Network Models

Operations Research: Applications and Algorithms, 4th edition, Wayne L. Winston, Chapters 7 and 8

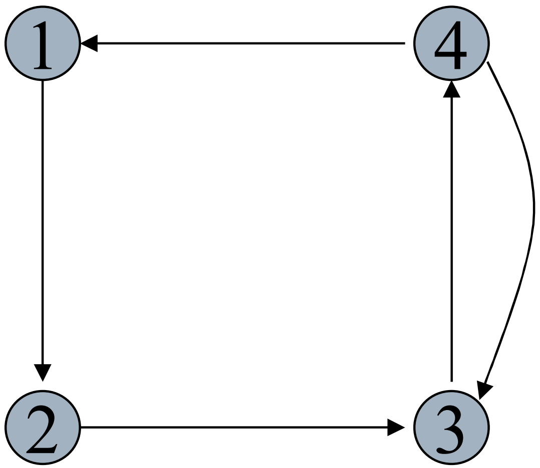
- Minimum Cost Network Flow Problems
- Shortest Path Problems
- Maximum Flow Problems
- Transportation
- Assignment
- Transshipment

Each of these can be solved by the simplex algorithm, but specialized algorithms for each type of problem are much more efficient.

1 Basic Definitions

- A **graph** or **network** is defined by two sets of symbols: nodes and arcs.
- A set (call it *V*) of points, or vertices. The vertices of a graph or network are also called **nodes**.
- An **arc** consists of an ordered pair of vertices and represents a possible direction of motion that may occur between vertices.
- A sequence of arcs such that every arc has exactly one vertex in common with the previous arc is called a **chain**.

- A **path** is a chain in which the terminal node of each arc is identical to the initial node of next arc.
- For example in the figure below:
  - (1,2)-(2,3)-(4,3) is a chain but not a path;
  - (1,2)-(2,3)-(3,4) is a chain and a path, which represents a way to travel from node 1 to node 4.



2 Minimum Cost Network Flow Problems

- The transportation, assignment, transshipment, shortest path, maximum flow, and critical path problems (project scheduling) are all special cases of minimum cost network flow problems (MCNFP).
- Any MCNFP can be solved by a generalization of the transportation simplex called the **network simplex**.
- LINGO/LINDO can easily be used to solve an MCNFP problem.

MCNFP model and structural properties:

$$\min \sum_{all\ arcs} cijXij$$

$$s.t.$$

$$\sum_j Xij - \sum_k Xki = bi \qquad \text{(for each node } i \text{ in the network)}$$

$$Lij \le Xij \le Uij \qquad \text{(for each arc in the network)}$$

- Decision variable X_{ij} is associated with each arc (*i,j*)
- Constraints stipulate that the net flow out of node *i* must equal b_i and are referred to as the **flow balance equation**.
- The flow balance equations have the important property: Each variable X_{ij} has a coefficient of +1 in the node i flow balance equation, a coefficient of -1 in the node j flow balance equation, and a coefficient of 0 in all other flow balance equations.
- Capacity constraints

LINGO code

```
MODEL:
SETS:
NODES/1..6/:SUPP;
ARCS(NODES,NODES)/1,2   1,3   2,4   2,5   3,4   3,5   4,5   4,6   5,6/
:CAP,FLOW,COST;
ENDSETS
MIN=@SUM(ARCS:COST*FLOW);
@FOR(ARCS(I,J):FLOW(I,J)<CAP(I,J));
@FOR(NODES(I):-@SUM(ARCS)(J,I):FLOW(J,I))
+@SUM(ARCS(I,J):FLOW(I,J))=SUPP(I));
DATA:
COST=10,50,30,70,10,60,30,60,30;
SUPP=900,0,0,0,0,-900;
CAP=800,600,600,100,300,400,600,400,600;
ENDDATA
END
```

- The solution is z=95,000 minutes, $x12$=700, $x13$=200, $x24$=600, $x25$ =100, $x34$ =200, $x45$= 400, $x46$ =400, $x56$ =500.
- This LINGO program can be used to solve any MCNFP. Just input the set of nodes, supplies, arcs, and unit shipping cost; hit **GO** and you are done!

3. Transshipment Problems

- **A transportation problem** allows only shipments that go directly from supply points to demand points.
- Shipping problems with any or all of these characteristics are transshipment problems.
  - In many situations, **shipments** are allowed **between supply points** or between **demand points**.
  - Sometimes there may also be points (called **transshipment points**) through which goods can be transshipped on their journey from a supply point to a demand point.
- **Supply points**: can send goods but cannot receive goods;
  Demand points: can receive goods but cannot send goods;
  Transshipment points: can both receive and send goods

