BDC5101

Deterministic Operations Research Models

Zhenyu Hu

Semester II, 2018/2019

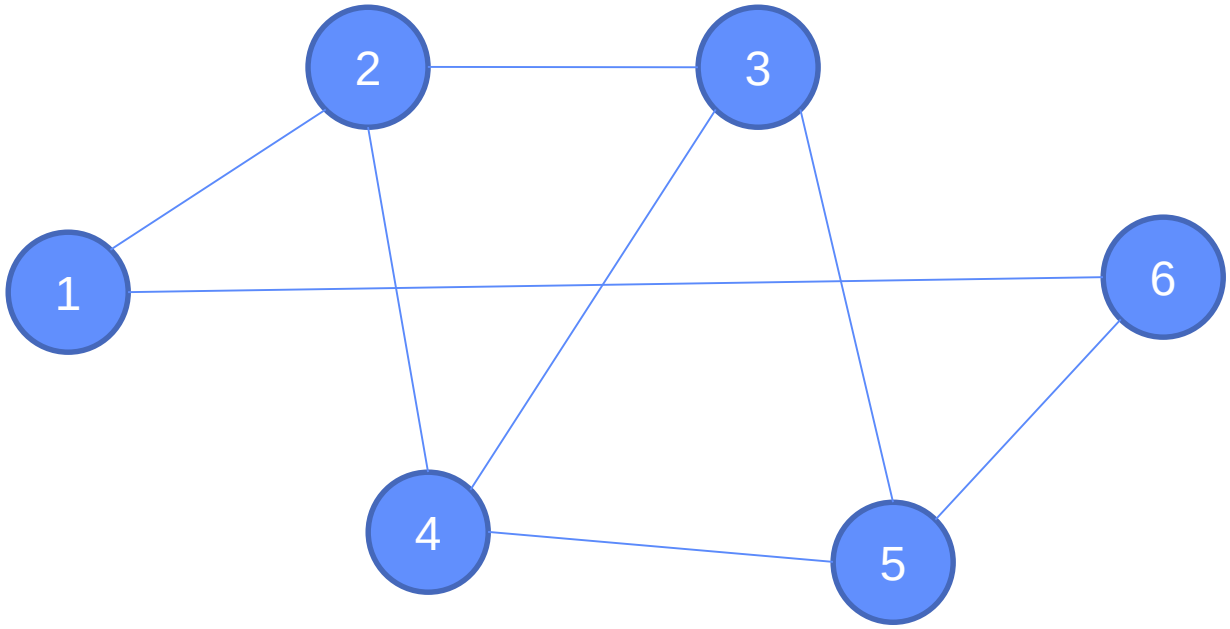
Network Flow Models

- Basic Graph Theory
- General Network-Flow Problem
- Special Network Models
- Solution Methods
- Application of Network Models

Basic Graph Theory

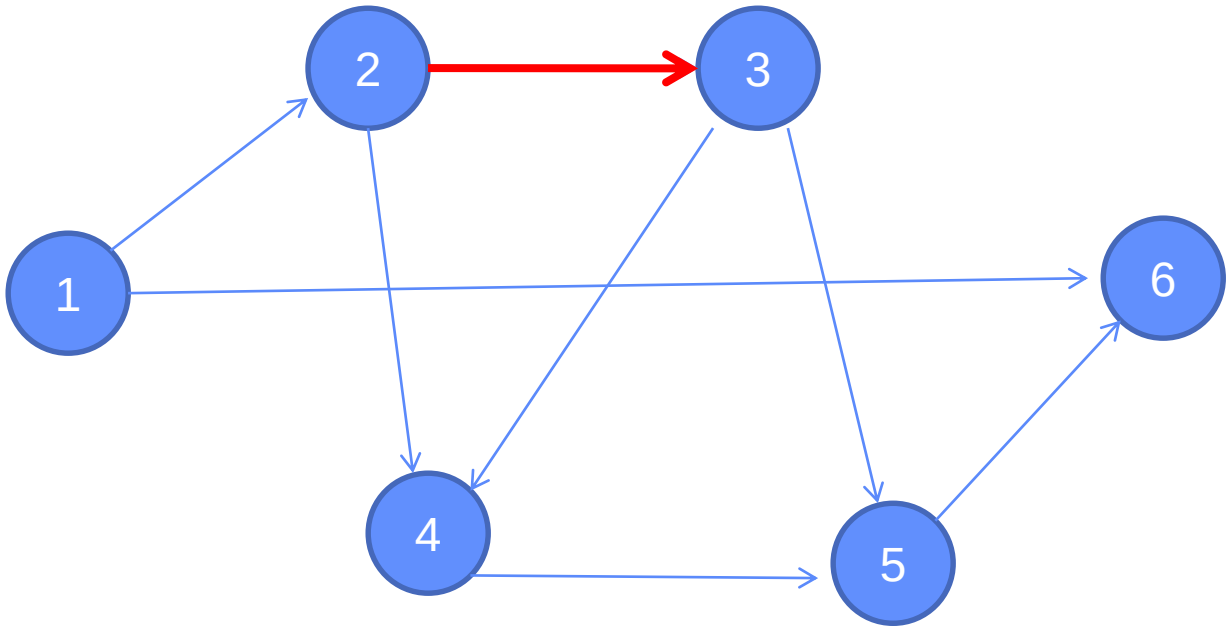
Undirected Network

- **A network is denoted as G = (N, A), where N is the set nodes and A is the set of undirected arcs.**



Directed Network

- **A network whose arcs are all ordered pairs of distinct nodes.**

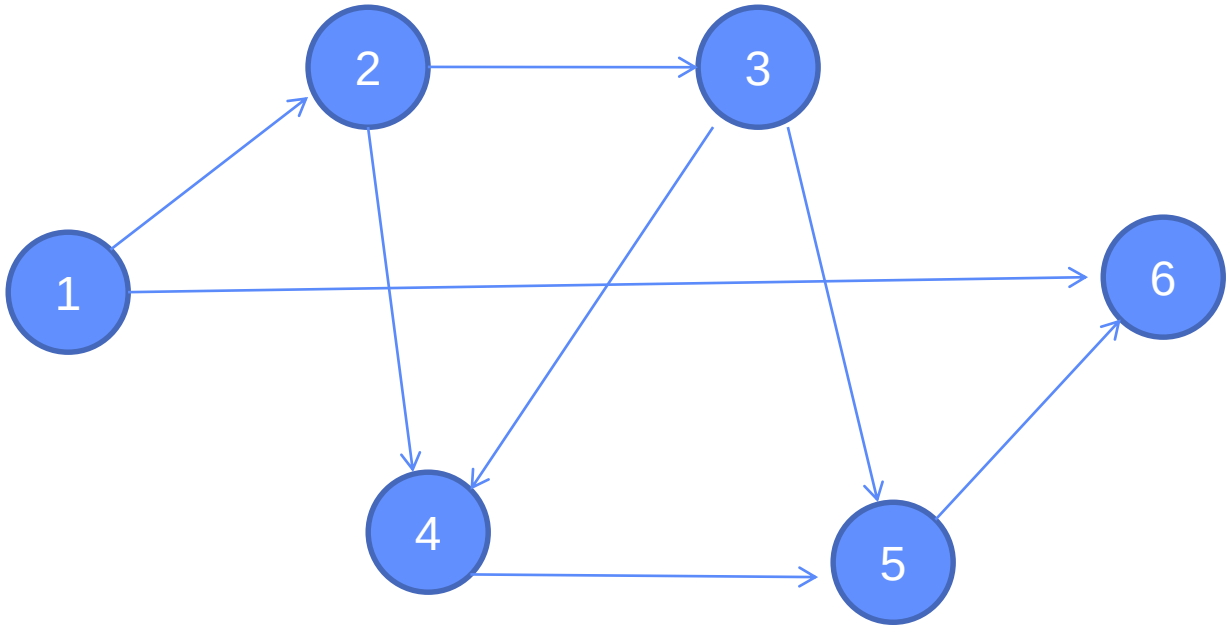


Degree

- The *indegree* of a node is the number of incoming arcs of that node and its *outdegree* is the number of outgoing arcs.
- The *degree* of a node is the sum of its indegree and outdegree.

Path

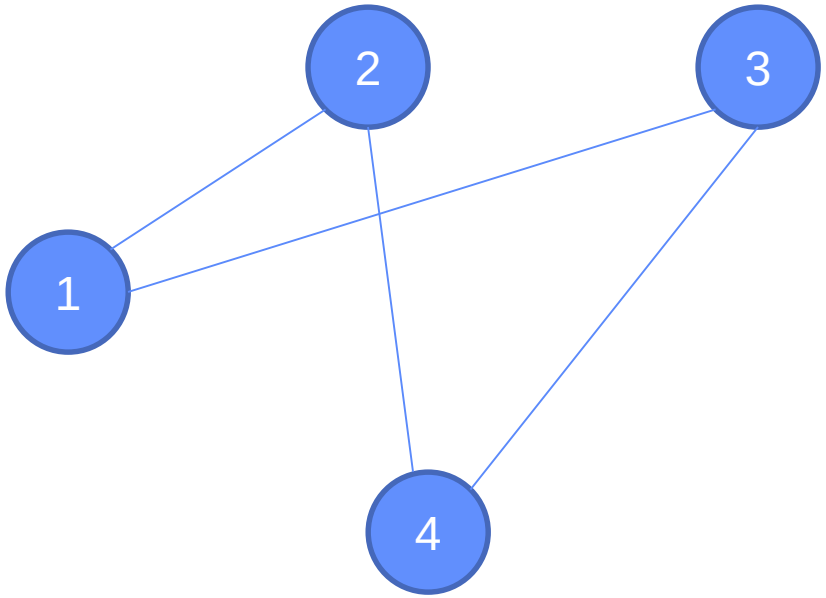
- **A *path* is a sequence of arcs which connect a sequence of distinct nodes. *Directed path* requires the arcs to be in the same direction**



Cycle

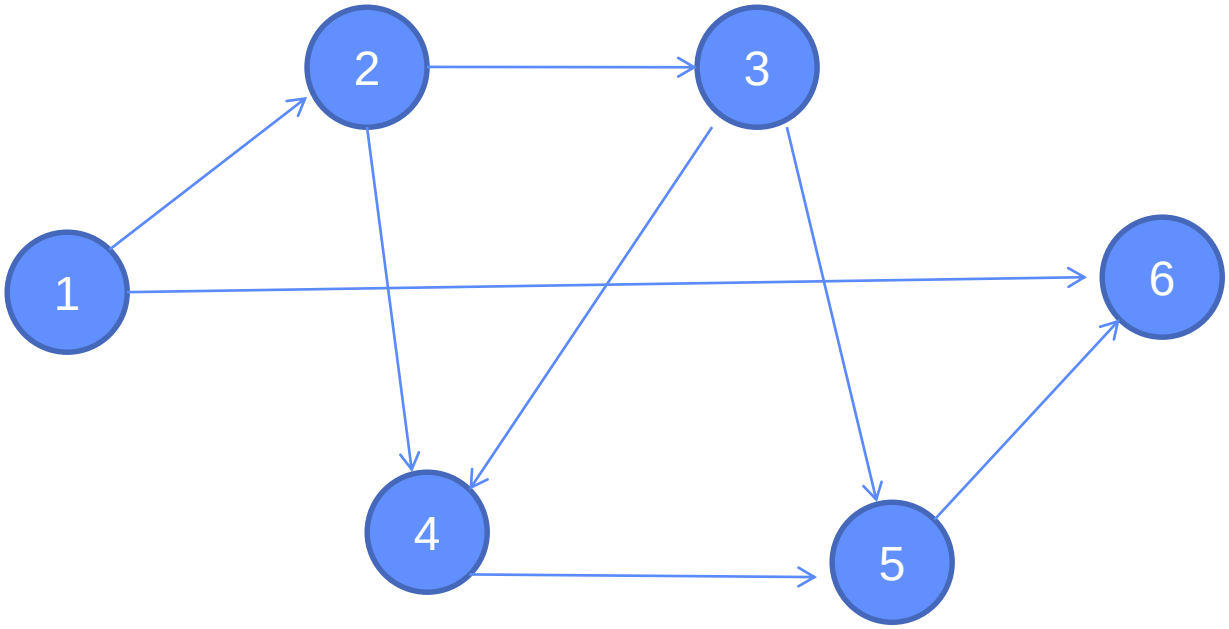
- **A *cycle* is a path** $i_1 - a_1 - i_2 - a_2 - \ldots - i_{r-1} - a_{r-1} - i_r$

  together with the arc (i_1, i_r) or (i_r, i_1)



