"Life has got all those twists and turns. You've got to hold on tight and off you go."
— Nicole Kidman

Network models

LINEAR PROGRAMMING AND OPTIMIZATION

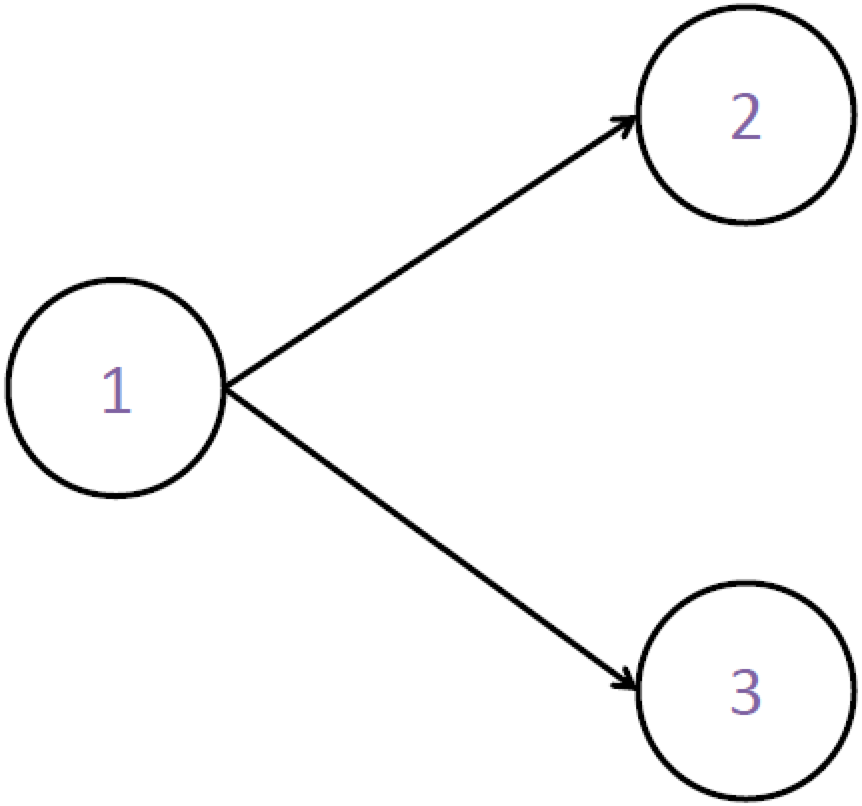
Network Models

Network Models – models that describe the pattern of flow in a connected system

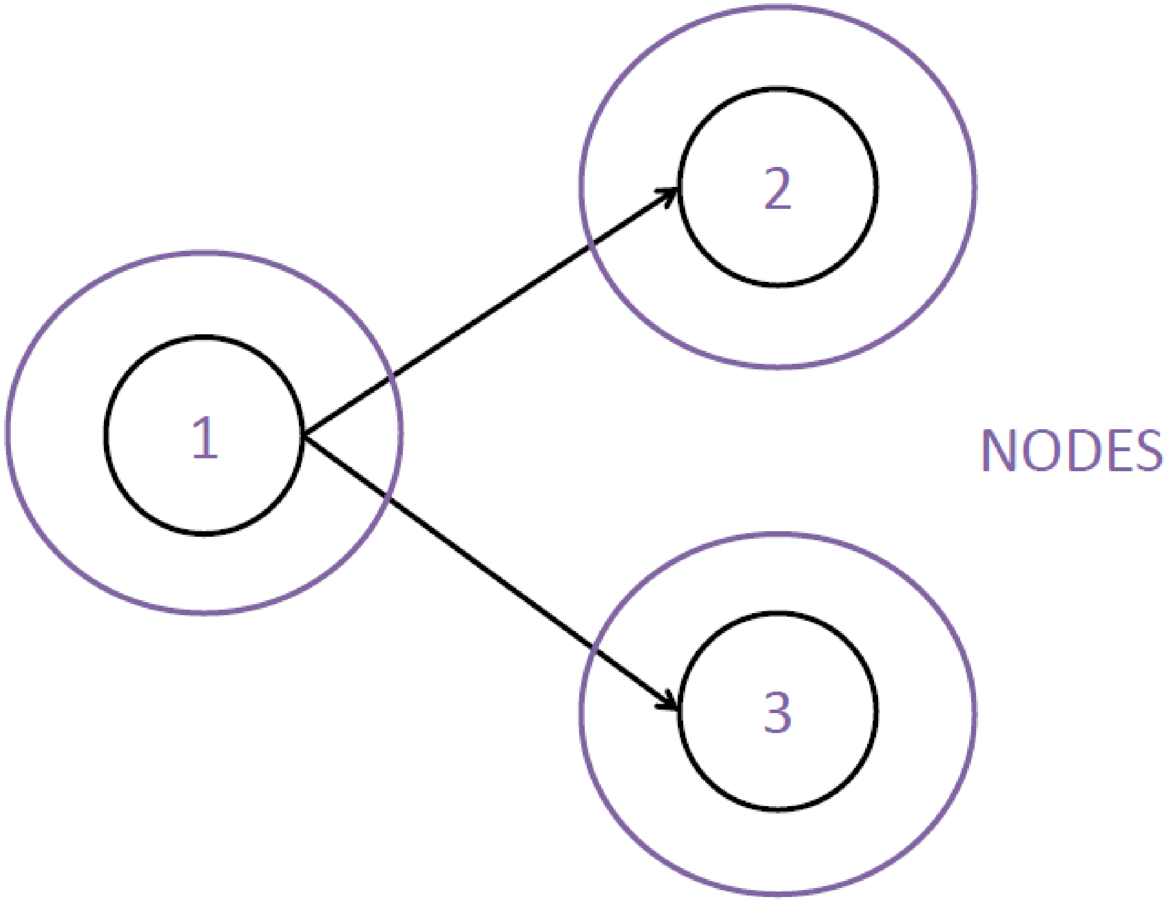
Nodes – system elements that are points in time and space