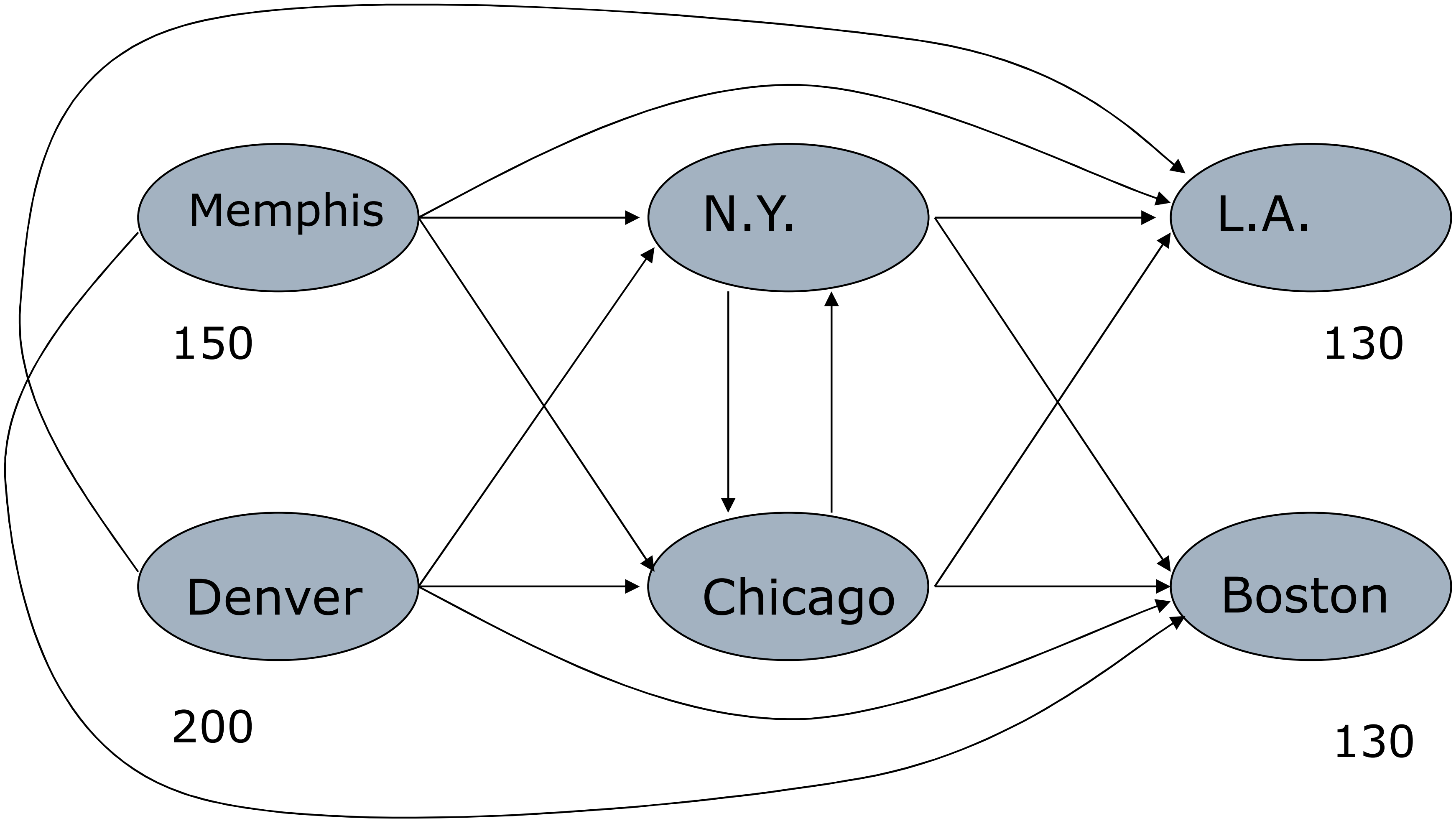
Widgetco Example

Widgetco manufactures widgets at 2 factories, one in Memphis and one in Denver. The Memphis factory can produce as many as 150 widgets per day, and the Denver factory can produce as many as 200 widgets per day. Widgets are shipped by air to customers in Los Angeles and Boston. The customers in each city require 130 widgets per day. Because of the deregulation of airfares, Widgetco believes that it may be cheaper to first fly some widgets to New York or Chicago and then fly them to their final destinations. The costs of flying a widget are shown in the following table. Widgetco wants to minimize the total cost of shipping the required widgets to its customers.

	To ($)					
From	Memphis	Denver	N.Y.	Chicago	L.A.	Boston
Memphis	0	–	8	13	25	28
Denver	–	0	15	12	26	25
N.Y.	–	–	0	6	16	17
Chicago	–	–	6	0	14	16
L.A.	–	–	–	–	0	–
Boston	–	–	–	–	–	0