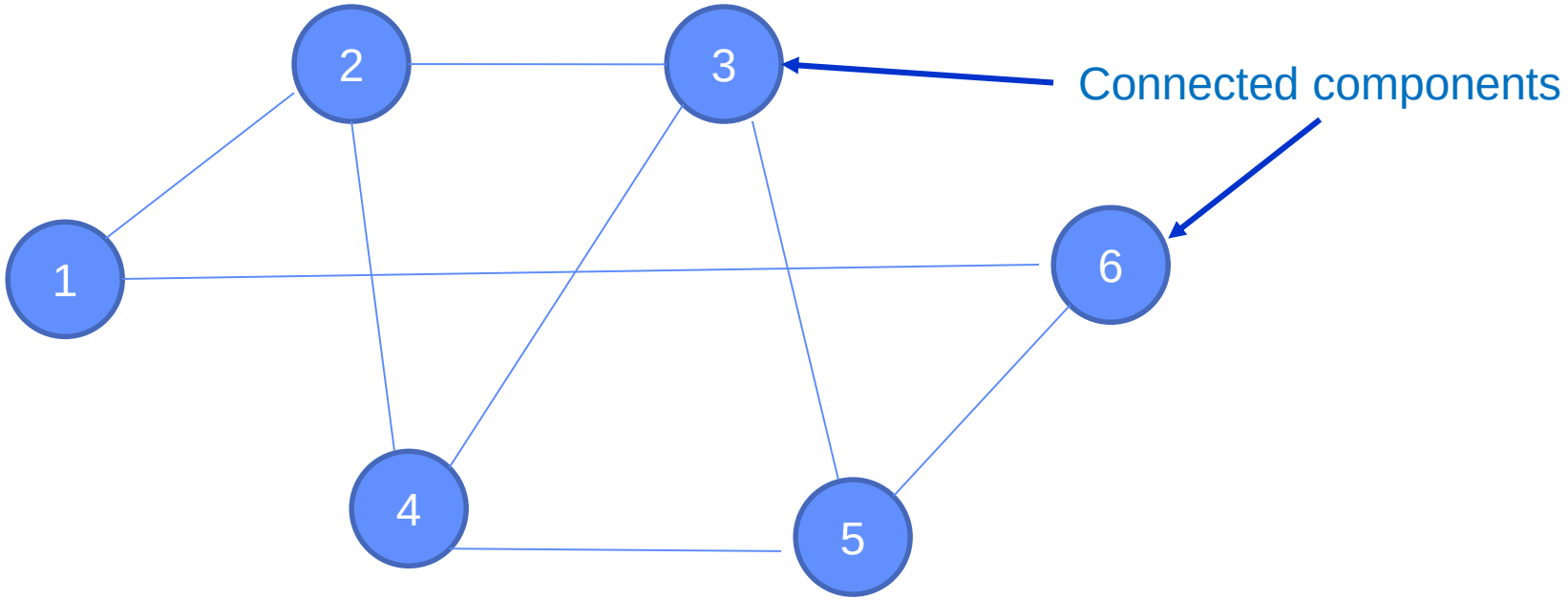
Acyclic network

- **A (directed) graph is *acyclic* if it contains no (directed) cycle.**



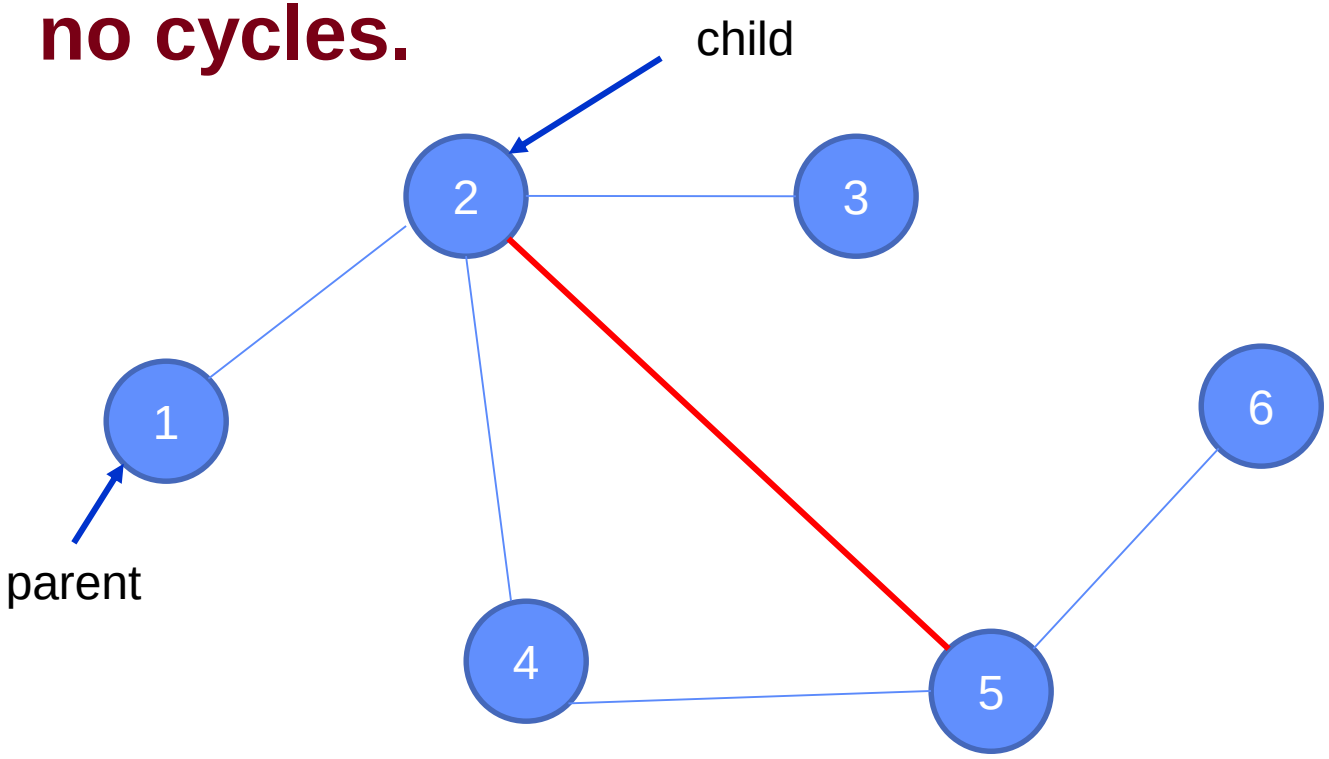
Connectivity

- **Two nodes *i* and *j* are *connected* if the graph contains at least one path from node *i* to node *j*.**
- **A graph is *connected* if every pair of its nodes is connected. Otherwise, the graph is *disconnected*.**



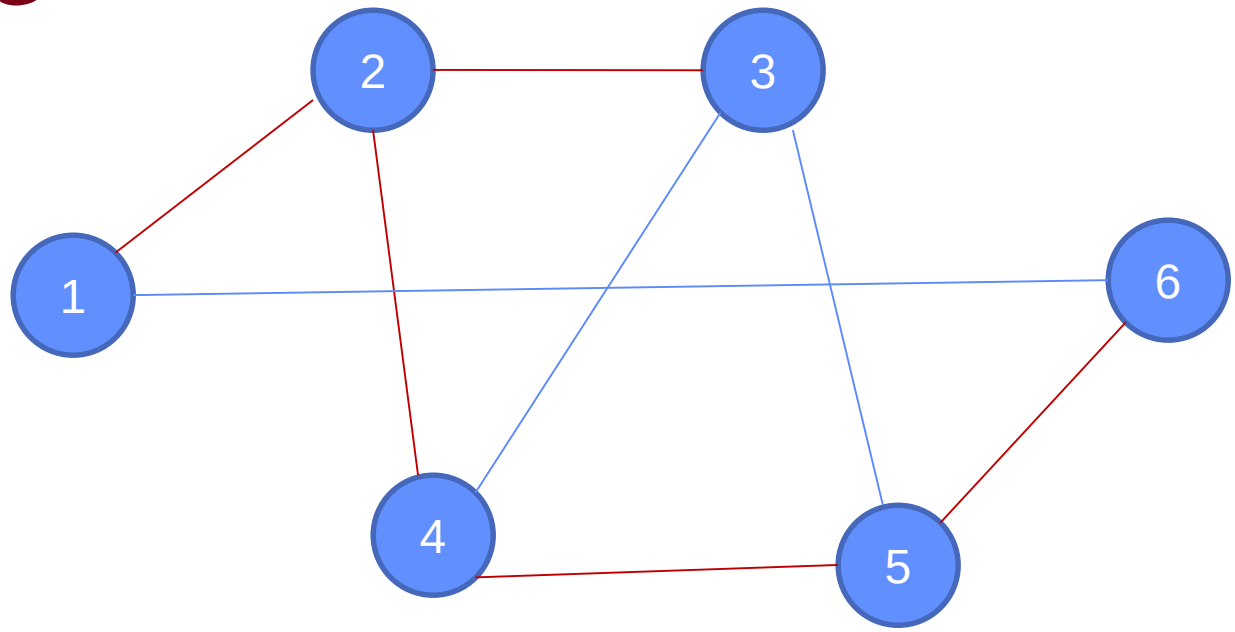
Tree

- **A *tree* is a connected graph that contains no cycles.**



Spanning Tree

- **A *spanning tree* T of a connected graph G satisfies: T is a tree and T contains every node in G**



Obtaining a Spanning Tree: Prim's Algorithm

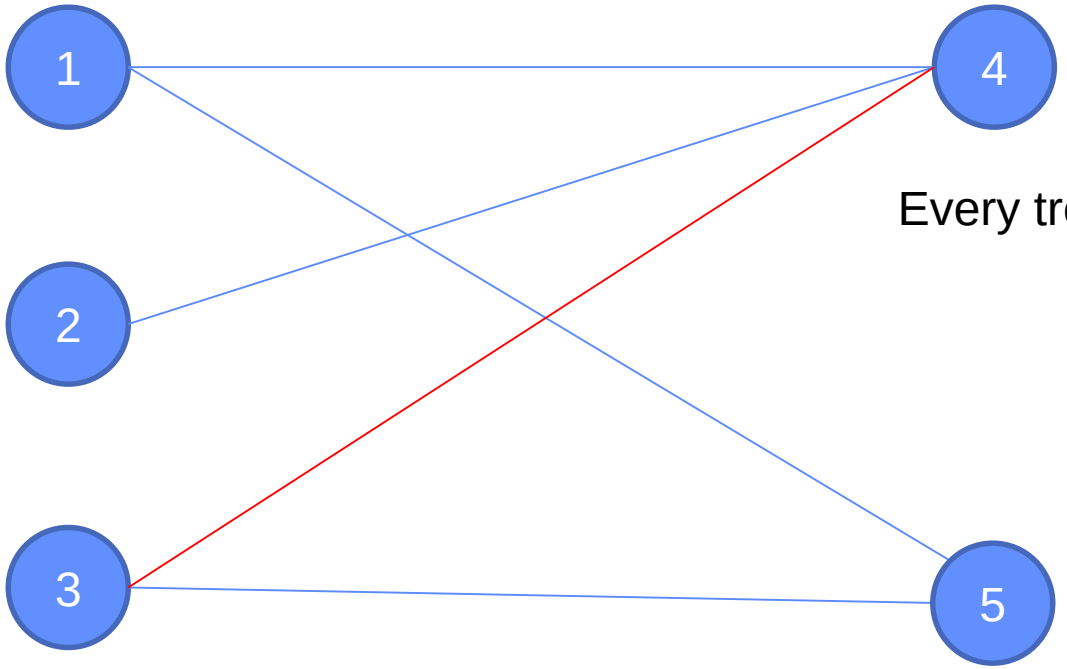
- Starting with N isolated nodes and adding arcs one at a time.
- Each time you add an arc, you either
  - Connect two components together or
  - ~~Create a cycle~~
- Stop when the graph is connected.