Arcs (edges) – paths of flow from one element/node to another

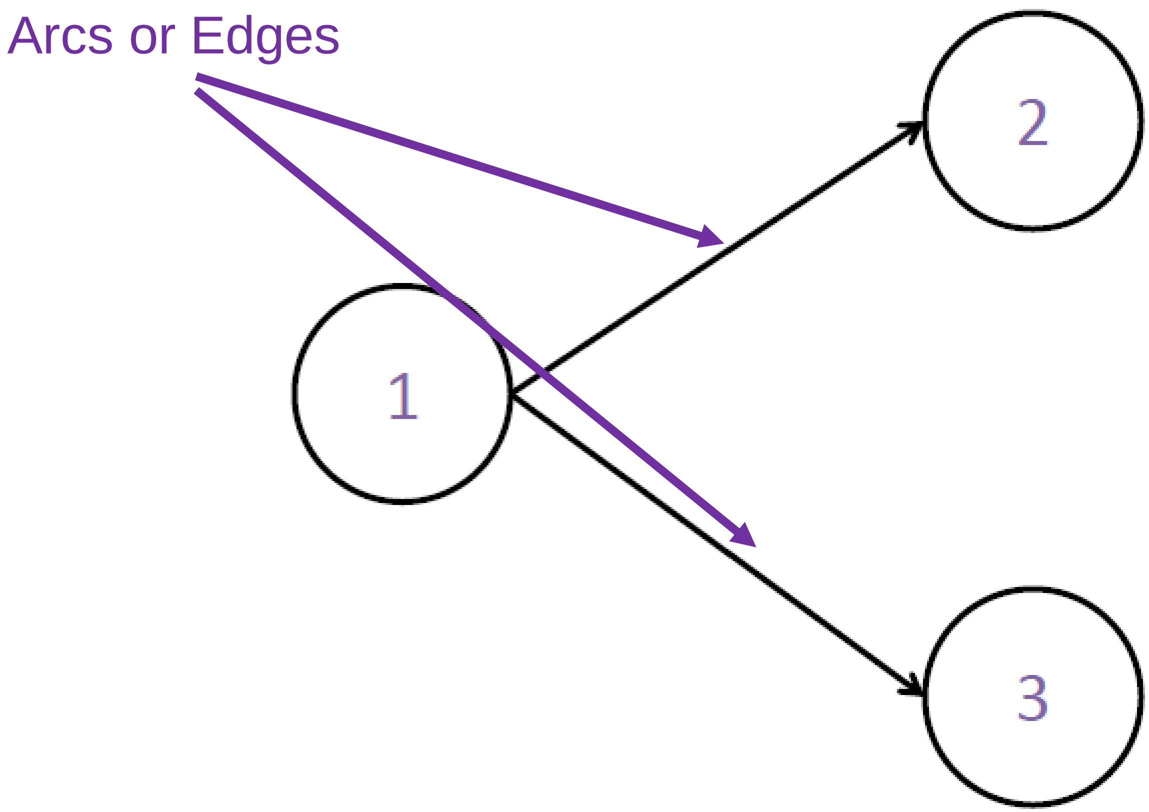
Example of a Network



Example of a Network



Example of a Network



Types of Models

- There are 2 common types of network models:
  1. Transportation Model (Supply chain model optimizes how goods are shipped from suppliers to customers)
  2. Assignment Model (special case of transportation model)

Transportation Model

You've seen this before!!

