**Network Models Notes:**

*Shortest Path Problem*

Find the shortest path from node 1 to node m. (Where m is the terminal node.)

Let (i,j) be an arc from node i to node j.

Decision Variables: xij = 1 if (i,j) is in the shortest path; 0 otherwise.  
  
Data: cij = cost to traverse arc (i, j)

Think of shipping one unit from node 1 to node m.

Formulation:  
  
s.t.  
  


However, the **A** matrix is unimodular and **b** is all integer; therefore, we can replace the binary constraints with:

0 ≤ xij ≤ 1 and we will still get an integer solution.

Also if cij ≥ 0 ; then we can replace with xij ≥ 0  since we will not use more than one xij.

And we will end up with a binary answer using LP methods. Similar to assignment problems and transportation problem (integer); again because of unimodular **A** matrix and integer **b**.

Dual:  
Max w1 - wm  
s.t.  
wi - wj ≤ cij   
wi ~ unrestricted 

Luckily, under certain conditions we don't have to solve the above formulation to find the solution to the shortest path problem. We can use Dijkstra's Algorithm if all arc lengths are nonnegative. Dijkstra's Algorithm is a *label setting* algorithm. Whereas, if there were negative costs, then we may have to use a *label correcting* algorithm.

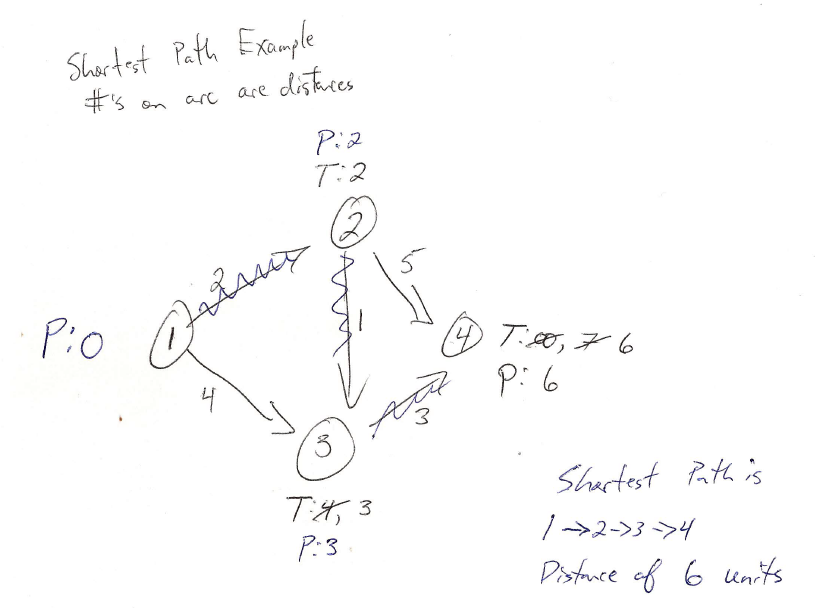
Dijkstra's Algorithm:  
Step 0: Set node 1's permanent label to 0.  
Step 1: Set each node that is connected to node 1 by a single arc with a temporary label equal to c1j (where j is the node being labeled). Set a temporary label for all other nodes as ∞ (remember, node 1's permanent label is 0).  
Step 2: Make the smallest temporary label permanent. If there is a tie, then pick one node to make permanent (leaving the rest that are tied as temporary). Call the new node that was just made permanent, node i.  
Step 3: For each node j that has a temporary label and is connected to node i by a single arc, replace node j's temporary label with .  
Step 4: Repeat Steps 2 through 3 until all nodes have permanent labels.

Result: The permanent labels are the shortest paths from node 1 to all other nodes.

Shortest Path Problem (Solution on Next Page):



Solution:



Instances where Dijkstra's Algorithm will fail (i.e., negative arcs):

Problem 1: Problem 2:





Example: At the beginning of year 1, a new machine must be purchased. The cost of maintaining a machine i years old is given in Table 1. The cost of purchasing a machine at the beginning of each year is given in Table 2. There is no trade-in value when a machine is replaced. Your goal is to minimize the total cost (purchase plus maintenance) of having a machine for five years. Determine the years in which a new machine should be purchased.  
Table 1:  
  
Table 2:  


Network will start at beginning of year 1 and end at beginning of year 6 (e.g., end of year 5).  
Let i and j denote the nodes; where a node is the beginning of the year.