Properties of Spanning Tree

- There are exactly N – 1 arcs; it is acyclic; it connects all N nodes.
- Conversely, if a set of N – 1 arcs is connected to all N nodes, then it is a spanning tree.

Other Special Graphs: Bipartite Graph

- **The node is separated into two sets: arcs only connect nodes in different sets.**

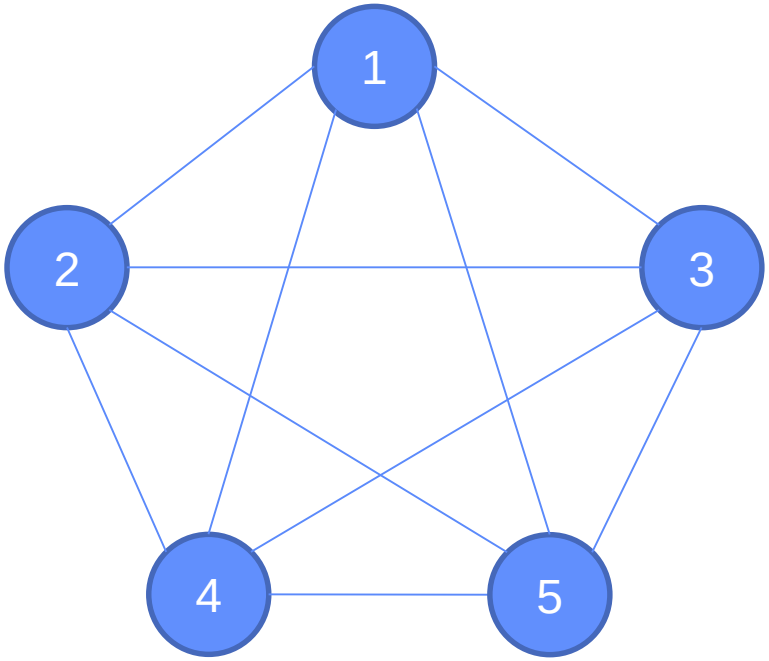


Every tree is a bipartite graph, but not vice versa.

Other Special Graphs: Complete Graph

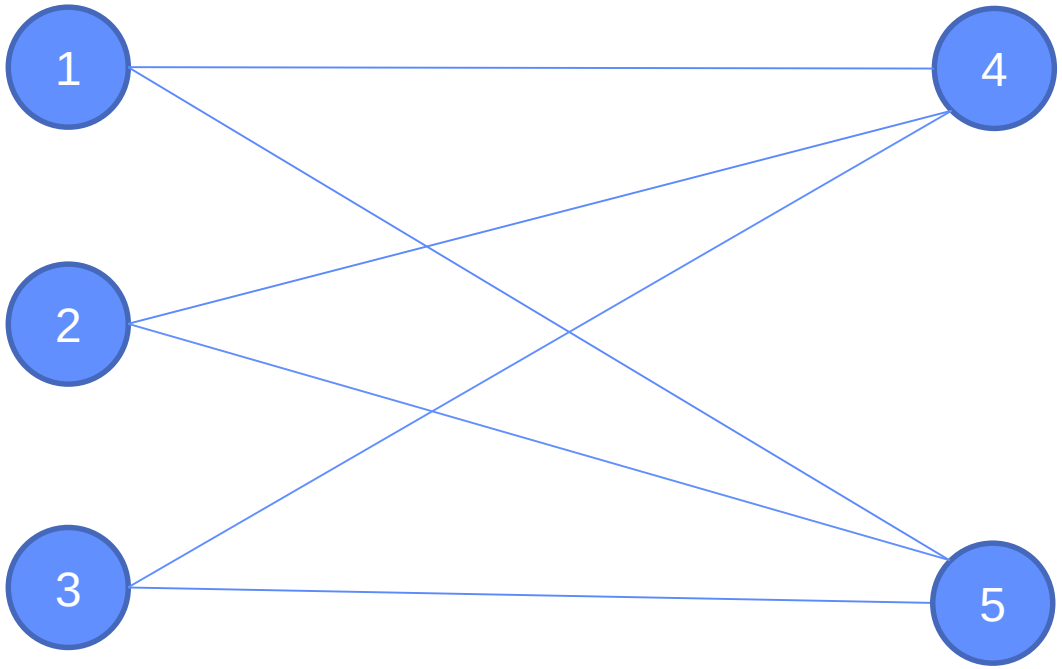
- **Every pair of node is connected by an arc.**

How many arcs are in a complete graph with n nodes? n(n-1)/2



Other Special Graphs: Complete Bipartite Graph

- **Every node in one set is connected with every other node in the other set.**



General Network-Flow Problem

Network Flow Framework

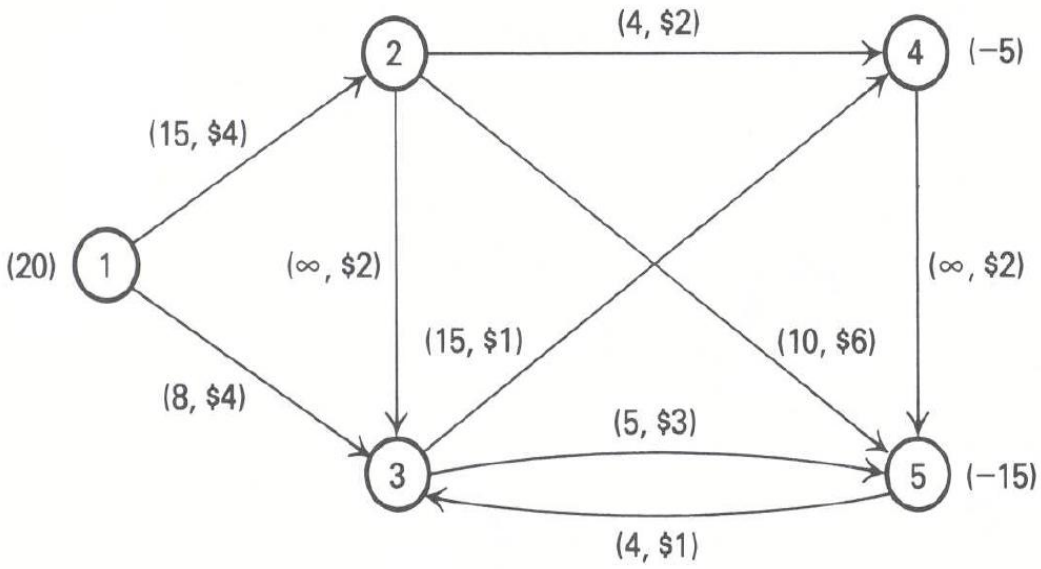
Let $G = (N, A)$ be a directed network with a cost c_{ij}, a lower bound l_{ij}, and a capacity u_{ij} associated with each arc $(i, j) \in A$.

We associate with each node $i \in N$ an integer number $b(i)$.

If ***b(i)>0***, the node i is called a ***supply*** node.

If ***b(i)<0***, the node i is called a ***demand*** node.

If ***b(i)=0***, the node i is called a ***transshipment*** node.



Assume balanced: $\sum_i b_i = 0$

Minimum Cost Network Flow Problem

Let x_{ij}, where $(i, j) \in A$, be the flow along arc (i, j).

Flow Balance Constraint

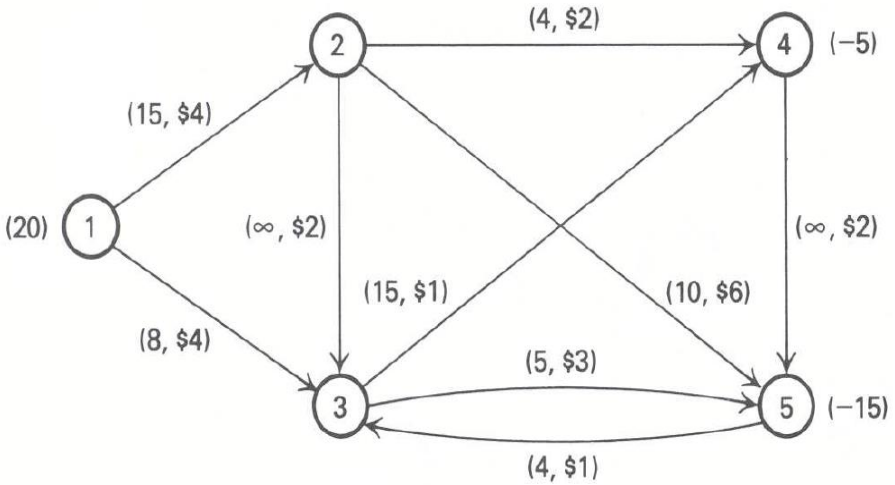
$$\min \quad \sum_{(i,j)\in A} c_{ij} x_{ij}$$

$$s.t. \quad \sum_{\{j:(i,j)\in A\}} x_{ij} - \sum_{\{j:(j,i)\in A\}} x_{ji} = b(i), \quad \text{for all } i \in N$$

$$l_{ij} \le x_{ij} \le u_{ij}, \quad \text{for all } (i, j) \in A$$

In this course, unless specified, we set $l_{ij} = 0$.

Network Flow Problem: LP Tableau



	x_{12}	x_{13}	x_{23}	x_{24}	x_{25}	x_{34}	x_{35}	x_{45}	x_{53}	*Righthand side*
Node 1	1	1								20
Node 2	−1		1	1	1					0
Node 3		−1	−1			1	1		−1	0
Node 4				−1		−1		1		−5
Node 5					−1		−1	−1	1	−15
Capacities	15	8	∞	4	10	15	5	∞	4	
Objective function	4	4	2	2	6	1	3	2	1	(Min)

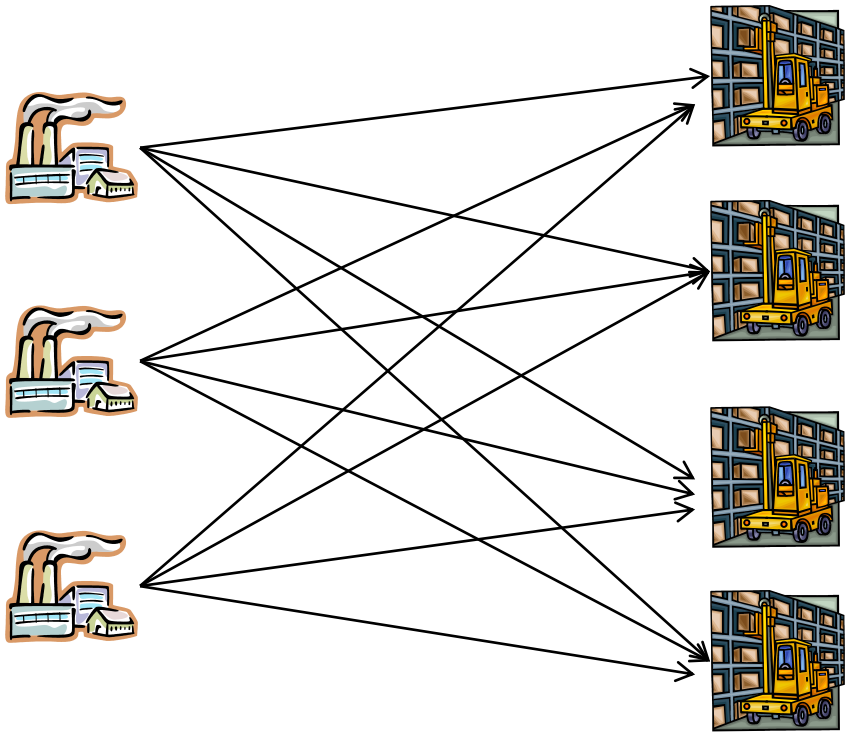
Network Flow Problem: Special Structure

- The coefficients of x_{ij} in the constraint matrix is either 0, +1 or -1.
- Each column in the constraint matrix has exactly 2 non-zero entries.
- The two non-zero entries are respectively +1 and -1 for all columns.