Transportation Model Example

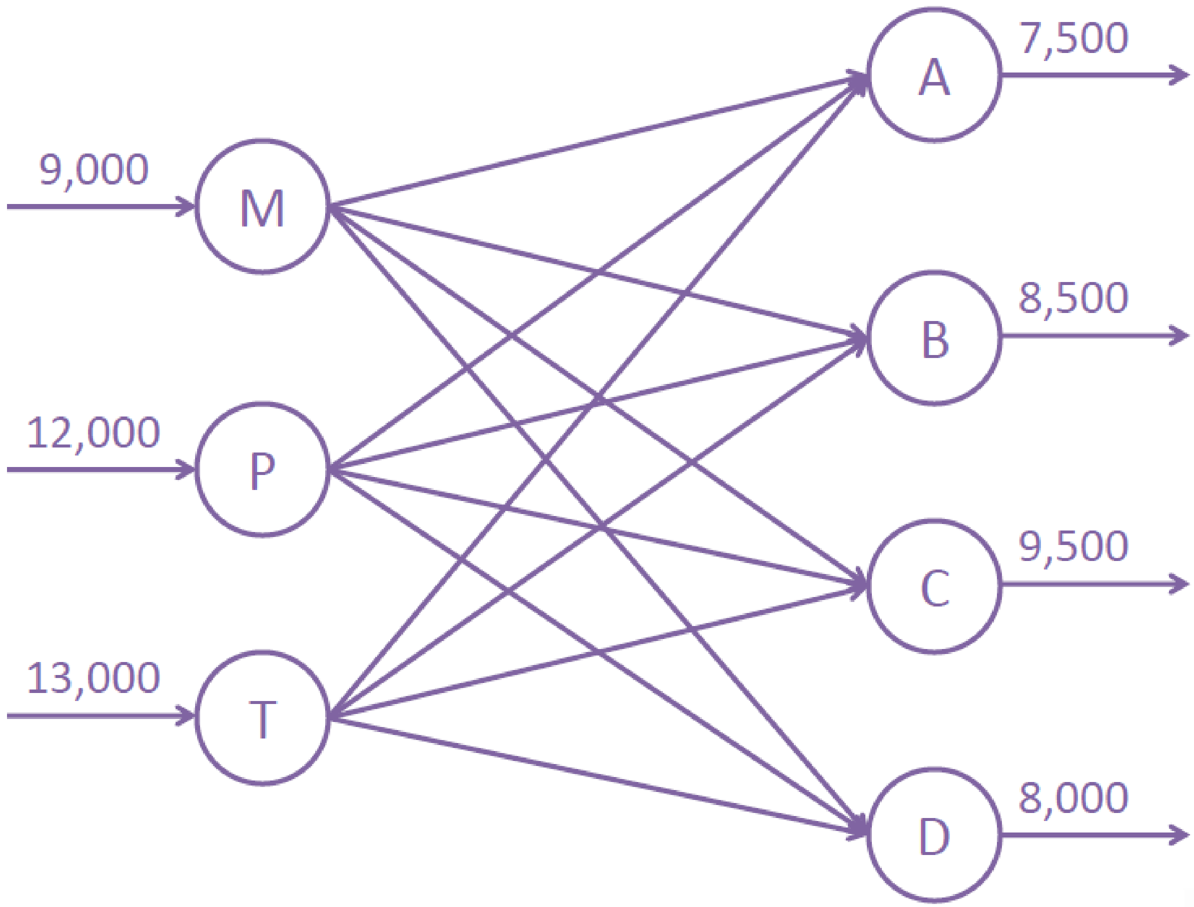
Bonner Electronics

- 3 Manufacturing Plants (different capacities)
- 4 Distribution Warehouses (different demands)
- Different costs between each shipping path combination
- (No fixed costs)
- Want to minimize cost

Transportation Model Example

Plant	Atlanta Warehouse	Boston Warehouse	Chicago Warehouse	Denver Warehouse	Capacity
Minneapolis	$0.60	$0.56	$0.22	$0.40	9,000
Pittsburgh	$0.36	$0.30	$0.28	$0.58	12,000
Tuscon	$0.65	$0.68	$0.55	$0.42	13,000
Demand	**7,500**	**8,500**	**9,500**	**8,000**	

Transportation Model Example



Optimization set-up

12 Decision variables: $x_{i,j}$ for i =1..3 (plants) and j = 1..4 (warehouses)

Objective function: Minimize cost = $\sum_i \sum_j C_{i,j} x_{i,j}$

Constraints (12 constraints):

<u>Capacity constraints:</u> For every plant (i..3 of these):
$\sum_j x_{i,j} \leq Capacity_i$