cij = (cost of purchasing at beginning of year i) + (maintenance cost incurred during years i, i+1, …, j-1)

c12 = 170,000 + 38,000 = 208,000  
c13 = 170,000 + 38,000 + 50,000 = 258,000  
c14 = 170,000 + 38,000 + 50,000 + 97,000 = 355,000  
c15 = 170,000 + 38,000 + 50,000 + 97,000 + 182,000 = 537,000  
c16 = 170,000 + 38,000 + 50,000 + 97,000 + 182,000 + 304,000 = 841,000  
c23 = 190,000 + 38,000 = 228,000  
c24 = 190,000 + 38,000 + 50,000 = 278,000  
c25 = 190,000 + 38,000 + 50,000 + 97,000 = 375,000  
c26 = 190,000 + 38,000 + 50,000 + 97,000 + 182,000 = 557,000  
c34 = 210,000 + 38,000 = 248,000  
c35 = 210,000 + 38,000 + 50,000 = 298,000  
c36 = 210,000 + 38,000 + 50,000 + 97,000 = 395,000  
c45 = 250,000 + 38,000 = 288,000  
c46 = 250,000 + 38,000 + 50,000 = 338,000  
c56 = 300,000 + 38,000 = 338,000  


**Network Models:**

*Min Cost Network Flow Problems*

The transportation, assignment, transshipment, shortest path, maximum flow, and CPM problems are all special cases of the minimum cost network flow problem. There is an expedited solution process known as the network simplex that can be used to solve these problems.

**General Formulation:**

Indexed Sets:  
i: node set (1, …, m)  
j: node set (1, …, m)  
k: node set (1, …, m)

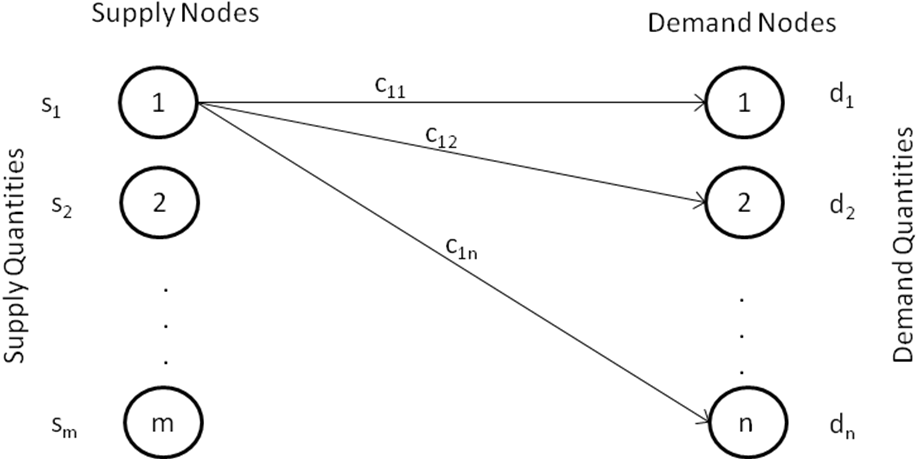
Data:  
bi: net supply (outflow - inflow) at node i [positive = source, supply, negative = sink, demand]  
c­ij­: cost of transporting 1 unit of flow from i to j  
lij: lower bound of flow on arc (i,j) (default is 0)  
uij: upper bound of flow on arc (i,j) (default is ∞)

Decision Variables:  
xij: amount of units on arc (i,j)

Objective: (Minimize Cost)  
Minimize 

S.T.  
 (flow conservation)  
 (bounds)

**Transportation Problem Formulation (as an MCNFP):**  
Also known as bipartite graph (i.e., one set of nodes are supply nodes and one set are demand nodes; where no arcs go from a supply node to another supply node and no arcs from a demand node to another demand node). Note: figure does not include all arcs - there is an arc from every supply node to every demand node.



Indexed Sets:  
i: supply node set (1, …, m)  
j: demand node set (1, …, n)

Data:  
si: supply at node i  
dj: demand at node j  
c­ij­: cost of transporting 1 unit of flow from i to j  
lij: lower bound of flow on arc (i,j) (default is 0)  
uij: upper bound of flow on arc (i,j) (default is ∞)

Decision Variables:  
xij: amount of units on arc (i,j)

Objective: (Minimize Cost)

Minimize 

S.T.  
 m constraints

 n constraints  
(to put in standard network form rewrite as [])  


Assumption:

Supply = Demand  


If supply does not equal demand, add a "dummy" supply node or a "dummy" demand node to balance problem.