Special Network Models

Transportation Problem

- ***m*** **plants,** ***n*** **distribution centers.**
- a_i **supply from plant** ***i*****,** ***i=1,2,…,m***
- b_j **demand of distribution center** ***j*****,** ***j=1,2,…,n***
- c_{ij} **cost of transportation from** ***i*** **to** ***j***



Transportation Problem is a special case of min-cost flow problem. (why?)

Assume balanced: $\sum_i a_i = \sum_j b_j$

Transportation Problem: Network Formulation

$$\text{Minimize } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij},$$

subject to:

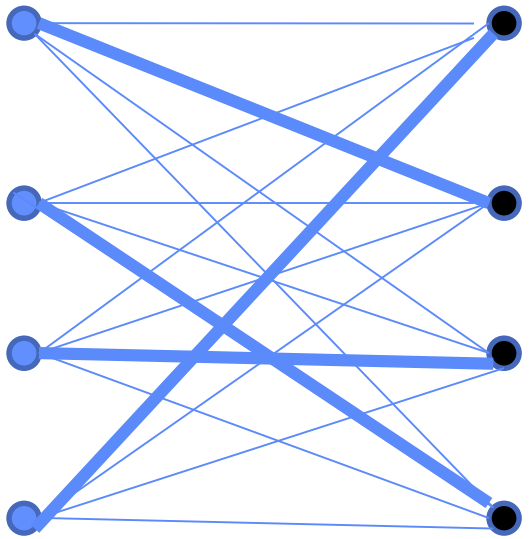
$$\sum_{j=1}^{n} x_{ij} = a_i \qquad (i = 1, 2, \ldots, m),$$

$$\sum_{i=1}^{m} (-x_{ij}) = -b_j \qquad (j = 1, 2, \ldots, n),$$

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

Assignment Problem

- **Determine the minimal cost of assigning elements of a set A to elements of another set B.**
- **Applications**
  - **People to projects**
  - **Jobs to machines**
  - **Boarding gate to aircrafts**
  - **Graduates to internships**
  - **…**



Assignment Problem: Network Formulation

$$\text{Minimize } z = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij},$$

subject to:

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

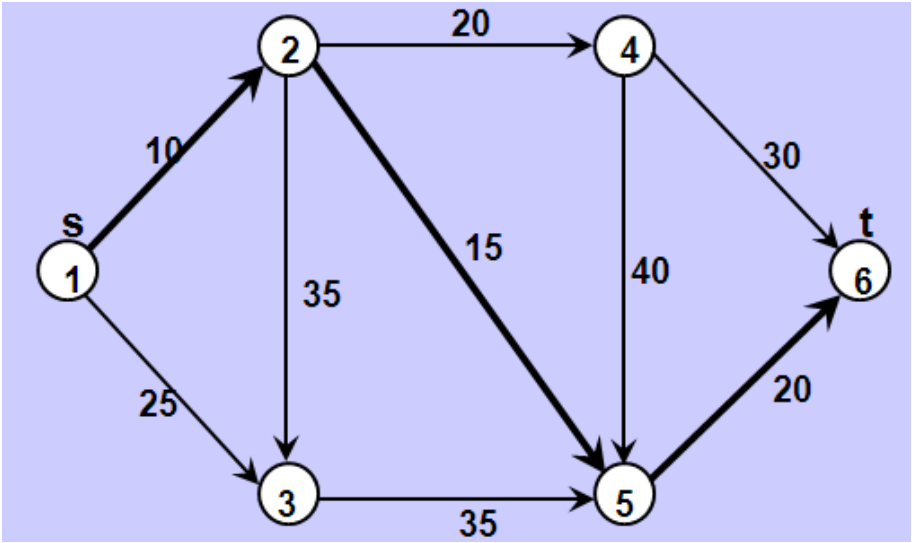
$$\sum_{i=1}^{n} x_{ij} = 1 \qquad (j = 1, 2, \ldots, n),$$

$$x_{ij} = 0 \quad \text{or} \quad 1 \qquad (i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n).$$

What if the graph is not a complete graph?

Shortest Path Problem

- **Identify the shortest path from a source node to a sink node through the given network.**
  - **Finding a path of minimum length**
  - **Finding a path taking minimum time**
  - **Finding a path of maximum reliability.**



Shortest Path Problem: Network Formulation

$$\text{Minimize } z = \sum_i \sum_j c_{ij} x_{ij},$$

subject to:

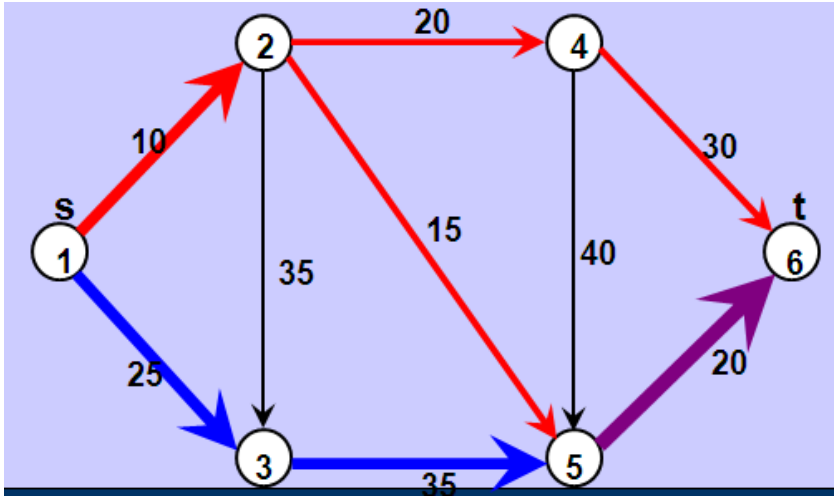
$$\sum_j x_{ij} - \sum_k x_{ki} = \begin{cases} 1 & \text{if } i = s \text{ (source)}, \\ 0 & \text{otherwise}, \\ -1 & \text{if } i = t \text{ (sink)} \end{cases}$$

$$x_{ij} \geq 0 \quad \text{for all arcs } i\text{–}j \text{ in the network.}$$

$$x_{ij} = 0 \quad \text{or} \quad 1$$

Maximum Flow Problem

- **Determine the maximum flow that can be sent from a given source node to a sink node in a capacitated network.**
- **Determine maximum steady-state flow of**
  - **petroleum products in a pipeline network,**
  - **cars in a road network,**
  - **message in a telecommunication network,**
  - **electricity in an electrical network.**



Maximum Flow: Network Formulation

Maximize v,

subject to:

$$\sum_j x_{ij} - \sum_k x_{ki} = \begin{cases} v & \text{if } i = s \text{ (source)}, \\ -v & \text{if } i = t \text{ (sink)}, \\ 0 & \text{otherwise}, \end{cases}$$

$$0 \le x_{ij} \le u_{ij} \qquad (i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n).$$

Maximize x_{ts},

subject to:

$$\sum_j x_{ij} - \sum_k x_{ki} = 0 \qquad (i = 1, 2, \ldots, n),$$

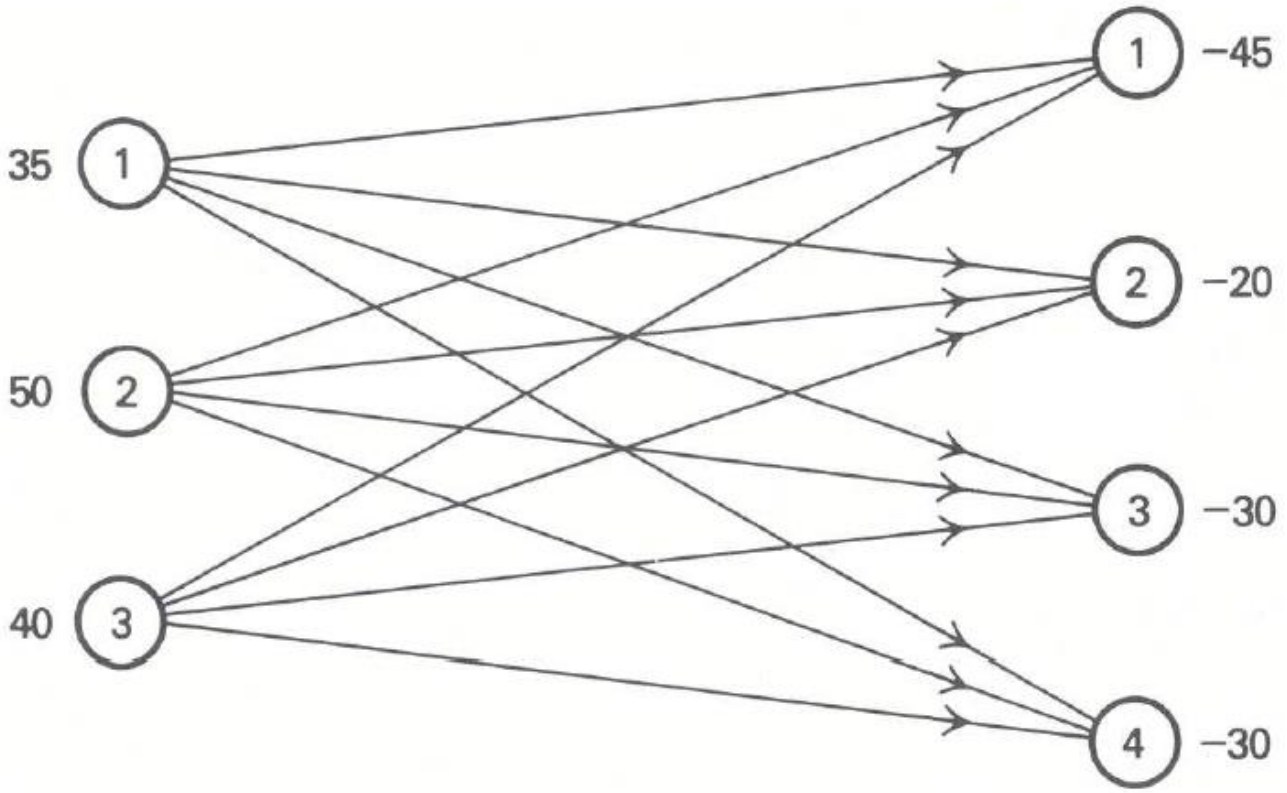
$$0 \le x_{ij} \le u_{ij} \qquad (i = 1, 2, \ldots, n;\ j = 1, 2, \ldots, n).$$

Solution Methods

Transportation Problem: Example

Plants	*Distribution centers*				*Availability* (units)
	Dallas	Atlanta	San Francisco	Phila.	
Cleveland	8	6	10	9	35
Chicago	9	12	13	7	50
Boston	14	9	16	5	40
Requirements (units)	45	20	30	30	[125]