"–" indicates a shipment is impossible

Widgetco Example



Supply

Total = 350

Demand

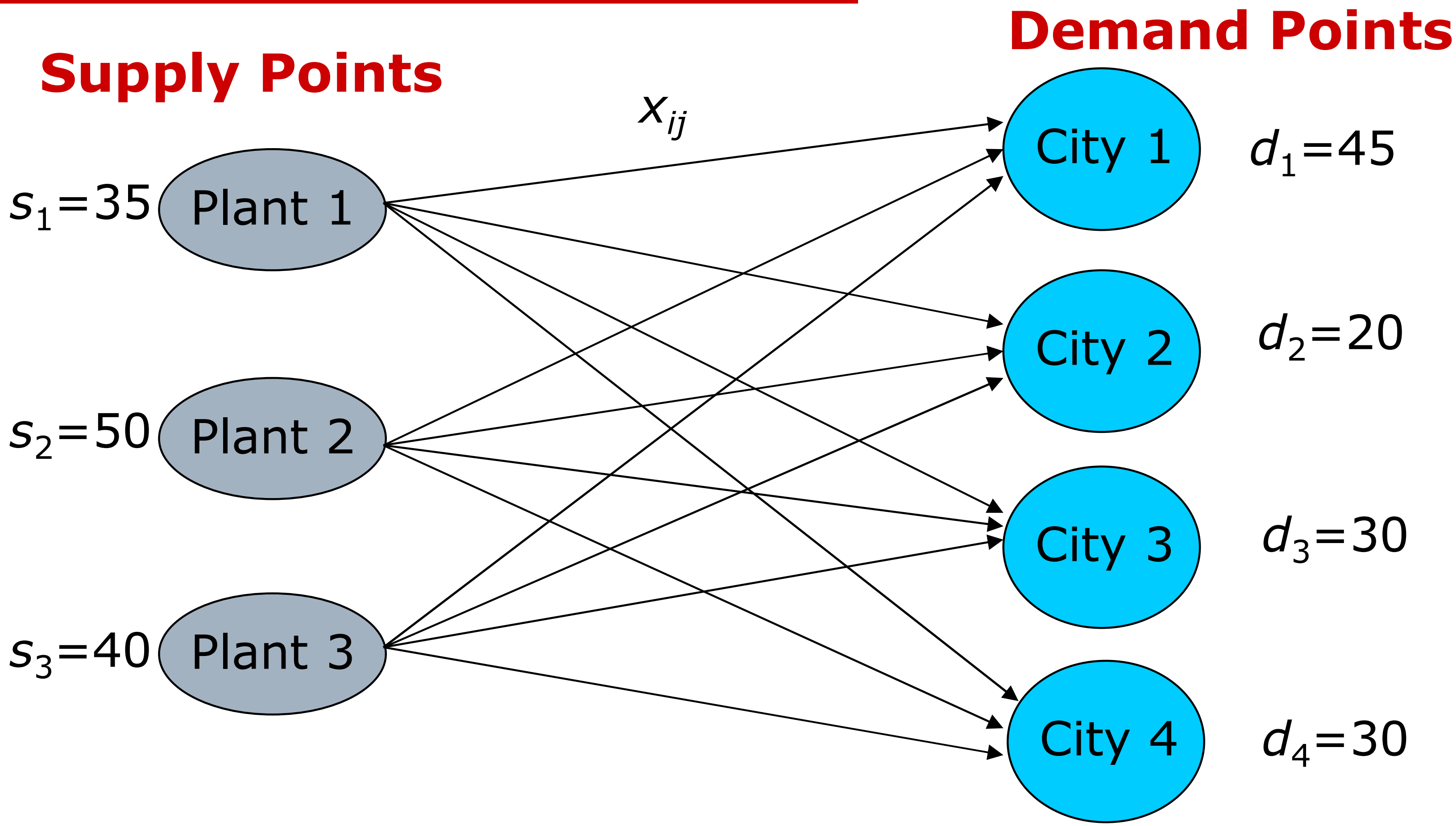
Total = 260

4 Transportation Problems

Example. **Powerco** has 3 electric power plants that supply the needs of 4 cities. Each power plant can supply the following numbers of million kwh of electricity: plant 1: 35; plant 2: 50; plant 3: 40. The peak power demands in these cities which occur at the same time (2pm), are as follows (in million kwh): city 1: 45; city 2: 20; city 3: 30; city 4: 30. The costs of sending 1 million kwh of electricity from plant to city depend on the distance the electricity must travel. Formulate an LP to minimize the cost of meeting each city's peak power demand.

	To				Supply
From	City 1	City 2	City 3	City 4	(million kwh)
Plant 1	\$8	\$6	\$10	\$9	35
Plant 2	\$9	\$12	\$13	\$7	50
Plant 3	\$14	\$9	\$16	\$5	40
Peak demand (million kwh)	45	20	30	30	

Graphical Representation



Decision Variables: x_{ij} = # of million kwh produced at plant i and sent to city j, i=1,2,3; j=1,2,3,4.

Constraints: Supply (capacity constraints)
Demand (demand constraints)

LP Formulation

$$\text{Min } z = 8x_{11} + 6x_{12} + 10x_{13} + 9x_{14} + 9x_{21} + 12x_{22} + 13x_{23} + 7x_{23} + 14x_{31} + 9x_{32} + 16x_{33} + 5x_{34}$$

Minimize total shipping costs, coefficient for x_{ij} is the cost (distance) of sending 1 million kwh of electricity from plant *i* to city *j*