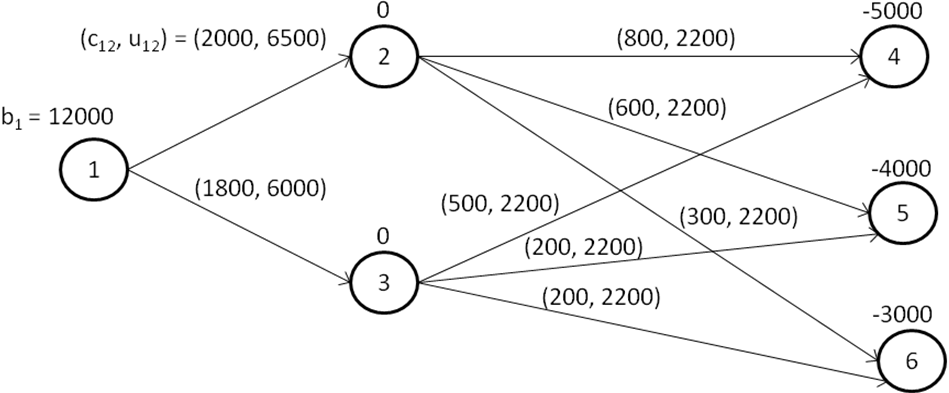
Solve using transportation simplex (usually quicker than network simplex since problem is bipartite & balanced).

Note: In order to put into standard form - you will need to rename/renumber the arcs/nodes.

**Example 1:**  
Fordco produces cars in Detroit and Dallas. The Detroit plant can produce as many as 6500 cars, and the Dallas plant can produce as many as 6000 cars. Producing a car costs $2000 in Detroit and $1800 in Dallas. Cars must be shipped to three cities. City 1 must receive 5000 cars, city 2 must receive 4000 cars, and city 3 must receive 3000 cars. The cost of shipping a car from each plant to each city is given in the table below. At most, 2200 cars may be sent from a given plant to a given city. Formulate an MCNFP that can be used to minimize the cost of meeting demand. Also, draw the appropriate network and denote the cij, bi, and uij & lij if necessary.

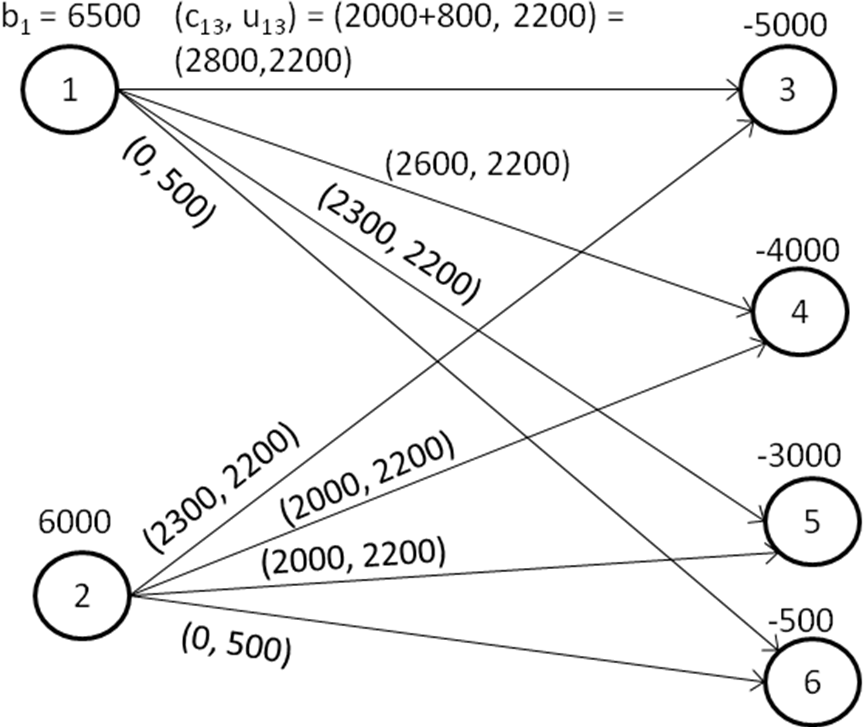
|  |  |  |  |
| --- | --- | --- | --- |
|  | To ($) | | |
| From | City 1 | City 2 | City 3 |
| Detroit | 800 | 600 | 300 |
| Dallas | 500 | 200 | 200 |

Solution:  
Node 1 represents a start node.  
Nodes 2 and 3 are the plants (Detroit is 2, Dallas is 3).  
Nodes 4, 5, and 6 are Cities 1, 2, and 3, respectively.  
The labels on the arcs are (cij, uij). lij is 0 for all arcs.  
The labels on the nodes are bi.  
b1 is equal to total demand, which is 12000.



***Note: Problem is infeasible since 2200 upper limit.*OR**

Nodes 1 and 2 are the plants (Detroit is 1, Dallas is 2).  
Nodes 3, 4, and 5 are Cities 1, 2, and 3, respectively.  
Node 6 is the dummy demand node (supply is 500 more than demand).  
The labels on the arcs are (cij, uij). lij is 0 for all arcs.  
The labels on the nodes are bi.