Transportation Problem: Example

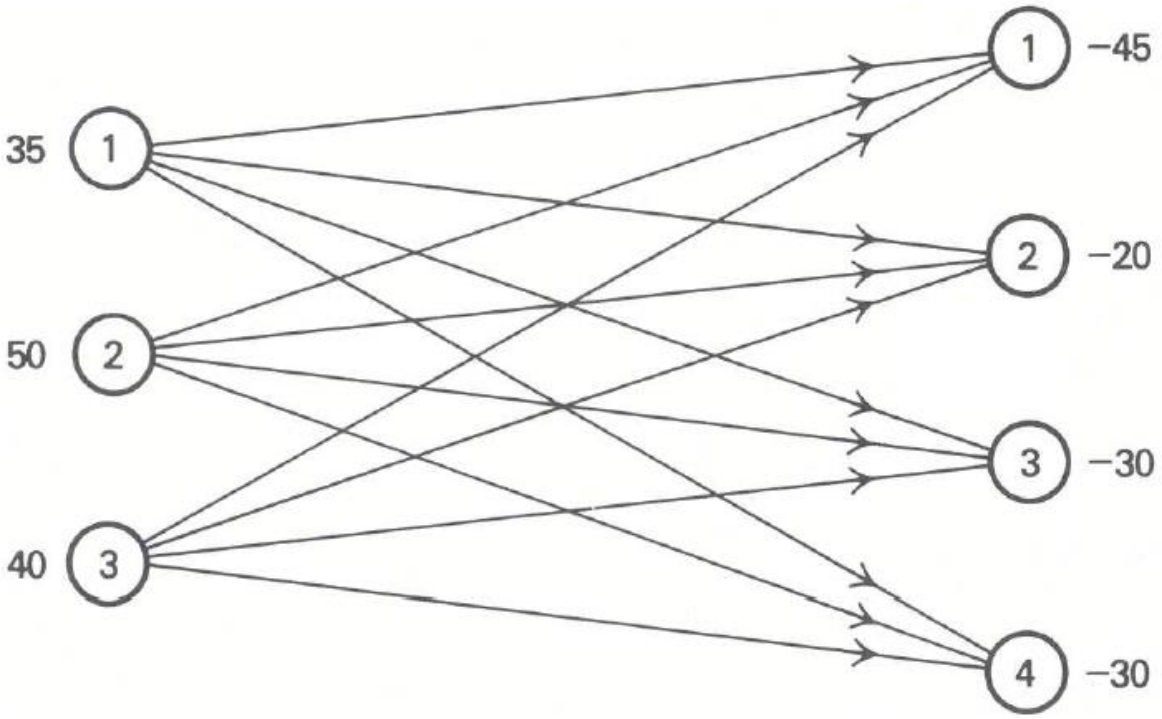


Solving Transportation Problem (TP): Procedure

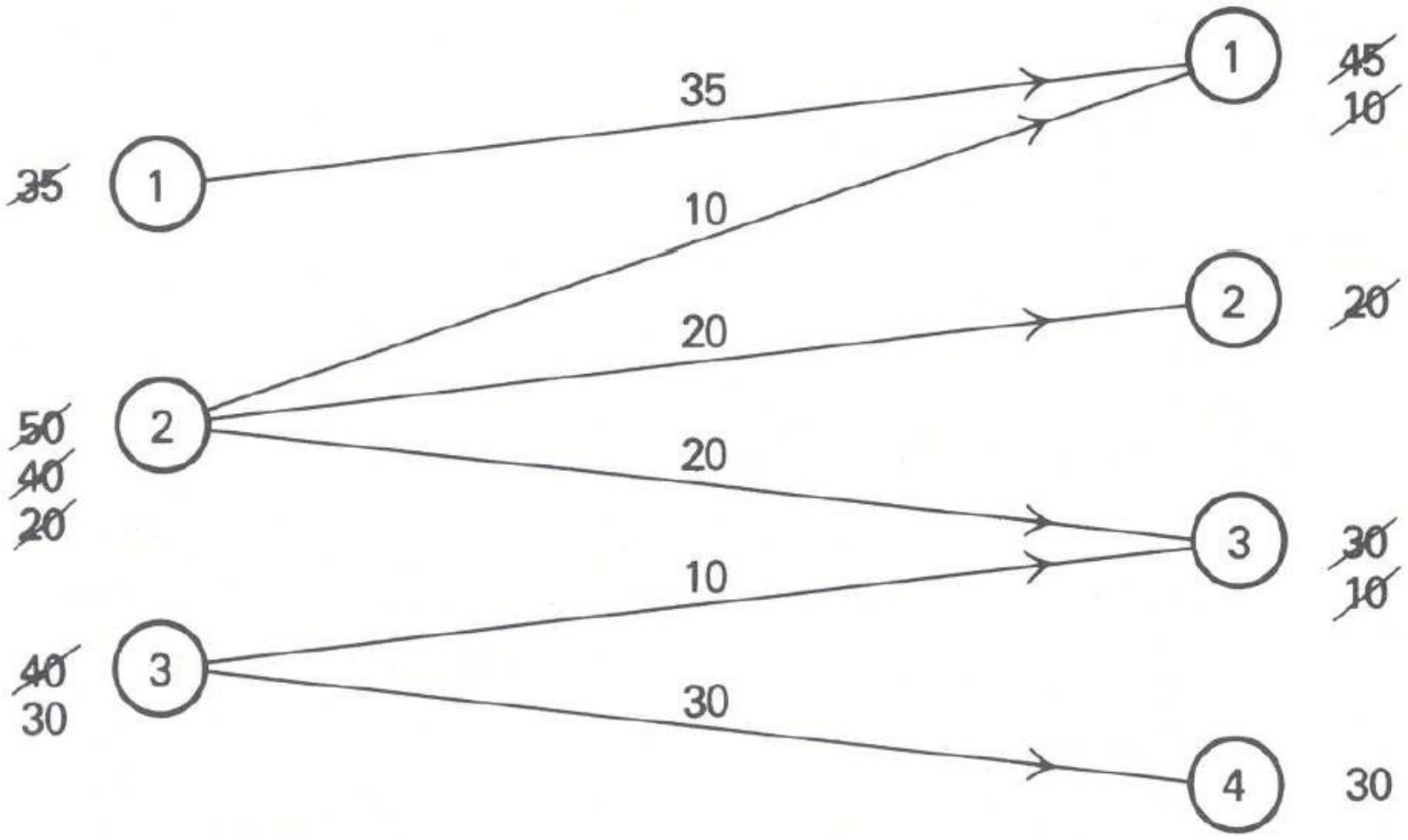
- **Obtaining an initial solution;**
- **Checking optimality condition;**
- **Developing a procedure to improve the current solution if the optimality condition is not met**

Solving TP: Finding Initial Solution

Plants	*Distribution centers*				*Availability* (units)
	Dallas	Atlanta	San Francisco	Phila.	
Cleveland	8	6	10	9	35
Chicago	9	12	13	7	50
Boston	14	9	16	5	40
Requirements (units)	45	20	30	30	[125]

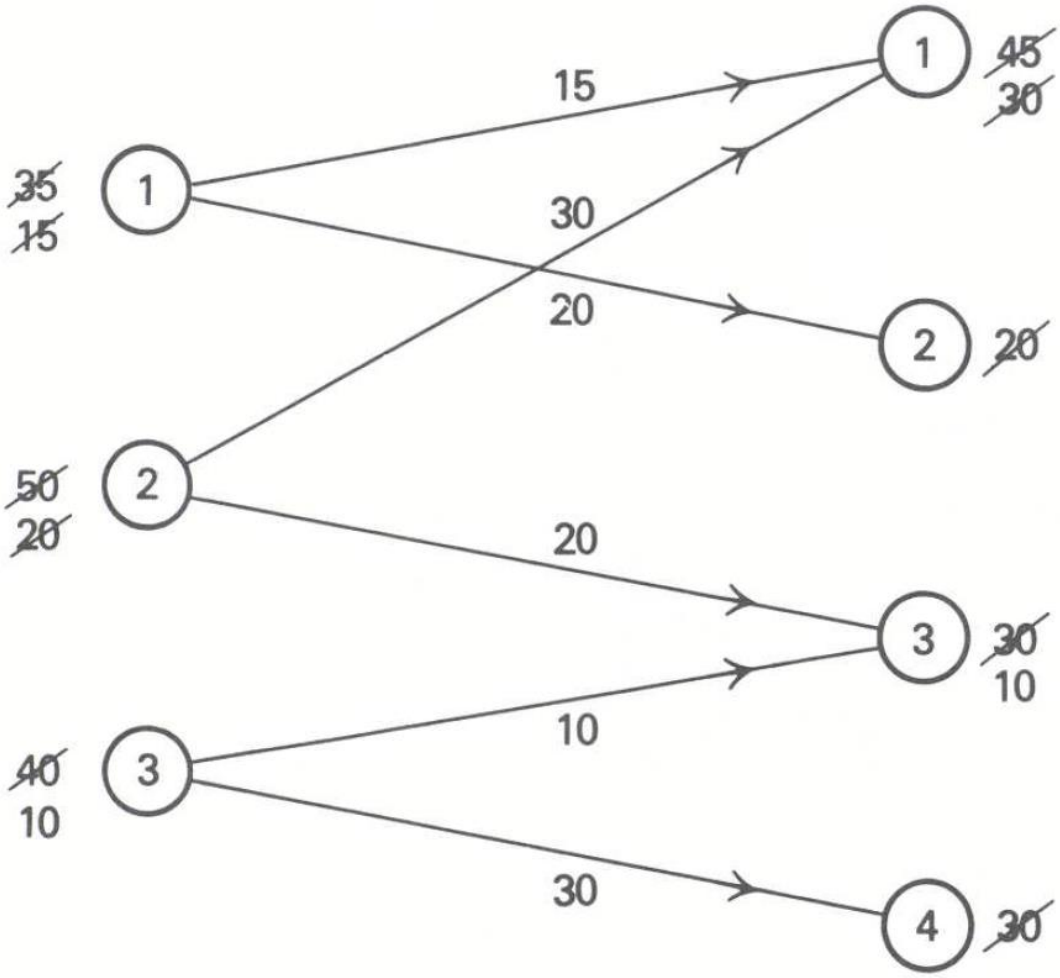


Finding Initial Solution : Myopic



Finding Initial Solution : Greedy Algorithm

Plants	*Distribution centers*				*Availability* (units)
	Dallas	Atlanta	San Francisco	Phila.	
Cleveland	8	6	10	9	35
Chicago	9	12	13	7	50
Boston	14	9	16	5	40
Requirements (units)	45	20	30	30	[125]

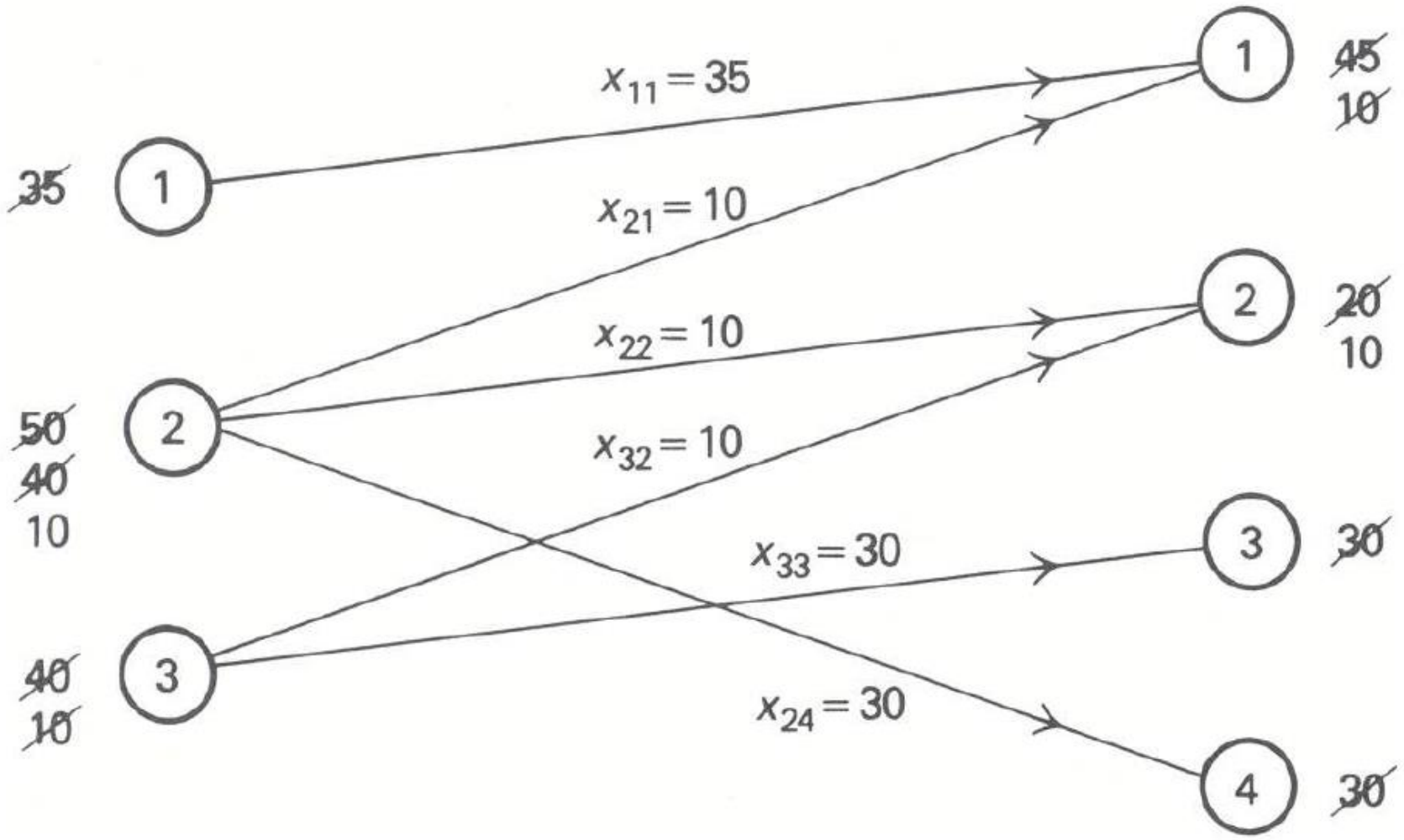


Properties of the Initial Solution

- **In both methods, the solutions have exactly (m + n - 1) arcs;**
- **Every node in the network is connected through the solution;**
- **There is no cycle in the solution.**

With the above properties, what do we know about the solution?