$$\text{s.t.} \quad x_{11} + x_{12} + x_{13} + x_{14} \le 35$$

$$x_{21} + x_{22} + x_{23} + x_{24} \le 50$$

$$x_{31} + x_{32} + x_{33} + x_{34} \le 40$$

Supply Constraints

$$x_{11} + x_{21} + x_{31} \ge 45$$

$$x_{12} + x_{22} + x_{32} \ge 20$$

$$x_{13} + x_{23} + x_{33} \ge 30$$

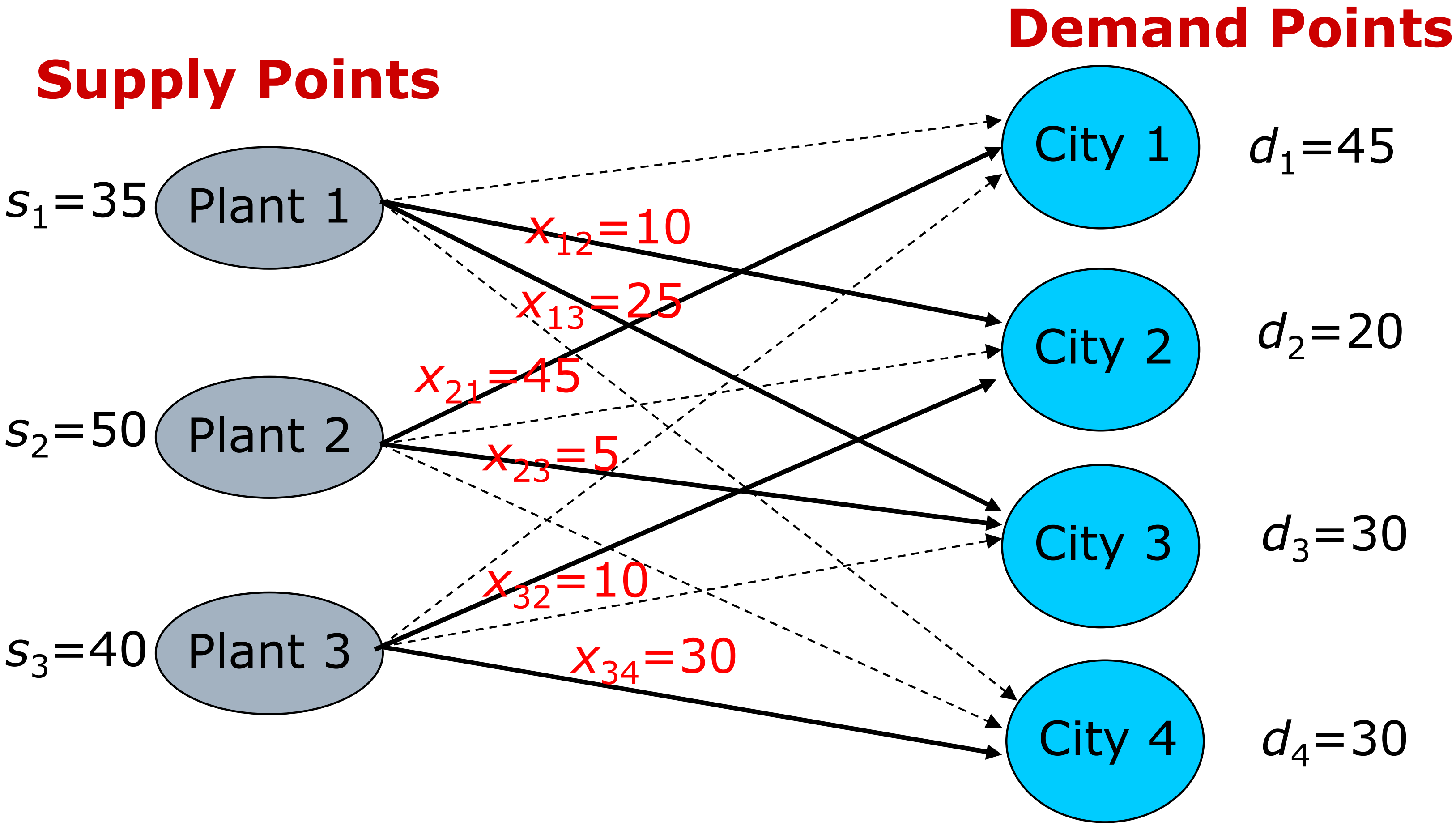
$$x_{14} + x_{24} + x_{34} \ge 30$$

$$x_{ij} \ge 0 \; (i = 1,2,3; \; j = 1,2,3,4)$$

Demand Constraints

Optimal Solutions

$$z = 1020, x_{12} = 10, x_{13} = 25, x_{21} = 45, x_{23} = 5, x_{32} = 10, x_{34} = 30,$$
$$x_{11} = x_{14} = x_{22} = x_{24} = x_{31} = x_{33} = 0$$



General Description of a Transportation Problem

In general, a transportation problem is specified by the following information:

1. A set of m **supply points** from which a good is shipped. Supply point i can supply as most s_i units.
2. A set of n **demand points** to which the good is shipped. Demand point j must receive at least d_j units of the shipped good.
3. Each unit produced at supply point i and shipped to demand point j incurs a variable cost of c_{ij}.

Model of a Transportation Problem

Decision Variables:

x_{ij} = # of units shipped from supply point i to demand point j,
i=1,…,m; j=1,…,n.

$$\min \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

If it is a maximization objective, it is still a transportation problem.

s.t.

$$\sum_{j=1}^{n} x_{ij} \le s_i \quad (i = 1,2,\dots,m)$$

Supply constraints

$$\sum_{i=1}^{m} x_{ij} \ge d_j \quad (j = 1,2,\dots,n)$$

Demand constraints

$$x_{ij} \ge 0 \quad (i = 1,2,\dots,m;\ j = 1,2,\dots,n)$$

A Balanced Transportation Problem

In a "***balanced transportation***" problem, the total supply equals the total demand:

$$\sum_{i=1}^{m} s_i = \sum_{j=1}^{n} d_j$$

$$\begin{aligned}
\text{min} \quad & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \\
\text{s.t.} \quad & \sum_{j=1}^{n} x_{ij} = s_i \quad (i = 1,2,\dots,m) \\
& \sum_{i=1}^{m} x_{ij} = d_j \quad (j = 1,2,\dots,n) \\
& x_{ij} \geq 0 \quad (i = 1,2,\dots,m; \\
& \qquad\qquad j = 1,2,\dots,n)
\end{aligned}$$

- Powerco Problem is a balanced transportation problem.
- All constraints must be binding.
- It is relatively simple to find a basic feasible solution.
- Simplex pivots for these problems do not involve multiplication and reduce to additions and subtractions.
- Thus, it is desirable to formulate a transportation problem as a balanced transportation problem.