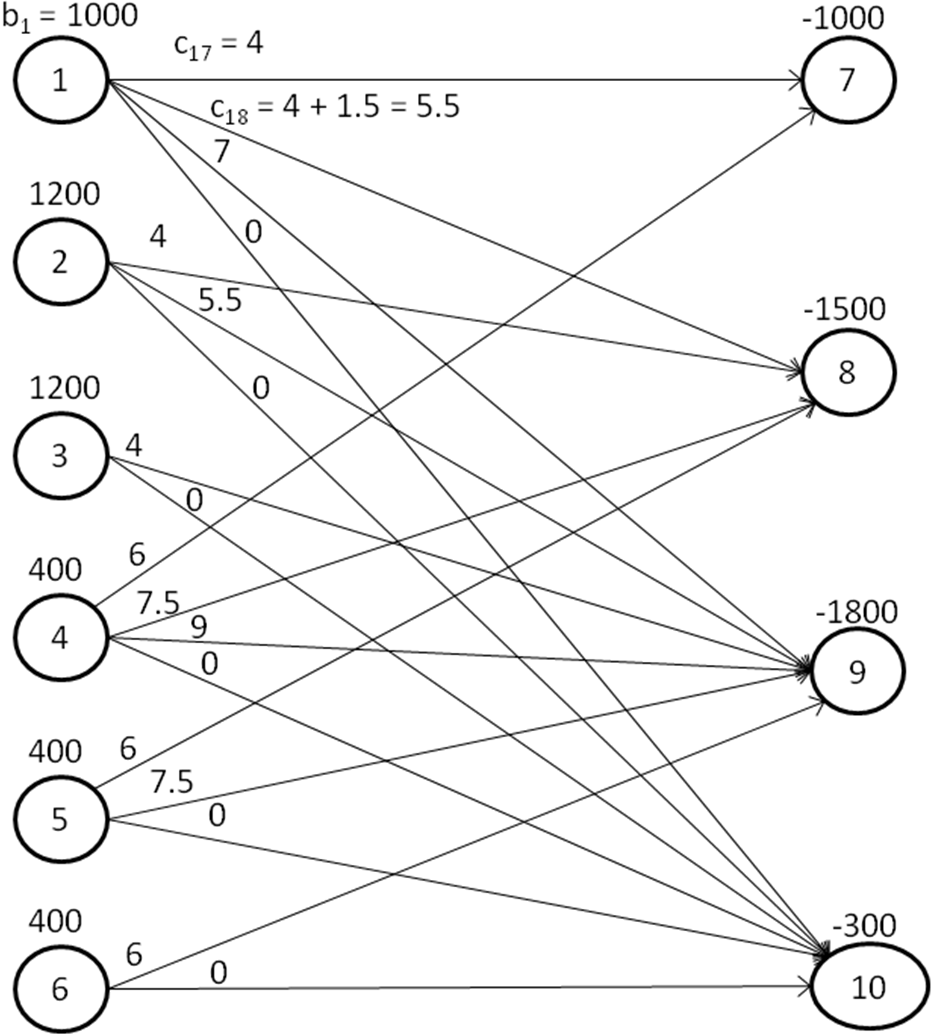
***Note: Problem is infeasible since 2200 upper limit.***

**Example 2:**  
During the next three months, Shoemakers, Inc. must meet (on time) the following demands for shoes: month 1, 1000 pairs; month 2, 1500 pairs; month 3, 1800 pairs. It takes 1 hour of labor to produce a pair of shoes. During each of the next three months, the following number of regular-time labor hours are available; month 1, 1000 hours, month 2, 1200 hours; month 3, 1200 hours. Each month, the company can require workers to put in up to 400 hours of overtime. Workers are paid only for the hours they work, and a worker receives $4/hr for regular-time work and $6/hr for overtime work. At the end of each month, a holding cost of $1.50 per pair of shoes is incurred. Formulate an MCNFP that can be used to minimize the total cost incurred in meeting the demands of the next three months. A formulation requires drawing the appropriate network and determining the cij's, bi's, and arc capacities. How would you modify your answer if demand could be backlogged (all demand must still be met by the end of month 3) at a cost of $20/pair/month?