Finding Initial Solution : Spanning Tree



Spanning Tree vs. BS

- **Is a solution generated by spanning tree a basic solution?**
  - **Is the solution unique?**
- **Is a basic solution a spanning tree?**
  - **How many basic variables do we have?**
  - **It cannot have a cycle (otherwise one can have more than one solution to the equation).**

Observation: In transportation problem, a vector (not necessarily feasible) is a basic solution if and only if it is generated by a spanning tree.

Solving TP: Python Implementation

	Distribution centers				
Plants	Dallas	Atlanta	San Francisco	Phila.	*Availability* (units)
Cleveland	8	6	10	9	35
Chicago	9	12	13	7	50
Boston	14	9	16	5	40
Requirements (units)	45	20	30	30	[125]

$$\text{Minimize } z = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij},$$

subject to:

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

$$\sum_{i=1}^{m} (-x_{ij}) = -b_j \qquad (j = 1, 2, \ldots, n),$$

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

Solving TP: Python Implementation

```
from gurobipy import *
import numpy as np

#########Parameters Set-up############

#Objective coefficient: transportation cost from supply node i to demand node j
cost = np.array([[8, 6, 10, 9],
                 [9, 12, 13, 7],
                 [14, 9, 16, 5]])


#supply and demand
supply = np.array([35, 50, 40])

demand = np.array([45, 20, 30, 30])

#From the cost matrix, extract the number of supply nodes: M and the number of demand nodes: N
M, N = cost.shape
```

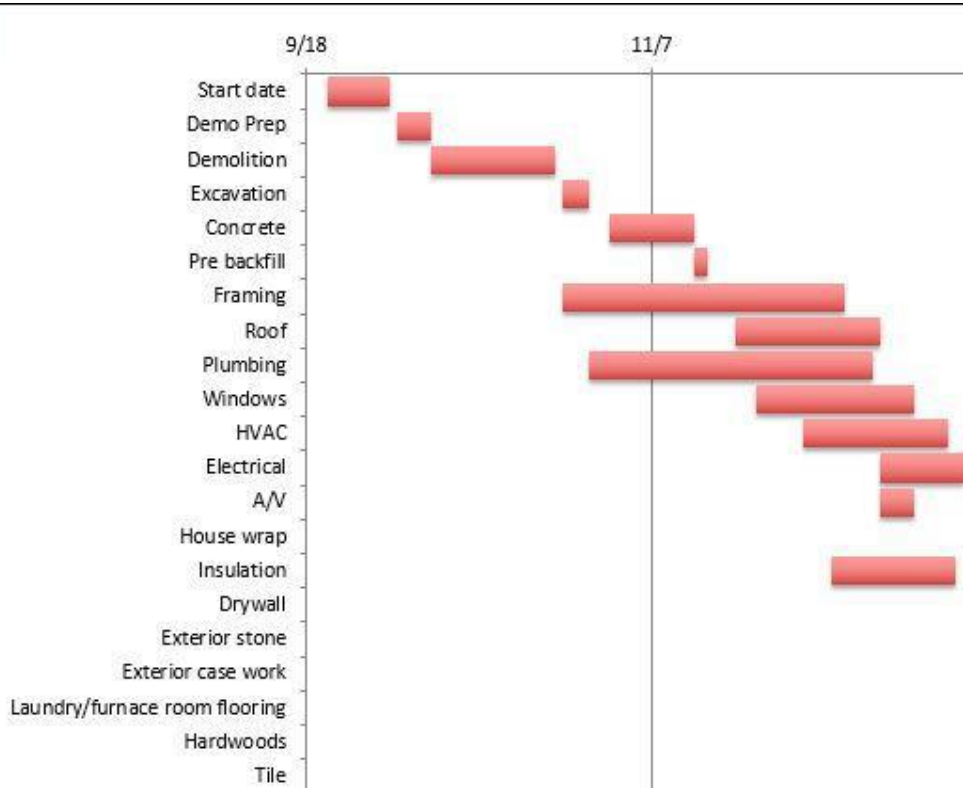
Solving TP: Python Implementation

```
#########Model Set-up###############

tp = Model("transportation")

# Creat variables
# addVars ( *indices, lb=0.0, ub=GRB.INFINITY, obj=0.0, vtype=GRB.CONTINUOUS, name="" )
x = tp.addVars(M, N)

# Set objective
tp.setObjective( quicksum(cost[i,j]*x[i,j] for i in range(M) for j in range(N)), GRB.MINIMIZE)

# Add supply constraints:
tp.addConstrs(( quicksum(x[i,j] for j in range(N)) == supply[i] for i in range(M) ), "Supply")

# Add demand constraints:
tp.addConstrs(( quicksum(x[i,j] for i in range(M)) == demand[j] for j in range(N) ), "Demand")

# Solving the model
tp.optimize()

#  Print optimal solutions and optimal value
for i in range(M):
    for j in range(N):
        print("\n Supply node %g to demand node %g amount: %g" % (i+1, j+1 , x[i,j].x))

print('Obj:', tp.objVal)
```

Application of Network Models

Project Management

Task Name	Start	End	Duration (days)
Start date	9/21/2014	9/30/2014	9
Demo Prep	10/1/2014	10/6/2014	5
Demolition	10/6/2014	10/24/2014	18
Excavation	10/25/2014	10/29/2014	4
Concrete	11/1/2014	11/13/2014	12
Pre backfill	11/13/2014	11/15/2014	2
Framing	10/25/2014	12/5/2014	41
Roof	11/19/2014	12/10/2014	21
Plumbing	10/29/2014	12/9/2014	41
Windows	11/22/2014	12/15/2014	23
HVAC	11/29/2014	12/20/2014	21
Electrical	12/10/2014	12/22/2014	12
A/V	12/10/2014	12/15/2014	5
House wrap	12/27/2014	12/30/2014	3
Insulation	12/3/2014	12/21/2014	18
Drywall	1/3/2015	1/23/2015	20
Exterior stone	1/3/2015	1/17/2015	14
Exterior case work	1/4/2015	1/13/2015	9
Laundry/furnace room flooring	2/16/2015	3/11/2015	23
Hardwoods	1/19/2015	3/10/2015	50
Tile	2/3/2015	2/14/2015	11

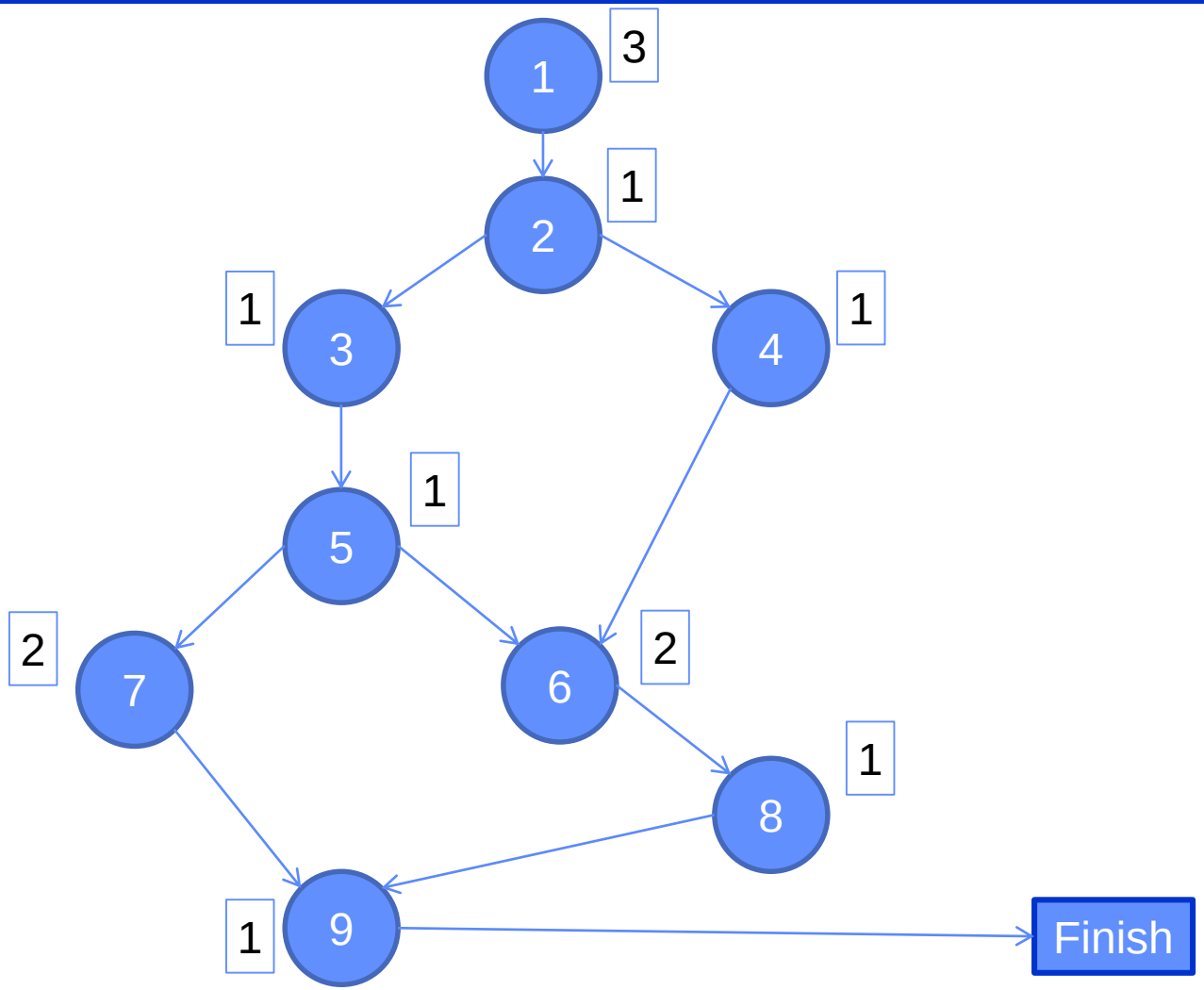


Project Management

Activity	Description	Immediate Predecessors	Estimated Duration
1	Wait until last 3 weeks	---	3 weeks
2	Looking for topic and data	1	1 week
3	Looking for reference	2	1 week
4	Cleaning data	2	1 week
5	Building model	3	1 week
6	Coding	4, 5	2 weeks
7	Writing motivation/model	5	2 weeks
8	Analyzing results	6	1 week
9	Finalizing report	7, 8	1 week

