y_1 - 150y_2 - 100y_3\) \(\Leftarrow\) Net Revenues – rental costs
\[s.t.\]
\(3x_1 + 2x_2 + 6x_3 \leq 150\) \(\Leftarrow\) Labor constraint
\(4x_1 + 3x_2 + 4x_3 \leq 160\) \(\Leftarrow\) Raw materials constraint
\[x_1 \leq M_1 y_1\]
\[x_2 \leq M_2 y_2\]
\[x_3 \leq M_3 y_3\]
\(x_1 \leq N_1 z_1\)
\(25 - x_1 \leq N_1 (1 - z_1)\)
\(x_2 \leq N_2 z_2\)
\(26 - x_2 \leq N_2 (1 - z_2)\)
\(x_3 \leq N_3 z_3\)
\(27 - x_3 \leq N_3 (1 - z_3)\)
\[x_1, x_2, x_3 \geq 0, \text{integer}\]
\[y_1, y_2, y_3, z_1, z_2, z_3 \in \{0, 1\}\]

Constraints that ensure if \(x_i > 0\) \(\Rightarrow y_i = 1\)
note that if \(x_i = 0\) \(\Rightarrow y_i = 0 \text{ or } 1\) but we get \(y_i = 0\) at an optimal solution (for cost reasons)
max \(x\) values give \(M\) values: 40, 53, 25

Either-or constraints, \(N\) values need to be chosen suitably large, for example take all \(N\)’s to be equal to 100

LINDO Formulation and Solution to Either-Or Problem

max \(6x_1 + 4x_2 + 7x_3 - 200y_1 - 150y_2 - 100y_3\)
\[s.t.\]
\(3x_1 + 2x_2 + 6x_3 \leq 150\)
\(4x_1 + 3x_2 + 4x_3 \leq 160\)
\(x_1 - 40y_1 = 0\)
\(x_2 - 53y_2 = 0\)
\(x_3 - 25y_3 = 0\)
\(x_1 - 100z_1 = 0\)
\(-x_1 + 100z_1 = 75\)
\(x_2 - 100z_2 = 0\)
\(-x_2 + 100z_2 = 74\)
\(x_3 - 100z_3 = 0\)
\(-x_3 + 100z_3 = 73\)
end

gin x1
gin x2
gin x3
int y1
int y2
int y3
int z1
int z2
int z3

Solution:
Optimal profit of $62
\(x_2 = 53\)
\(y_2 = 1\)
\(z_2 = 1\)
all other variables = 0
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

\[
\begin{align*}
\text{max} & \quad 6x_1 + 4x_2 + 7x_3 - 200y_1 - 150y_2 - 100y_3 \\
\text{s.t.} & \quad 3x_1 + 2x_2 + 6x_3 \leq 150 \\
& \quad 4x_1 + 3x_2 + 4x_3 \leq 160 \\
& \quad x_1 - 40y_1 \leq 0 \\
& \quad x_2 - 53y_2 \leq 0 \\
& \quad x_3 - 25y_3 \leq 0 \\
& \quad -x_1 - 100z \leq -30 \\
& \quad x_2 + x_3 + 100z \leq 124 \\
\end{align*}
\]

end

gin x_1

gin x_2

gin x_3

int y_1

int y_2

int y_3

int z

Solution:

Optimal profit of $68

x_3 = 24

y_3 = 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

Typical Picture

- Inventory from period t-1
- Demand at period t
- Production at period t
- Inventory at period t
Multiperiod Production-Inventory Problem

<table>
<thead>
<tr>
<th>Month</th>
<th>Number of units (tons)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

- **Demand schedule**: 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?

---

Multiperiod Production-Inventory Problem

Production schedule (try without LP/IP first)

Can relate to a shortest path problem in a network as follows.
Multiperiod Production-Inventory Problem

Production schedule IP

Min $1x_1+1x_2+1x_3+1x_4$ ! Production variable costs
$+3y_1+3y_2+3y_3+3y_4$ ! Production fixed costs
$+0.5i_1+0.5i_2+0.5i_3+0.5i_4$ ! Inventory costs

s.t.