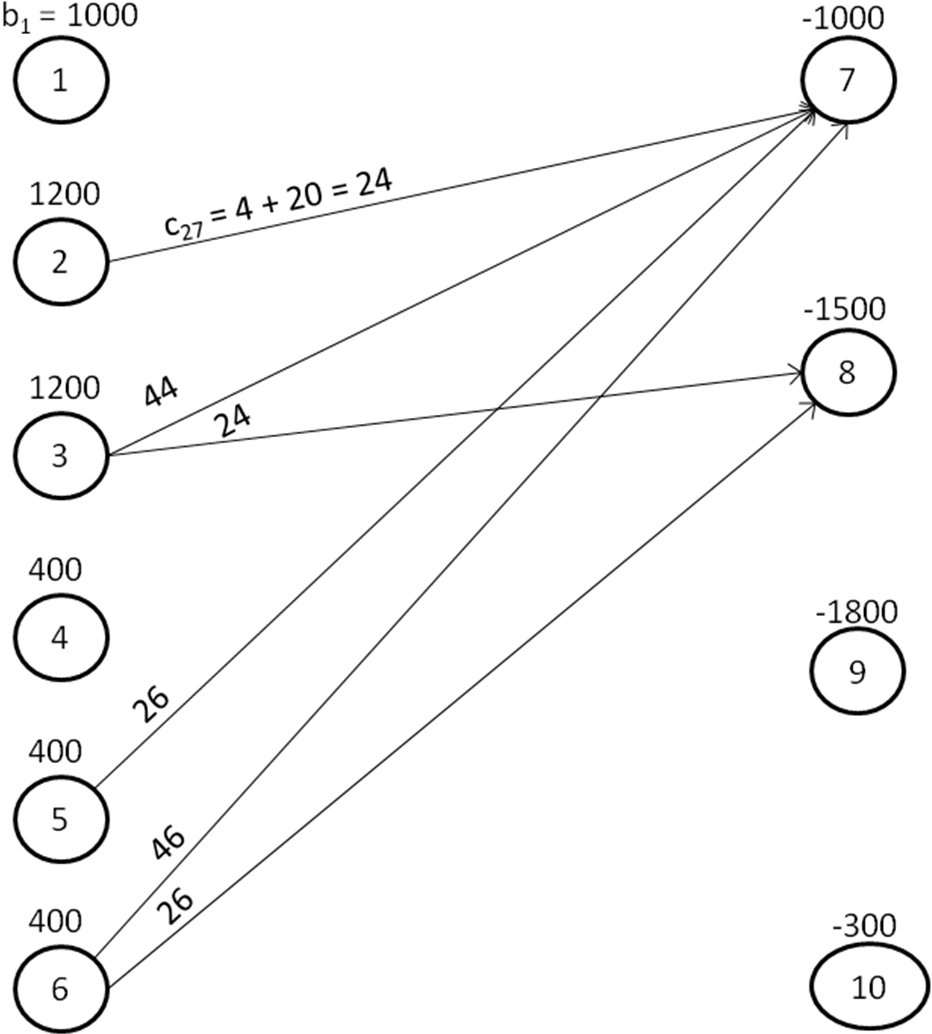
Nodes 1, 2, and 3 are the regular labor hours for months 1, 2, and 3, respectively.  
Nodes 4, 5, and 6 are the overtime labor hours for months 1, 2, and 3, respectively.  
Nodes 7, 8, and 9 are the demand nodes for months 1, 2, and 3, respectively.  
Node 10 is the dummy demand node (total supply is 4600, total demand is 4300, dummy demand is 300).  
The labels on the nodes are bi.  
The labels on the arcs are cij. lij is 0 for all arcs, uij is ∞ for all arcs (the true upper bound is imposed by bi)

To ensure we have no backlogs; we do not include the arcs from supply months to previous demand months. If those arcs were included, then the upper bound for those arcs would be set to 0.

If we include backlogs, then we must include the arcs (and include the associated cost), and an upper bound would not be needed.



Note: the following figure only shows the ADDITIONAL arcs to include backlogging. The total network would include the arcs from the previous figure as well.



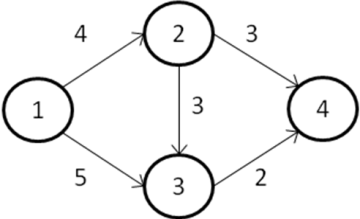
**Network Models:**

*Maximum Flow*

Find the maximum flow from node 1 to node m. (Where m is the terminal node.)

Let node 1 be the source and node m be the sink.  
  
Let (i,j) be an arc from node i to node j.

Let the values on the arcs denote an upper bound on the flow across that arc.



Obvious solution: 5

Formulation (General - not specific to the above problem):

Index Set:  
i: nodes (i = 1, 2, …, m)  
j: nodes (j = 1, 2, …, m)  
(i,j) represents arc from i to j

Data:  
uij: upper-limit on flow on arc (i.j)

Decision Variables:  
xij: flow from node i to j

Objective: Maximize Flow (from m to 1, by creating an arc that links sink to source)  
Maximize xm1S.T.  
 (Flow Conservation)  
xij ≤ uij   
xij ≥ 0 

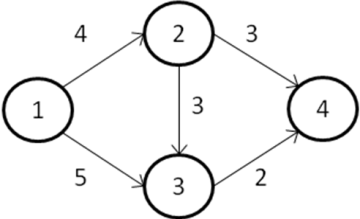
Dual:  
Minimize   
S.T.  