Deadline: 7 weeks

Activity-on-node (AON) Project Network



Crashing Projects

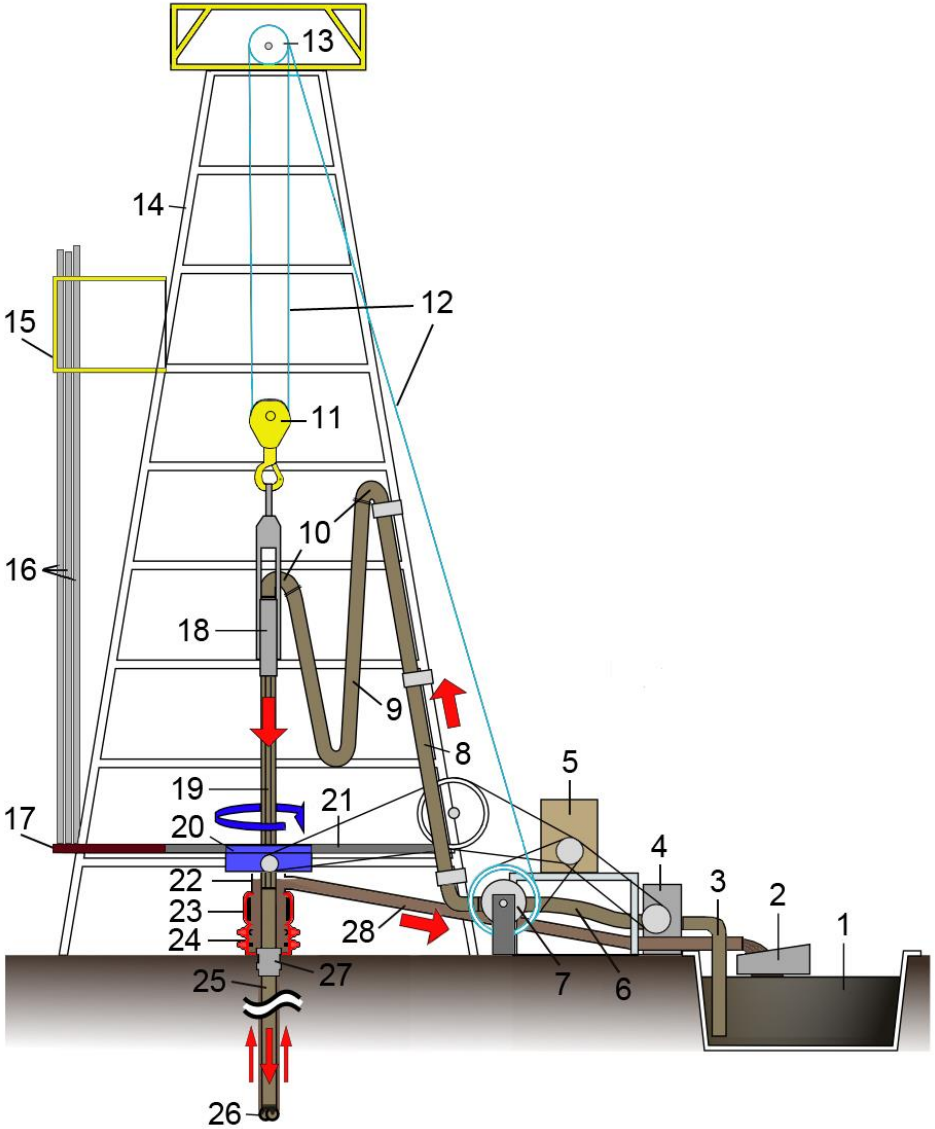
Activity	Description	Maximum Reduction Time	Crash Cost per Week
1	Wait until last 3 weeks	3 weeks	1
2	Looking for topic and data	0 week	0
3	Looking for reference	0 week	0
4	Cleaning data	½ week	2
5	Building model	½ week	2
6	Coding	1 week	2
7	Writing motivation/model	½ week	2
8	Analyzing results	0 week	0
9	Finalizing report	½ week	2

Objective: Minimizing the crash cost while meeting the deadline.

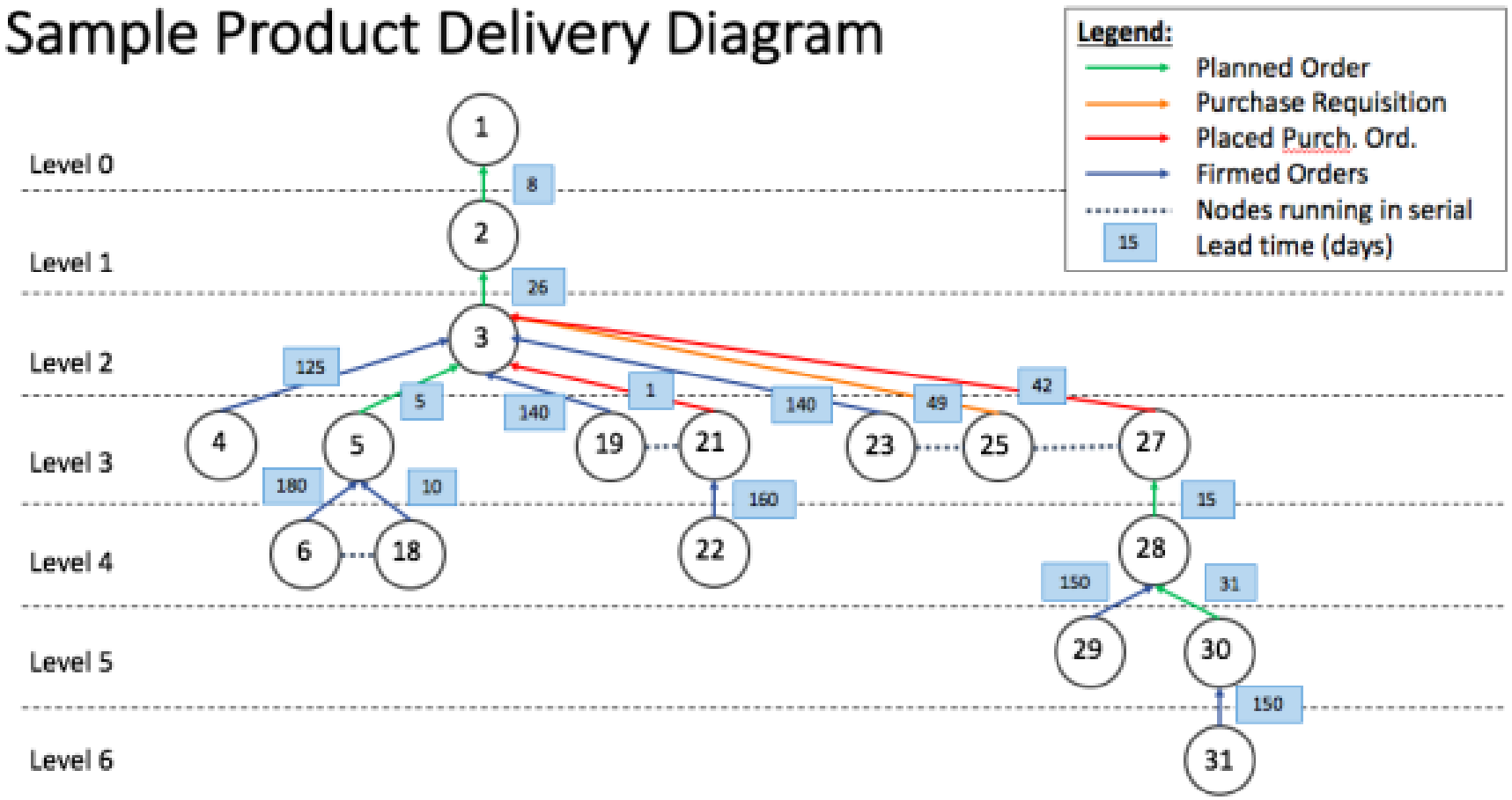
Linear Programming Formulation

- **Decision variables:**
  - x_i: reduction in activity i through crashing.
  - y_i: staring time of activity i.
  - y_{Finish}: project duration or starting time of the finish node
- **Objective:**
  - Minimize: $\sum_i c_i x_i$
- **Constraints:**
  - Maximum reduction time constraint: $x_i \leq b_i$
  - Precedence constraints, e.g.
    $y_6 \geq y_4 + 1 - x_4$; $y_6 \geq y_5 + 1 - x_5$; $y_{Finish} \geq y_9 + 1 - x_9$
  - Deadline constraint: $y_{Finish} \leq 7$
  - Non-negativity constraint: $x_i, y_i \geq 0$

Application: Manufacturing of Complex Machine



Application: Manufacturing of Complex Machine



Source: Chen Changhua, Goh Zhi Wen, Chuah Sim Teng, Ng Changwei and Yang Kan, "Optimizing Expediting Cost for Complex Product Delivery", Term Paper, 2017

Application: Manufacturing of Complex Machine

- **The real business problem:**
  - 6 levels.
  - 1307 nodes.

- **Decisions:**
  - Expediting decisions
  - Outsourcing decisions

- **Objective:**
  - Minimize expediting cost + outsourcing cost
  - Meeting the final product delivery time

- **Other considerations:**
  - Random lead time
  - Multiple suppliers to outsource from