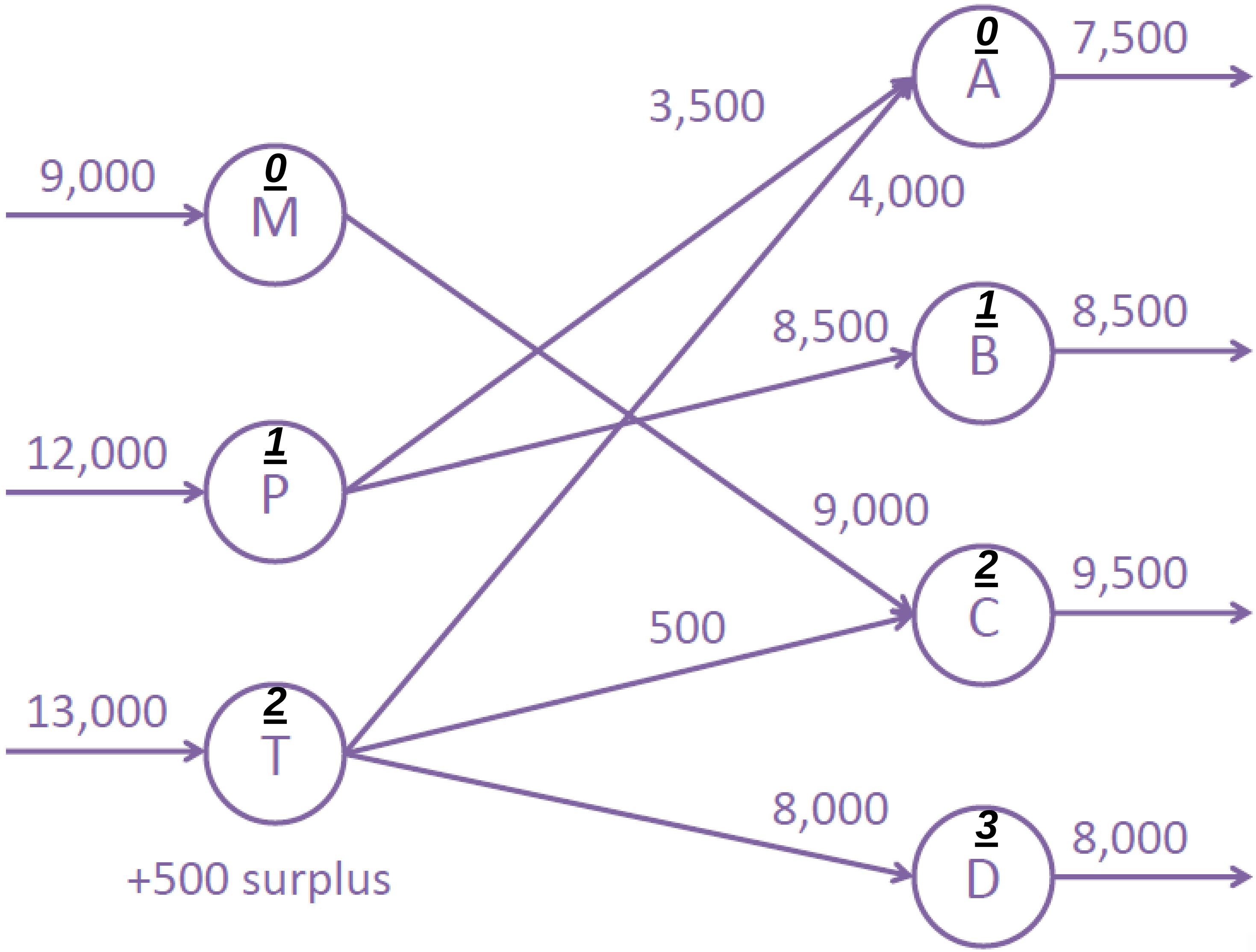
<u>Demand constraints:</u> For every warehouse (j…there are 4 of these): $\sum_i x_{i,j} \geq Demand_j$

```
ShipCost =   [[0.60, 0.56, 0.22, 0.40],
              [0.36, 0.30, 0.28, 0.58],
              [0.65, 0.68, 0.55, 0.42]]
Capacity = [9000, 12000, 13000]
Demand = [7500, 8500, 9500, 8000]
plants = len(Capacity) #should be 3
warehouses = len(Demand) #should be 4
m=Model()
x = {} #Amount to ship (continuous)
for i in range(plants):
    for j in range(warehouses):
        x[(i,j)] = m.addVar(vtype=GRB.CONTINUOUS, lb=0, name='x%d,%d' % (i,j))
m.setObjective( quicksum(quicksum(ShipCost[i][j]*x[(i,j)]
                for i in range(plants)) for j in range(warehouses)), GRB.MINIMIZE)
for i in range(plants):
    m.addConstr(quicksum(x[(i,j)] for j in range(warehouses)) <= Capacity[i])
for j in range(warehouses):
    m.addConstr(quicksum( x[(i,j)] for i in range(plants)) >= Demand[j] )
m.optimize()
for v in m.getVars():