All dual variables will end up being 0 or 1. wi = 0 if i is in Set 1. wi = 1 if i is in Set 2.

So the objective is adding up the uij for the arcs that go from Set 1 to Set 2; which is the cutset capacity (and also the maximum flow).

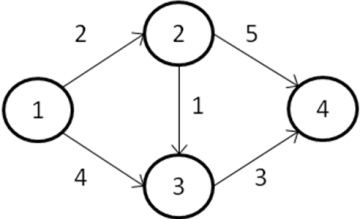
Cutset is where you split the nodes into two mutually exclusive sets; where one set contains node 1 (i.e., Set 1) and the other set contains node m (i.e., Set 2). The cutset capacity adds the arcs going from Set 1 to Set 2.

Example:



Cutsets:  
Set 1: Set 2: Cutset Capacity:  
{1} {2,3,4} 4 + 5 = 9  
{1,2} {3,4} 3 + 3 + 5 = 11  
{1,3} {2,4} 4 + 2 = 6  
{1,2,3} {4} 2 + 3 = 5

The smallest cutset is: 5

Example 2:  


Cutsets:  
Set 1: Set 2: Cutset Capacity:  
{1} {2,3,4} 2 + 4 = 6  
{1,2} {3,4} 4 + 1 + 5 = 10  
{1,3} {2,4} 2 + 3 = 5  
{1,2,3} {4} 5 + 3 = 8

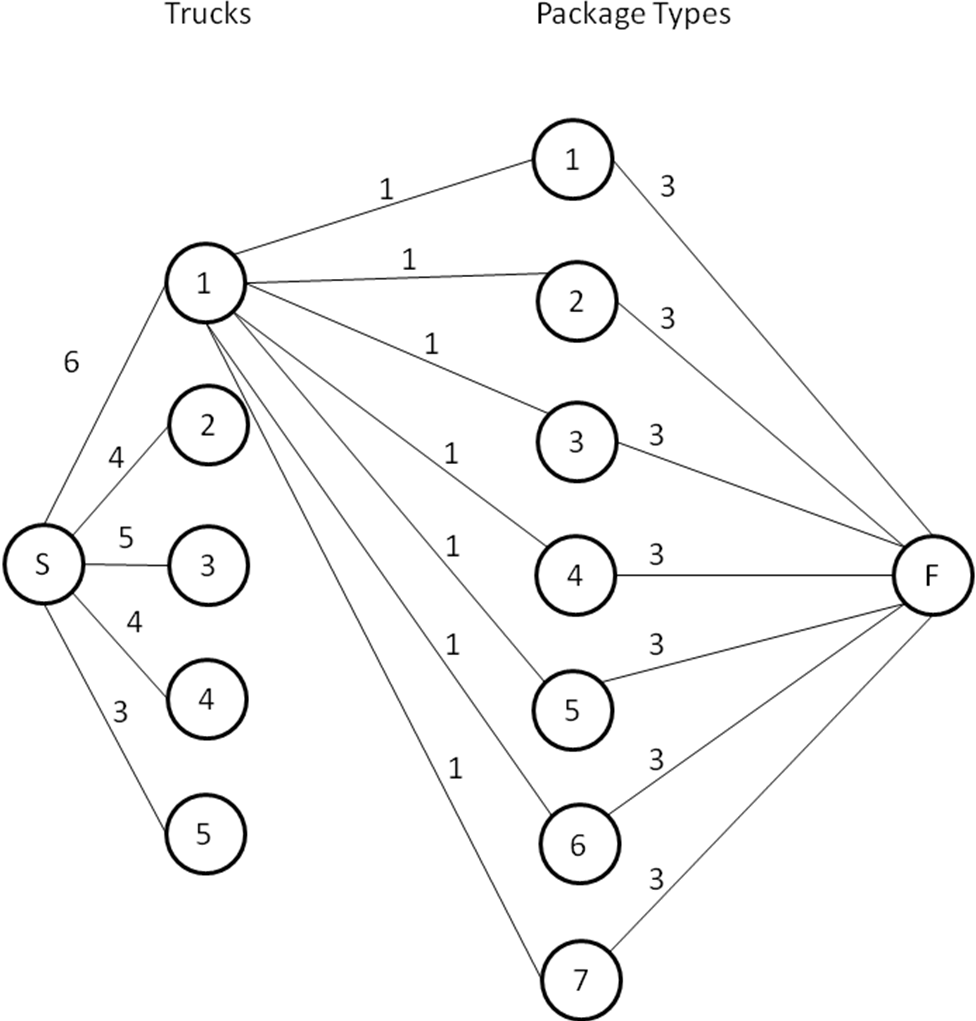
The smallest cutset is: 5

Can solve algorithmically using a label correcting method; see book for details.