- $d_1=1$ ! Demand for period 1
- $d_2=3$ ! Demand for period 2
- $d_3=2$ ! Demand for period 3
- $d_4=4$ ! Demand for period 4
- $i_1=0$ ! Initial inventory
- $i_2<=4$ $i_3<=4$ $i_4<=4$ ! Inventory capacity
- $x_1<=5$ $x_2<=5$ $x_3<=5$ $x_4<=5$ ! Production capacity
- $i_1+x_1-d_1-i_2=0$ ! Period 1 material balance
- $i_2+x_2-d_2-i_3=0$ ! Period 1 material balance
- $i_3+x_3-d_3-i_4=0$ ! Period 1 material balance
- $i_4+x_4-d_4=0$ ! Period 1 material balance
- $x_1-100000y_1 <=0$ ! Consistency between production and set-up cost variables
- $x_2-100000y_2 <=0$ ! Consistency between production and set-up cost variables
- $x_3-100000y_3 <=0$ ! Consistency between production and set-up cost variables
- $x_4-100000y_4 <=0$ ! Consistency between production and set-up cost variables

end

inte $y_1$
inte $y_2$
inte $y_3$
inte $y_4$

Nonnegativity implied by LINDO

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

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

Can relate to a shortest path problem in a network as follows.
**Multiperiod Production-Inventory Problem**

- Representation of Inventory Example

![Diagram showing inventory representation with months and states](image)

**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)
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)

\[
\begin{align*}
\text{minimize total costs} \\
\sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \\
\text{s.t.} \\
\sum_{j=1}^{n} x_{ij} = 1, i = 1,2,\ldots,n \quad \text{ith worker must get assigned to exactly 1 task} \\
\sum_{i=1}^{n} x_{ij} = 1, j = 1,2,\ldots,n \quad \text{jth task must get exactly 1 worker} \\
x_{ij} \geq 0, i, j = 1,2,\ldots,n \quad \text{nonnegative variables, binary restriction satisfied indirectly}
\end{align*}
\]
Consider the following cost matrix for an assignment problem with n=5.

Select the cheapest workers by task first, will this work?

<table>
<thead>
<tr>
<th></th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worker 1</td>
<td>$200</td>
<td>$400</td>
<td>$500</td>
<td>$100</td>
<td>$400</td>
</tr>
<tr>
<td>Worker 2</td>
<td>$400</td>
<td>$700</td>
<td>$800</td>
<td>$1,100</td>
<td>$500</td>
</tr>
<tr>
<td>Worker 3</td>
<td>$300</td>
<td>$900</td>
<td>$800</td>
<td>$1,000</td>
<td>$500</td>
</tr>
<tr>
<td>Worker 4</td>
<td>$100</td>
<td>$300</td>
<td>$500</td>
<td>$100</td>
<td>$400</td>
</tr>
<tr>
<td>Worker 5</td>
<td>$700</td>
<td>$100</td>
<td>$200</td>
<td>$100</td>
<td>$200</td>
</tr>
</tbody>
</table>

The Assignment Problem

\[
\begin{align*}
\text{min} & \quad 200x_{11} + 400x_{12} + 500x_{13} + 100x_{14} + 400x_{15} + 400x_{21} + 700x_{22} + 800x_{23} + 1100x_{24} + 500x_{25} + 300x_{31} + 900x_{32} + 800x_{33} + 1000x_{34} + 500x_{35} + 100x_{41} + 300x_{42} + 500x_{43} + 100x_{44} + 400x_{45} + 700x_{51} + 100x_{52} + 200x_{53} + 100x_{54} + 200x_{55} \\
\text{s.t.} & \quad x_{11} + x_{12} + x_{13} + x_{14} + x_{15} = 1 \quad \text{! worker 1} \\
& \quad x_{21} + x_{22} + x_{23} + x_{24} + x_{25} = 1 \quad \text{! worker 2} \\
& \quad x_{31} + x_{32} + x_{33} + x_{34} + x_{35} = 1 \quad \text{! worker 3} \\
& \quad x_{41} + x_{42} + x_{43} + x_{44} + x_{45} = 1 \quad \text{! worker 4} \\
& \quad x_{51} + x_{52} + x_{53} + x_{54} + x_{55} = 1 \quad \text{! worker 5} \\
& \quad x_{11} + x_{21} + x_{31} + x_{41} + x_{51} = 1 \quad \text{! task 1} \\
& \quad x_{12} + x_{22} + x_{32} + x_{42} + x_{52} = 1 \quad \text{! task 2} \\
& \quad x_{13} + x_{23} + x_{33} + x_{43} + x_{53} = 1 \quad \text{! task 3} \\
& \quad x_{14} + x_{24} + x_{34} + x_{44} + x_{54} = 1 \quad \text{! task 4} \\
& \quad x_{15} + x_{25} + x_{35} + x_{45} + x_{55} = 1 \quad \text{! task 5} \\
\end{align*}
\]

! and all variables nonnegative
### The Assignment Problem

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

<table>
<thead>
<tr>
<th></th>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Worker 1</td>
<td></td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Worker 2</td>
<td></td>
<td></td>
<td></td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Worker 3</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Worker 4</td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Worker 5</td>
<td></td>
<td>✔</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 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

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

- 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)
  \[
  \text{max } z = 3x_1 + 2x_2 \\
  \text{s.t. } 3x_1 + x_2 \leq 6 \\
  x_1, x_2 \geq 0, x_1, x_2 \text{ integer}
  \]
- Feasible region? LP and IP solutions?
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?

\[
\begin{align*}
\text{max } & \quad 8x_1 + 5x_2 \\
\text{s.t. } & \quad x_1 + x_2 \leq 6 \\
& \quad 9x_1 + 5x_2 \leq 45 \\
& \quad x_1, x_2 \geq 0, x_1, x_2 \text{ integer}
\end{align*}
\]
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
• 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?