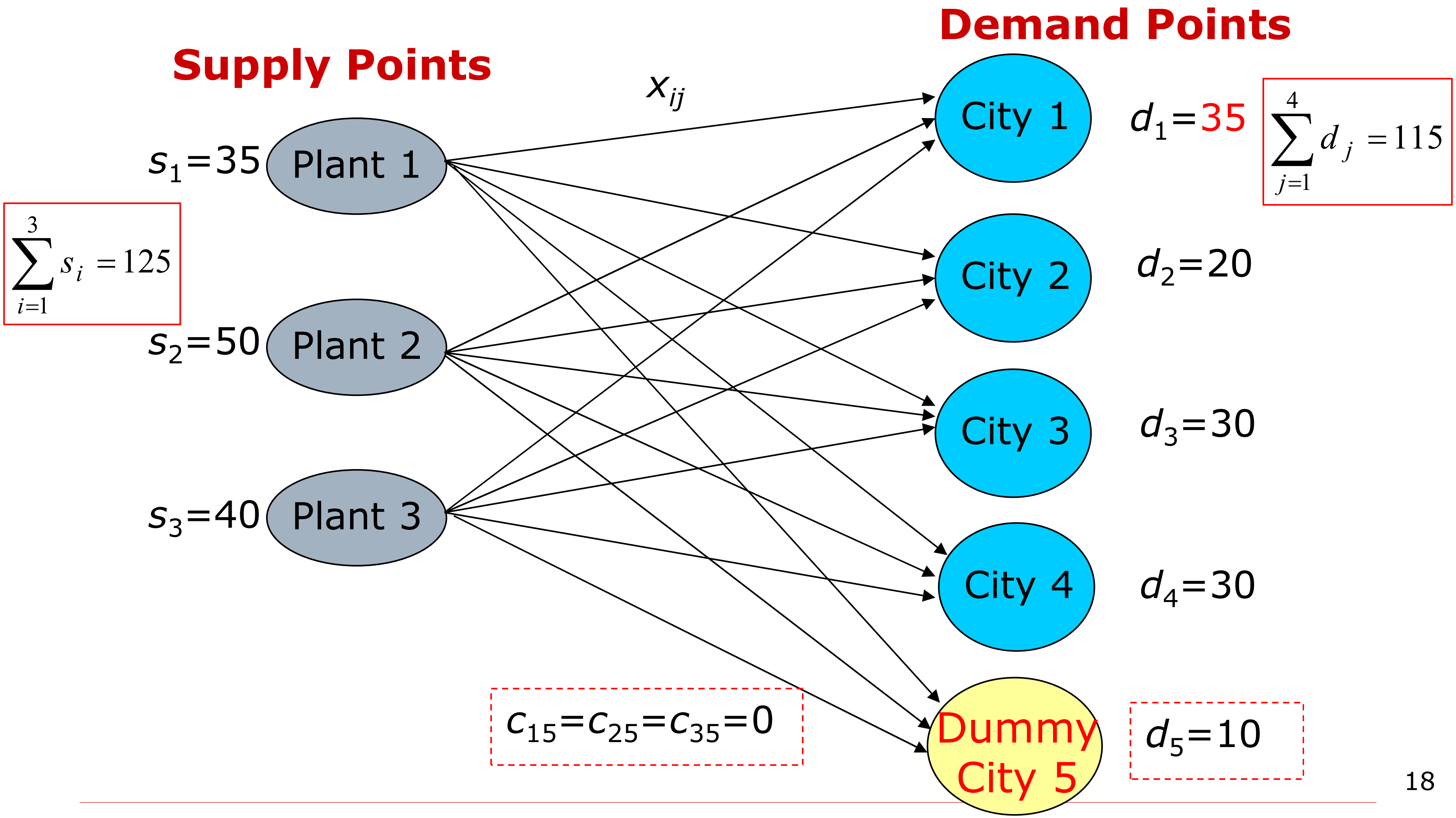
Balancing a Transportation Problem

If **total supply exceeds total demand**, we can balance a transportation problem by creating a ***dummy demand point.***

At dummy demand point:

- It has a demand equal to the amount of excess supply.
- Shipments to it are assigned a cost of zero ($c_{ij'}=0$ for all i where j' represents dummy demand point) because they are not real shipments.
- Shipments to it indicate unused supply capacity.

Balancing a Transportation Problem



Balancing a Transportation Problem

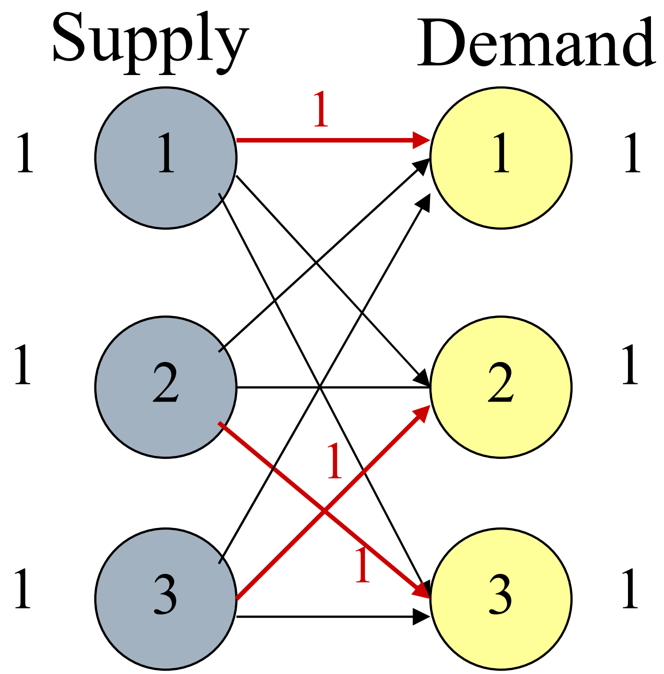
If **total supply is less than total demand**, then the problem has no feasible solution: *demand cannot be satisfied*.