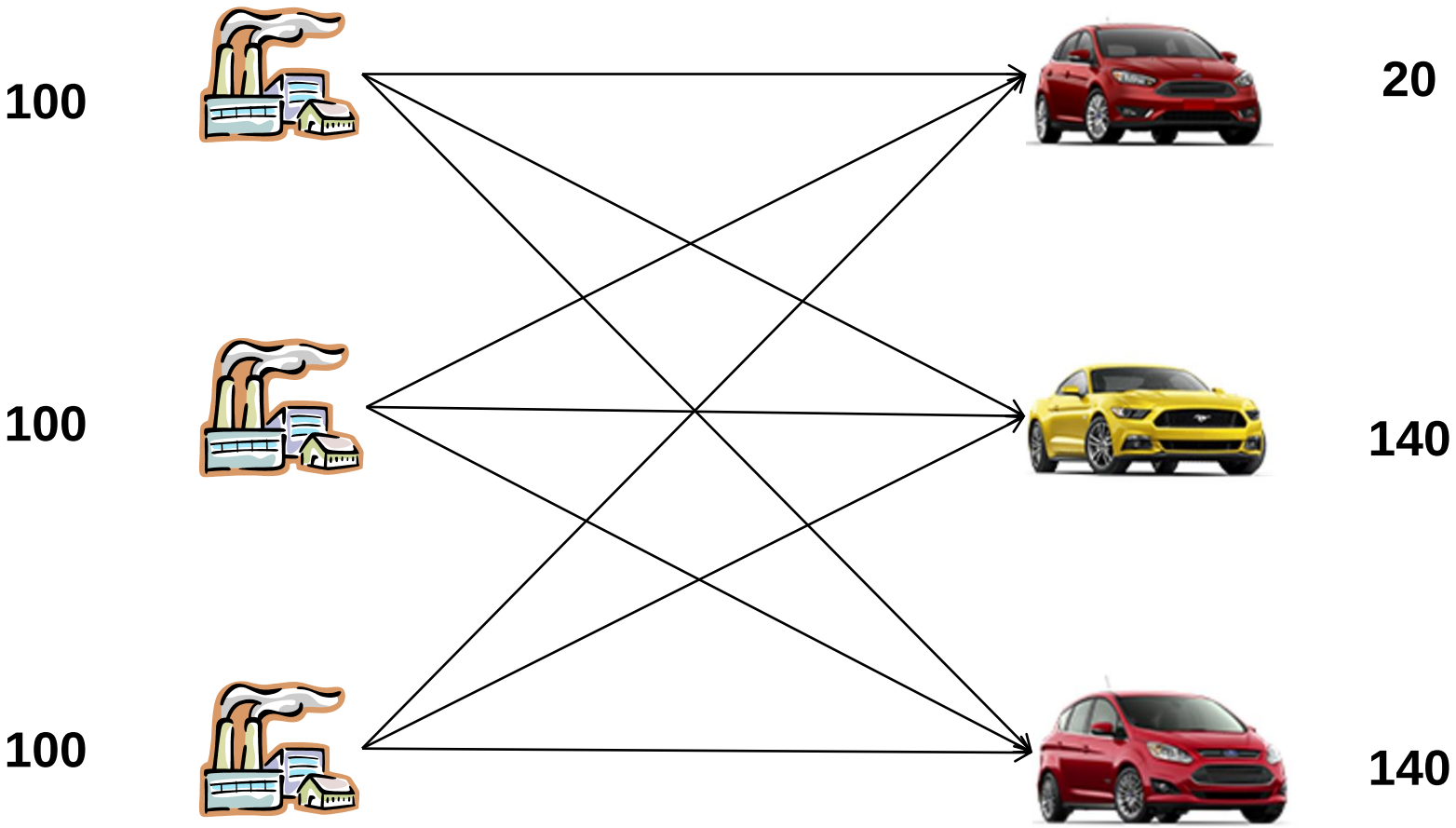
Process Flexibility: Dedicated System

100   20

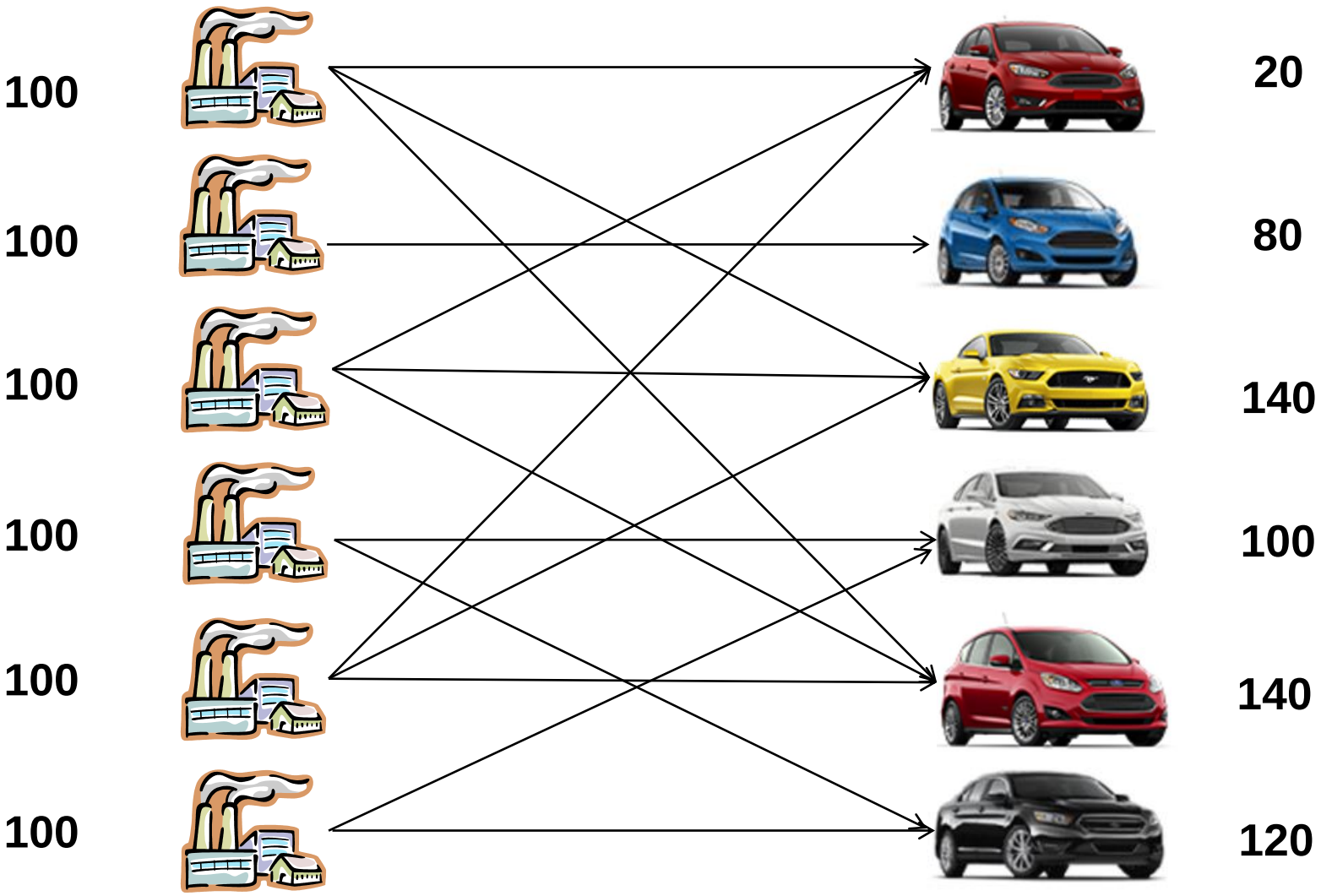
100   140

100   140

Full Flexible System



How to Compute the Maximum Sales?



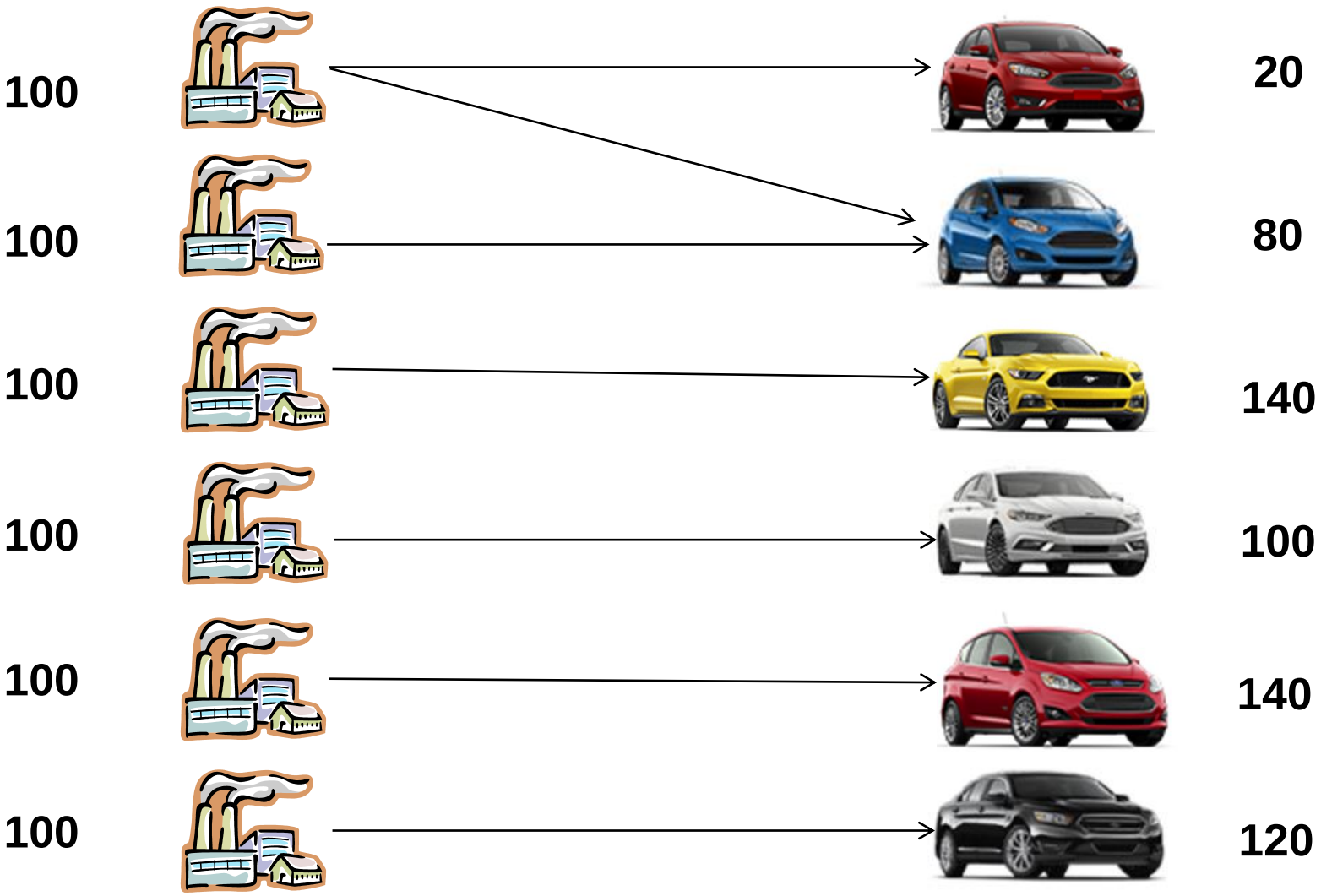
Linear Programming Formulation

- **Decision variables:**
  - x_{ij}: amount of product j produced at plant i.
- **Objective:**
  - Maximize: $\sum_{(i,j)\in A} x_{ij}$
- **Constraints:**
  - Capacity constraint at plant i : $\sum_{j,(i,j)\in A} x_{ij} \leq 100$
  - Demand constraint for product j : $\sum_{i,(i,j)\in A} x_{ij} \leq d_j$
  - Non-negativity constraint: $x_{ij} \geq 0$

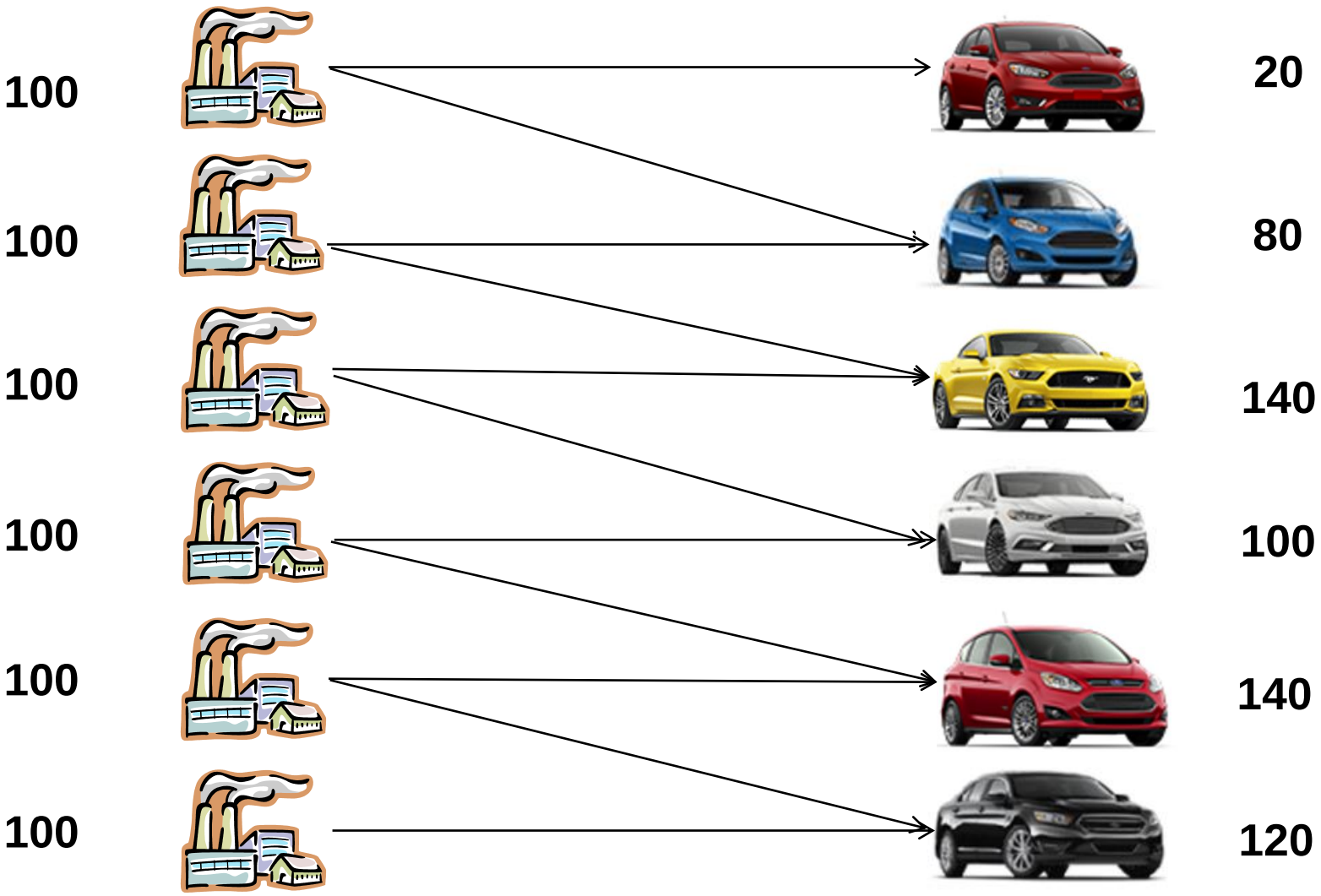
If demand is random…

- **Suppose demands of products are independent and demand for product *j*:** $D_j = max\{X_j, 0\}$
  - X_j **follows** $N(\mu_j, \sigma_j^2)$
- **Given a design (N, A), how do we evaluate its performance?**
  - **Draw M samples of** D_j
  - **Solve the network flow problem M times**
  - **Take the average of the objective values**

How to improve upon the dedicated system?



Open Chain



Long Chain

http://www.bizsimz.com/flexcap/