Complicating issues:  
Arcs can have lower bounds on flow.  
Undirected arcs (arc can go either 1 to 2 or 2 to 1 - but not both simultaneously).  
Arcs in both directions (an arc from 1 to 2, and an arc from 2 to 1) .

Can solve using sophisticated labeling strategies or algorithms; or can use linear programming/formulation so long as unimodularity (it is even with above issues so long as bounds are integer) is intact. Otherwise, you can use integer programming solution techniques.

Example Problem:  
Seven types of packages are to be delivered by five trucks. There are three packages of each type, and the capacities of the five trucks are 6, 4, 5, 4, and 3 packages, respectively. Set up a maximum-flow problem that can be used to determine whether the packages can be loaded so that no truck carries two packages of the same type.  
  
Possible: Yes!  
  
One Solution:  
Truck 1: 1, 2, 3, 4, 5, 6  
Truck 2: 7, 1, 2, 3  
Truck 3: 1, 2, 3, 4, 5  
Truck 4: 4, 5, 6, 7  
Truck 5: 6, 7



Note: Trucks to Type needs arcs for trucks 2 thru 5 for all product types with upper bounds of 1.

Then solve for a maximum flow problem (or minimum cutset). If solution is 21, then it is possible (and the solution will give you a feasible/optimal solution).

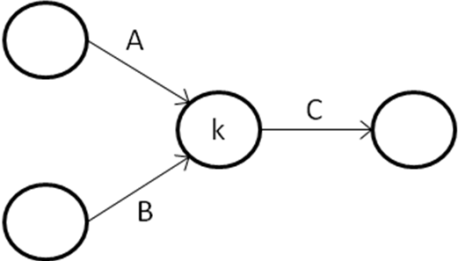
**Network Models:***CPM and PERT  
Note: Some notation used in this set of notes is not exactly the same as the notation in the textbook.*

CPM = critical path method  
PERT = Program Evaluation and Review Technique

Application: Primarily used in scheduling projects.  
Concept: Managing complex projects with inter-related activities.

*Project Network*  
activities - directed arcs  
events - nodes (event = completion or start of an activity)

The graph (or network) is designed to capture precedence relationships.



Activity C cannot begin until both Activities A and B are completed.  
Node k cannot *fire* until both events A and B are completed.

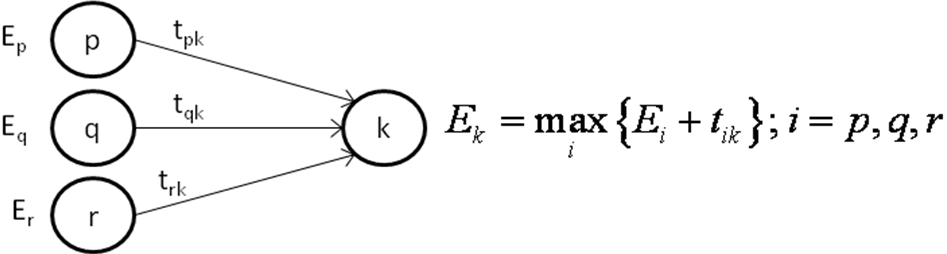
Goal: To minimize the amount of time it takes to finish the entire project.

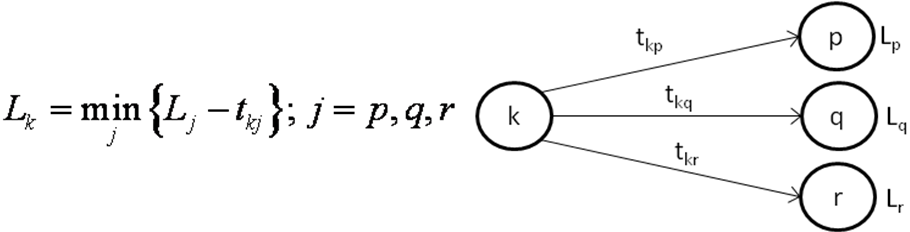
Consider activity (i,j).  
Suppose (i,j) requires tij time units to complete. How much flexibility do we have with activity (i,j)?

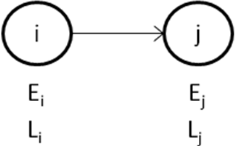
Activities with no flexibility are called *critical*.

Consider event k. Let Ek be the earliest time for event k. Let Lk be the latest time for event k.

What is the earliest time for event k?

This depends on predecessor activities.  


What is the latest time Lk for event k?  
This depends on successor activities.  