However, it is sometimes desirable to allow the possibility of leaving some demand unmet:

- A penalty (cost) is often associated with the unmet demand.
- To balance the problem, add a "dummy (or shortage) supply point" having a supply of unmet demand and the cost of shipping from the dummy supply point to any demand point is the unit penalty cost.

5 Assignment Problems

- Assignment problems is a special case of transportation problems: **Supplies and Demands are All equal to 1.**



- General formulation:

$$\min \sum_{i=1}^{m}\sum_{j=1}^{m} c_{ij}x_{ij}$$

$$\text{s.t.} \quad \sum_{j=1}^{m} x_{ij} = 1 \quad (i = 1,2,\ldots,m)$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad (j = 1,2,\ldots,m)$$

$$x_{ij} \geq 0 \quad (i = 1,2,\ldots,m;\ j = 1,2,\ldots,m)$$

A solution to this balanced transportation problem can be viewed as an "assignment".

Machineco Problem

Machineco has 4 machines and 4 jobs to be completed. Each machine must be assigned to complete one job. The time required to set up each machine for completing each job is shown in the following table. Machineco wants to minimize the total setup time needed to complete the 4 jobs. Formulate this problem as an LP.

Machine	Time (Hours)			
	Job 1	Job 2	Job 3	Job 4
1	14	5	8	7
2	2	12	6	5
3	7	8	3	9
4	2	4	6	10

Machineco Problem

Machineco must determine which machine should be assigned to each job.

Let the decision variables be defined as:

x_{ij} = 1 if machine i is assigned to meet the demands of job j;

x_{ij} = 0 if machine i is not assigned to meet demands of job j.

$$
\begin{aligned}
\min\ z =\ & 14x_{11} + 5x_{12} + 8x_{13} + 7x_{14} \\
& + 2x_{21} + 12x_{22} + 6x_{23} + 5x_{24} \\
& + 7x_{31} + 8x_{32} + 3x_{33} + 9x_{34} \\
& + 2x_{41} + 4x_{42} + 6x_{43} + 10x_{44} \\
s.t.\quad & x_{11} + x_{12} + x_{13} + x_{14} = 1 \\
& x_{21} + x_{22} + x_{23} + x_{24} = 1 \\
& x_{31} + x_{32} + x_{33} + x_{34} = 1 \\
& x_{41} + x_{42} + x_{43} + x_{44} = 1, \\
& x_{11} + x_{21} + x_{31} + x_{41} = 1 \\
& x_{12} + x_{22} + x_{32} + x_{42} = 1 \\
& x_{13} + x_{23} + x_{33} + x_{43} = 1 \\
& x_{14} + x_{24} + x_{34} + x_{44} = 1, \\
& x_{ij} = 0 \text{ or } x_{ij} = 1,\ (i = 1, 2, 3, 4;\ j = 1, 2, 3, 4)
\end{aligned}
$$

Relax Integrality Requirements

Recall the transportation simplex, each pivot involves only additions and subtractions. Using this fact, we can show that *if all the supplies and demands for a **transportation problem** are integers, then the transportation problem will have an optimal solution in which all the variables are integers*.

For the **assignment problem**, we can remove the integrality requirement due to two reasons:

- All the supplies and demands for the assignment problem are integers. This implies the optimal solution must be integer.
- Since rhs of each constraint equals 1, each x_{ij} must be nonnegative integer that is no larger than 1, so each x_{ij} must be equal to 0 or 1.

Solution Approach

- Although the transportation simplex appears to be very efficient, the transportation simplex is often very inefficient for **assignment problems**.
- A high degree of **degeneracy** in an assignment problem may cause the transportation simplex to be inefficient way of solving assignment problems.

  In any bfs to an $m \times m$ assignment problem, there will always be m basic variables that equal to 1 and m-1 basic variables that equal 0.
- For this reason, and the fact that the algorithm is even much simpler than the transportation simplex, the **Hungarian method** is usually used to solve assignment (min) problems.

LINGO code

Observe that this LINGO program can (with simple editing) be used to solve any assignment problem (even if it is not balanced!). For example, if you had 10 machines available to perform 8 jobs, you would edit line 2 to indicate that there are 10 machines (replace 1..4 with 1..10). Then edit line 3 to indicate that there are 8 jobs. Finally, in line 12,you would type the 80 entries of your cost matrix, following “COST” and you would be ready to roll!

```
MODEL:
  1]SETS:
  2]MACHINES/1..4/;
  3]JOBS/1..4/;
  4]LINKS(MACHINES,JOBS):COST,ASSIGN;
  5]ENDSETS
  6]MIN=@SUM(LINKS:COST*ASSIGN);
  7]@FOR(MACHINES(I):
  8]@SUM(JOBS(J):ASSIGN(I,J))<1);
  9]@FOR(JOBS(J):
 10]@SUM(MACHINES(I):ASSIGN(I,J))>1);
 11]DATA:
 12]COST = 14,5,8,7,
 13]2,12,6,5,
 14]7,8,3,9,
 15]2,4,6,10;
 16]ENDDATA
END
```

REMARK From our discussion of the Machineco example, it is unnecessary to force the ASSIGN(I,J) to equal 0 or 1; this will happen automatically!