    print('%s: %g' % (v.varName, v.x))
print('Obj: %g' % m.objVal)
m.printAttr('Pi')
m.printAttr('rc')
```

Output

Solved in 8 iterations and 0.02 seconds
Optimal objective 1.202500000e+04
x0,0: 0
x0,1: 0
x0,2: 9000
x0,3: 0
x1,0: 3500
x1,1: 8500
x1,2: 0
x1,3: 0
x2,0: 4000
x2,1: 0
x2,2: 500
x2,3: 8000
Obj: 12025

Variable	rc
x0,0	0.28
x0,1	0.3
x0,3	0.31
x1,2	0.02
x1,3	0.45
x2,1	0.09


9,000
0
M
12,000
1
P
13,000
2
T
+500 surplus
3,500
4,000
8,500
9,000
500
8,000
0
A
7,500
1
B
8,500
2
C
9,500
3
D
8,000

Assignment Model

Assignment Model Example

Buchanan Swim Club

- Need to assign 4 swimmers to the relay (4 strokes)
- Have times for each person’s best of each stroke
- Want to minimize relay time

Assignment Model Example

Person	Butterfly Stroke (sec.)	Breast Stroke (sec.)	Back Stroke (sec.)	Free Style (sec.)
Todd	38	75	44	27
Betsy	34	76	43	25
Lee	41	71	41	26
Carly	33	80	45	30

Optimization set-up

There are 16 decision variables (all binary): $x_{i,j}$ (i is for person and j is for event)