Consider Activity (i,j):  


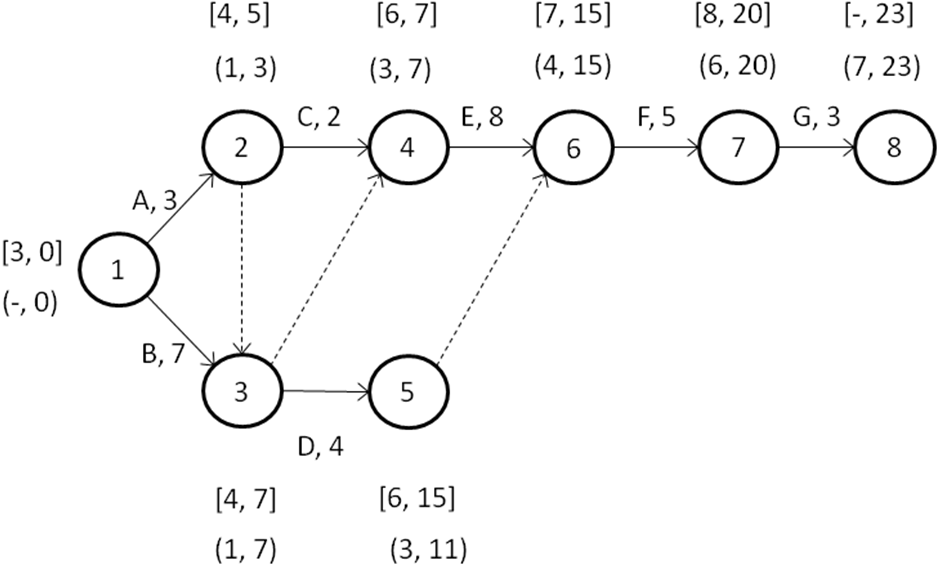
What is the maximum time that can be allocated to (i,j)? Lj - Ei

Activity (i,j) requires tij.  
Slack = Lj - Ei - tij  
Critical activities have slack = 0.  
Note: Generally, the last event will be critical since you will set Lj = Ej.  
Critical Path = Longest Path Problem  
Note: If you need to finish a project 'early', then it is assumed you can "crash" the project with additional resources.

**Example 1:**Determine the critical path.

|  |  |  |
| --- | --- | --- |
| Activity | Predecessor(s) | Activity Time (tij) |
| A | - | 3 |
| B | - | 7 |
| C | A | 2 |
| D | A, B | 4 |
| E | B, C | 8 |
| F | D, E | 5 |
| G | F | 3 |

Labels: (-, 0) = (Predecessor, Ek) [-, 23] = [Successor, Lk]



Critical Path = Longest Path Problem; Where slack = 0. Slack = Lj - Ei - tij.  
Could have drawn D from Node 3 to Node 6.  
Activity Lj - Ei - tij  
B; (1, 3) 7 - 0 - 7 = 0 Critical!  
A; (1, 2) 5 - 0 - 3 = 2 Not critical!

**PERT: Program Evaluation and Review Technique**  
CPM assumes tij is known with certainty (deterministic). Whereas, that is typically not the case. PERT models each activity time tij as a random variable using a beta distribution.

Define:  
a = estimate of the activity's duration under the most favorable conditions  
b = estimate of the activity's duration under the least favorable conditions  
m = most likely value for the activity's duration

; expected value of tij

; variance of tij

You can then use these values to determine the Expected Value and Variance of the Critical Path. Remember, variances are additive.

PERT then assumes that the Critical Path is normally distributed. Allowing you to infer about the probability a project will be completed within 10 days. However, this assumes that the critical path stays the same in favorable and unfavorable conditions.

Major drawbacks of PERT:  
1. Activities are assumed to be independent.  
2. Activities may not follow a beta distribution.  
3. Critical Path stays the same - regardless of conditions.

**Example 1 Continued:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Activity | Predecessor(s) | m | a | b | E(tij) | Var(tij) |
| A | - | 3 | 2 | 5 | 3.17 | 0.25 |
| B | - | 7 | 4 | 11 | 7.17 | 1.36 |
| C | A | 2 | 1 | 3 | 2.00 | 0.11 |
| D | A, B | 4 | 3 | 5 | 4.00 | 0.11 |
| E | B, C | 8 | 5 | 10 | 7.83 | 0.69 |
| F | D, E | 5 | 2 | 7 | 4.83 | 0.69 |
| G | F | 3 | 2 | 5 | 3.17 | 0.25 |

What is the probability that the project can be completed within 25 days?  
**Critical Path: 1 - 3 - 4 - 6 - 7 - 8  
Events: B, E, F, G**  
  
E(CP) = 7.17 + 7.83 + 4.83 + 3.17 = 23.00  
Var(CP) = 1.36 + 0.69 + 0.69 + 0.25 = 3.00; Standard Deviation = 1.732  
  
Assume CP is normally distributed.  
  