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

original IP
  SP0
  no integer constraints, LP relaxation
  SP1
  x1=3.75, x2=2.25, z=41.25
  IP upper bound is 41.25
  SP2
  x1=5, x2=0, z=40
  SP3
  x1=4

SP1
  SP2
  SP3
  SP0
  original IP
  no integer constraints, LP relaxation
  x1=3.75, x2=2.25, z=41.25
  IP upper bound is 41.25

Copyright 2008, Dr. Steven A. Gabriel
• Original problem
  \[ \text{max } 8x_1+5x_2 \]
  \[ \text{s.t.} \]
  \[ x_1+x_2 \leq 6 \]
  \[ 9x_1+5x_2 \leq 45 \]
  \[ x_1, x_2 \geq 0, \text{ integer} \]

• Subproblem 2
  \[ \text{add new constraint } x_1 \geq 4 \]
  \[ \text{max } 8x_1+5x_2 \]
  \[ \text{s.t.} \]
  \[ x_1+x_2 \leq 6 \]
  \[ 9x_1+5x_2 \leq 45 \]
  \[ x_1, x_2 \geq 0, \ x_1 \geq 4 \]
  \[ x_1=4, x_2=1.8, z=41 \]

• 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 \geq 2, subproblem #5: x2 \leq 1

---

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

SP0 original IP
  \[ x_1=5, x_2=0, z=40 \]
  \[ x_1 \geq 4 \]
  \[ x_1=4, x_2=1.8, z=41 \]

SP1 no integer constraints, LP relaxation
  \[ x_1=3.75, x_2=2.25, z=41.25 \]
  IP upper bound is 41.25

SP2
  \[ x_1 \geq 4, x_2 \geq 1.8, z=41 \]

SP3
  \[ x_1 \leq 3 \]

SP4
  \[ x_1 \geq 4, x_2 \geq 2 \]

SP5
  \[ x_1 \geq 4, x_2 \leq 1 \]
• 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 IP
x1=5, x2=0, z=40

SP0
SP1
no integer constraints, LP relaxation
x1=3.75, x2=2.25, z=41.25
IP upper bound is 41.25

SP2
x1>=4
x1=4, x2=1.8
z=41

SP3
x1<=3

SP4
SP5
x1>=4, x2=2
infeasible
fathom node

x1>=4, x2<=1

---

Copyright 2008, Dr. Steven A. Gabriel
• Original problem
  \[ \text{max } 8x_1 + 5x_2 \]
  \[ \text{s.t. } x_1 + x_2 \leq 6 \]
  \[ 9x_1 + 5x_2 \leq 45 \]
  \[ x_1, x_2 \geq 0, x_1, x_2 \text{ integer} \]

• Subproblem 5
  \[ \text{add new constraint } x_2 \leq 1 \]
  \[ \text{max } 8x_1 + 5x_2 \]
  \[ \text{s.t. } x_1 + x_2 \leq 6 \]
  \[ 9x_1 + 5x_2 \leq 45 \]
  \[ x_1, x_2 \geq 0, x_1 \geq 4, x_2 \leq 1 \]
  \[ x_1 = 4.44, x_2 = 1, z = 40.556 \]

• Conclusion: feasible solution, still fractional though so need to branch again
• Next step: branch on \(x_1\), Subproblem 6: add \(x_1 \geq 5\), subproblem 7: add \(x_1 \leq 4\)
• 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**

<table>
<thead>
<tr>
<th>Original IP</th>
<th>SP0</th>
<th>SP1</th>
<th>SP2</th>
<th>SP3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>(x_1 = 5, x_2 = 0, z = 40)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x_1 \geq 4)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(x_1 = 4, x_2 = 1.8, z = 41)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SP4</th>
<th>SP5</th>
<th>SP6</th>
<th>SP7</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(x_1 \geq 4, x_2 \leq 2)</td>
<td>(x_1 \geq 4, x_2 \leq 1)</td>
<td>(x_1 \geq 4, x_2 \leq 1, z = 40.556)</td>
<td>(x_1 \geq 4, x_2 \leq 1, x_1 \leq 4)</td>
</tr>
</tbody>
</table>

no integer constraints, LP relaxation
\[x_1 = 3.75, x_2 = 2.25, z = 41.25\]
IP upper bound is 41.25