6 Shortest Path Problems

- Assume that each arc in the network has a length associated with it.
- The problem of finding the shortest path from node 1 to any other node in the network is called a **shortest path problem**.
- Many important optimization problems can best be analyzed by means of a graphical or network representation, such as the example below.

Example 2: Equipment Replacement

- Assume that a new car (or machine) has been purchased for \$12,000 at time 0. To simplify the computations, we assume that at any time it costs \$12,000 to purchase a new car.
- The cost of maintaining the car during a year depends on the age of the car at the beginning of the year.
- In order to avoid the high maintenance cost associated with an older car, we may trade in the car and purchase a new car.
- The goal is to minimize the net cost incurred during the next five years.

Age of Car (Years	**0**	**1**	**2**	**3**	**4**	**5**
Annual Maintenance cost (\$)	2K	4K	5K	9K	12K	--
Trade-in Price (\$)	--	7K	6K	2K	1K	0

Ex. 2: Solution

- This problem should be formulated as a shortest path problem.
- The network will have six nodes.
  - Node i is the beginning of year i
  - arc (i,j) for $i<j$ corresponds to purchasing a new car at the beginning of year i and keeping it until the beginning of year j.
- The length of arc (i,j) (c_{ij}) is the total net cost incurred from year i to j.

c_{ij} = maintenance cost incurred during years $i, i+1, \ldots, j-1$
+ cost of purchasing a car at the beginning of year i
- trade-in value received at the beginning of year j

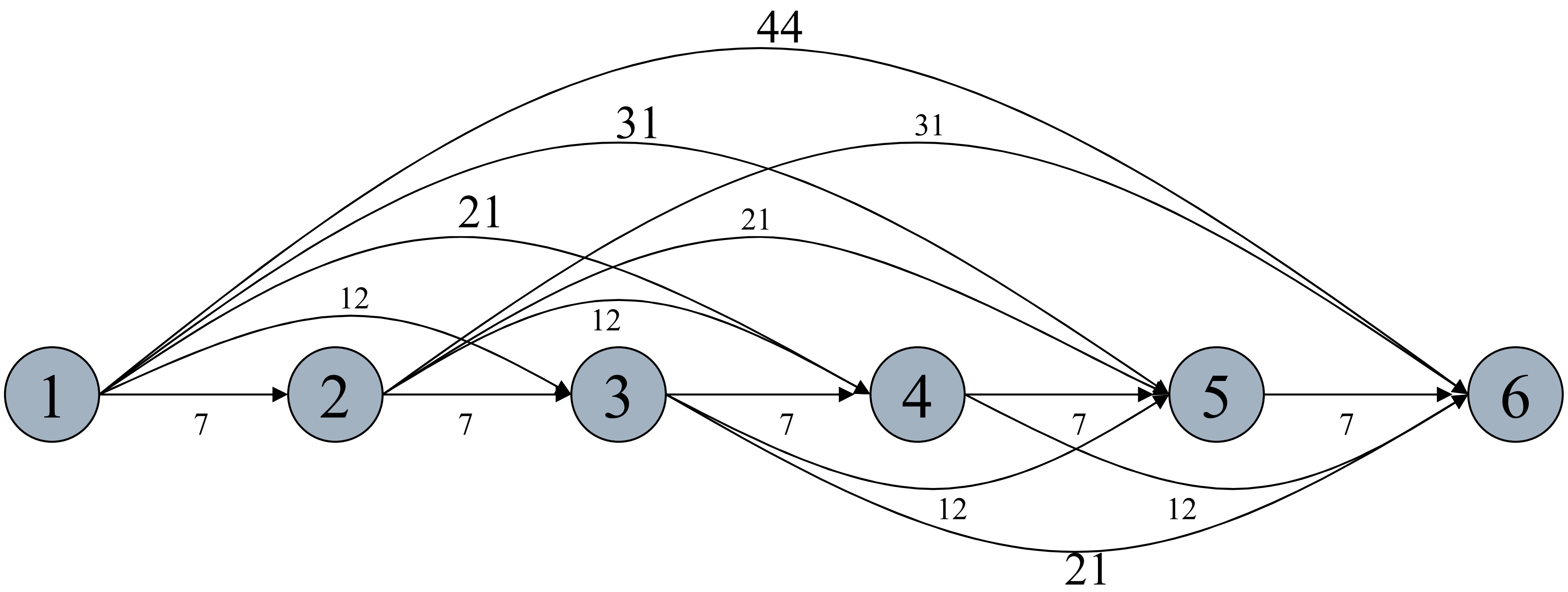
$c_{12}=2+12-7=7$
$c_{13}=2+4+12-6=12$
$c_{14}=2+4+5+12-2=21$
$c_{15}=2+4+5+9+12-1=31$
$c_{16}=2+4+5+9+12+12-0=44$

$c_{23}=2+12-7=7$
$c_{24}=2+4+12-6=12$
$c_{25}=2+4+5+12-2=21$
$c_{26}=2+4+5+9+12-1=31$
$c_{34}=2+12-7=7$

$c_{35}=2+4+12-6=12$
$c_{36}=2+4+5+12-2=21$
$c_{45}=2+12-7=7$
$c_{46}=2+4+12-6=12$
$c_{56}=2+12-7=7$

Ex. 2: Solution continued

- From the figure below we can see that paths 1-3-4-6, 1-3-5-6, and 1-2-4-6 give us the shortest path with a value of 31.