---

Copyright 2008, Dr. Steven A. Gabriel
• Original problem
  \[
  \begin{align*}
  \text{max } & \quad 8x_1 + 5x_2 \\
  \text{s.t. } & \quad x_1 + x_2 \leq 6 \\
  & \quad 9x_1 + 5x_2 \leq 45 \\
  & \quad x_1, x_2 \geq 0, \ x_1, x_2 \text{ integer}
  \end{align*}
  \]

• Subproblem 6
  \[
  \begin{align*}
  \text{max } & \quad 8x_1 + 5x_2 \\
  \text{s.t. } & \quad x_1 + x_2 \leq 6 \\
  & \quad 9x_1 + 5x_2 \leq 45 \\
  & \quad x_1, x_2 \geq 0, \ x_1 \geq 4, \ x_2 \leq 1, \ x_1 \geq 5
  \end{align*}
  \]

  \[x_1=5, x_2=0, z=40\]
  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 IP

- **SP0**
  - \(x_1=5, x_2=0, z=40\)
- **SP1**
  - no integer constraints, LP relaxation
  - \(x_1=3.75, x_2=2.25, z=41.25\)
  - IP upper bound is 41.25

- **SP2**
  - \(x_1 \geq 4, x_2 \leq 1.8, z=41\)

- **SP3**
  - \(x_1 \leq 3\)

- **SP4**
  - \(x_1 \geq 4, x_2 = 2\)
  - infeasible
  - fathom node

- **SP5**
  - \(x_1 \geq 4, x_2 \leq 1, x_1 = 4.44, x_2 = 1, z = 40.556\)

- **SP6**
  - \(x_1 \geq 4, x_2 \leq 1, x_1 \geq 5\)

- **SP7**
  - \(x_1 \geq 4, x_2 \leq 1, x_1 \leq 4\)
  - \(x_1=5, x_2=0, z=40\)
  - IP lower bound, \(z=40\)
• Original problem
max \( 8x_1 + 5x_2 \)
s.t.
\( x_1 + x_2 \leq 6 \)
\( 9x_1 + 5x_2 \leq 45 \)
\( x_1, x_2 \geq 0, \ x_1, \ x_2 \text{ integer} \)

• Subproblem 7
add new constraint \( x_1 \leq 4 \)
max \( 8x_1 + 5x_2 \)
s.t.
\( x_1 + x_2 \leq 6 \)
\( 9x_1 + 5x_2 \leq 45 \)
\( x_1, x_2 \geq 0, \ x_1 = 4, \ x_2 \leq 1, \ x_1 \leq 4 \)
\( x_1 = 4, x_2 = 1, z = 37 \)
IP lower bound, \( z = 37 \)

• Conclusion: further branching on subproblem 7 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**

<table>
<thead>
<tr>
<th>Original IP</th>
<th>SP0</th>
<th>SP1</th>
<th>SP2</th>
<th>SP3</th>
<th>SP4</th>
<th>SP5</th>
<th>SP6</th>
<th>SP7</th>
</tr>
</thead>
<tbody>
<tr>
<td>x1\geq4, x2\geq0, z=40</td>
<td>no integer constraints, LP relaxation</td>
<td>x1=3.75, x2=2.25, z=41.25</td>
<td>IP upper bound is 41.25</td>
<td>x1\leq3</td>
<td>x1\geq4, x2\leq1, x1=4.44, x2=1, z=40.556</td>
<td>infeasible fathom node</td>
<td>x1\geq4, x2\leq1, x1\leq4</td>
<td>x1=4, x2=1, z=40, fathom node</td>
</tr>
</tbody>
</table>

IP lower bound, \( z = 40 \)
• Original problem

\[
\begin{align*}
\text{max } & 8x_1 + 5x_2 \\
\text{s.t. } & x_1 + x_2 \leq 6 \\
& 9x_1 + 5x_2 \leq 45 \\
& x_1, x_2 \geq 0, \ x_1, \ x_2 \text{ integer}
\end{align*}
\]

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

• Subproblem 3

\[
\begin{align*}
\text{max } & 8x_1 + 5x_2 \\
\text{s.t. } & x_1 + x_2 \leq 6 \\
& 9x_1 + 5x_2 \leq 45 \\
& x_1, x_2 \geq 0, \ x_1 \leq 3 \\
& x_1=3, x_2=3, z=39
\end{align*}
\]