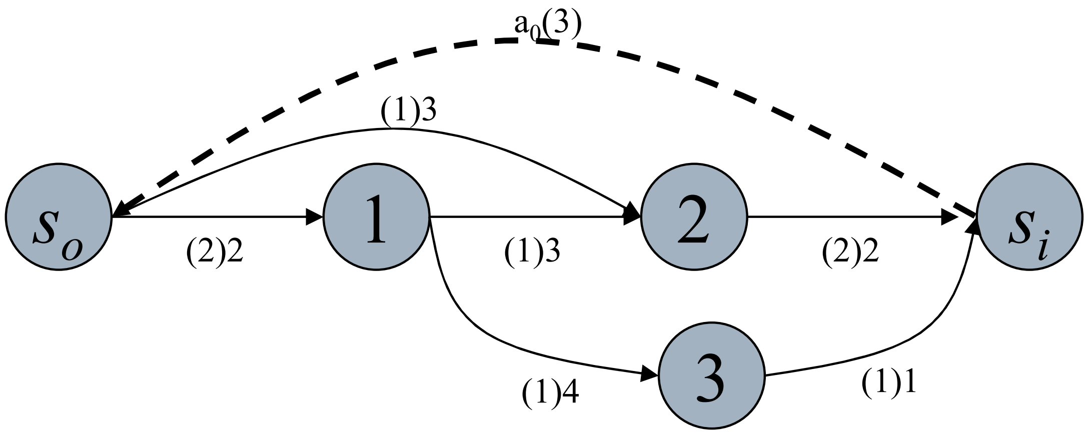
- Assuming that all arc lengths are non-negative, **Dijkstra's algorithm**, can be used to find the shortest path from a node.
- Finding the shortest path between node *i* and node *j* in a network may be viewed as a transshipment problem.

7 Maximum Flow Problems

- Many situations can be modeled by a network in which the arcs may be thought of as having a capacity that limits the quantity of a product that may be shipped through the arc.
- In these situations, it is often desired to transport the maximum amount of flow from a starting point (called the **source**) to a terminal point (called the **sink**).
- These types of problems are called **maximum flow problems**.

Example 3: Maximum Flow

- Sunco Oil wants to ship the maximum possible amount of oil (per hour) via pipeline from node s_o to node s_i.



Arc	*Capacity*
(*so*,1)	2
(*so*,2)	3
(1,2)	3
(1,3)	4
(3,*si*)	1
(2,*si*)	2

- The various arcs represent pipelines of different diameters.
- **arc capacity:** the maximum number of barrels of oil that can be pumped through each arc
- Formulate an LP that can be used to determine the maximum number of barrels of oil per hour that can be send from *so* to *si*.

Ex. 3: Solution

- An artificial arc called a_0 is added from the sink to the source.
- First determine the decision variable
  - x_{ij} = Millions of barrels of oil per hour that will pass through arc(i,j) of pipeline.
- For a flow to be feasible it must have two characteristics:
  - 0 <= flow through each arc <= arc capacity
  - Flow into node i = Flow out from node i

Ex. 3: Solution continued

- Let x_0 be the flow through the artificial arc, the conservation of flow implies that x_0 = total amount of oil entering the sink.
- Sunco's goal is to maximize x_0.

Max Z= X0

S.t.	Xso,1<=2	(Arc Capacity constraints)
	Xso,2<=3	
	X12 <=3	
	X2,si<=2	
	X13 <=4	
	X3,si<=1	
	X0 =Xso,1+Xso,2	(Node so flow constraints)
	Xso,1=X12+X13	(Node 1 flow constraints)
	Xso,2+X12 =X2,si	(Node 2 flow constraints)
	X13 =X3,si	(Node 3 flow constraints)
	X3,si+X2,si=X0	(Node si flow constraints)
	Xij>=0	

- One optimal solution to this LP is Z=3, $x_{so,1}$=2, x_{13}=1, x_{12}=1, $x_{so,2}$=1, $x_{3,si}$=1, $x_{2,si}$=2, x_o=3.

LINGO code

- The maximum flow in a network can be found using LINDO, but LINGO greatly lessens the effort needed to communicate the necessary information.

```
MODEL:
   1]SETS:
   2]NODES/1..5/;
   3]ARCS(NODES,NODES)/1,2   1,3   2,3   2,4   3,5   4,5   5,1/
   4]:CAP,FLOW;
   5]ENDSETS
   6]MAX=FLOW (5,1);
   7]@FOR(ARCS(I,J):FLOW(I,J)<CAP(I,J));
   8]@FOR(NODES(I):@SUM(ARCS(J,I):FLOW(J,I))
   9]=@SUM(ARCS(I,J):FLOW(I,J)));
  10]DATA:
  11]CAP=2,3,3,4,2,1,1000;
  12]ENDDATA
END
```

- If some nodes are identified by numbers, then LINGO will not allow you to identify other nodes with names involving letters.
- Begin by listing the network's nodes in line 2. Then list the network's arcs in line 3. Finally, list the capacity of each arc in the network in line 11, and you are ready to find the maximum flow in the network!

- Choose any set of nodes *V'* that contains the sink but does not contain the source. Then the set of arcs *(i,j)* with *i* not in *V'* and *j* a member of *V'* is a **cut** for the network.
- The **capacity** of a cut is the sum of the capacities of the arcs in the cut.
- The flow from source to sink for any feasible flow is less than or equal to the capacity of *any* cut.
- Max flow = Min capacity of cuts
- Important questions – Ford-Fulkerson Method
  - Given a feasible flow, how can we tell if it is an optimal flow (that is maximizes $x0$)?
  - If a feasible flow is nonoptimal, how can we modify the flow to obtain a new feasible flow that has a larger flow from the source to the sink?