Objective function: Minimize time: $\sum_i \sum_j x_{i,j} T_{i,j}$

Constraints:

Person Constraints: For every i: $\sum_j x_{i,j} = 1$

Event Constraints: For every j: $\sum_i x_{i,j} = 1$

Gurobi

```
m=Model()
Time = [[38, 75, 44, 27],
        [34, 76, 43, 25],
        [41, 71, 41, 26],
        [33, 80, 45, 30]]
for i in range(4):
    for j in range(4):
        x[(i,j)]=m.addVar(vtype=GRB.BINARY, name='swim%d,%d' % (i,j))
m.setObjective(quicksum(quicksum(Time[i][j]*x[(i,j)] for i in range(4)) for j in
range(4)) , GRB.MINIMIZE)
for i in range(4):
    m.addConstr(quicksum(x[(i,j)] for j in range(4)) ==1)
for j in range(4):
    m.addConstr(quicksum(x[(i,j)] for i in range(4)) ==1)

m.optimize()
for v in m.getVars():
    print('%s: %g' % (v.varName, v.x))
print('Obj: %g' % m.objVal)
```

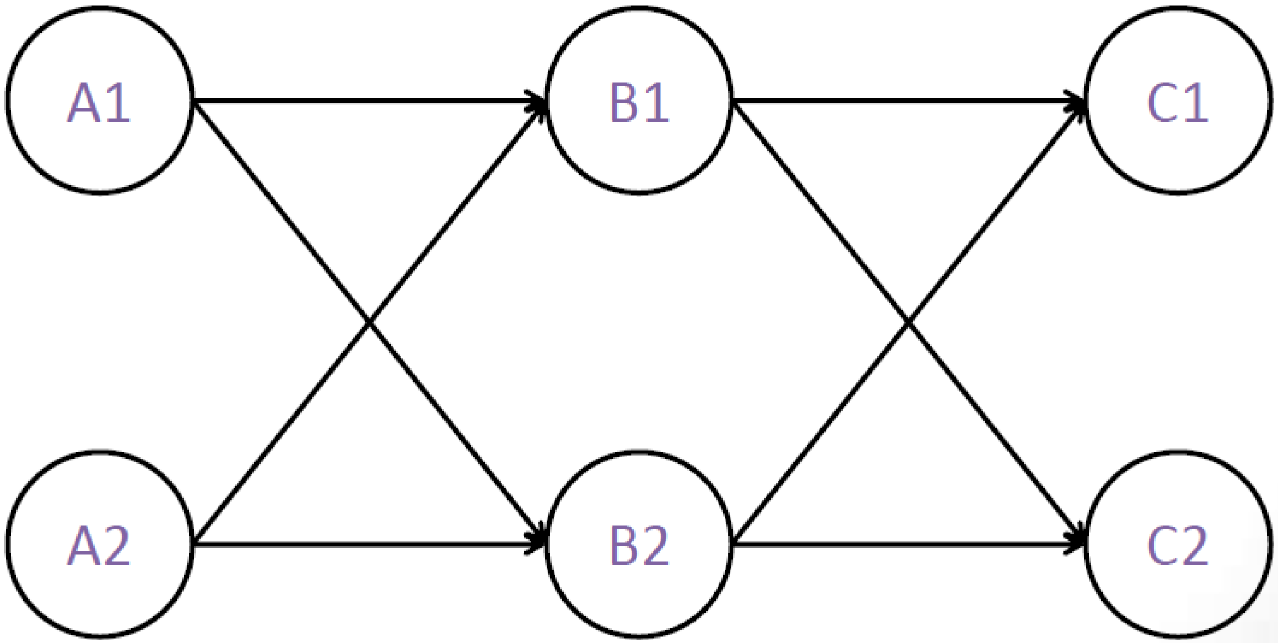
Output

Optimal solution found (tolerance 1.00e-04)
Best objective 1.730000000000e+02, best bound 1.730000000000e+02, gap 0.0000%
swim0,0: -0
swim0,1: -0
swim0,2: 1
swim0,3: 0
swim1,0: 0
swim1,1: -0
swim1,2: -0
swim1,3: 1
swim2,0: -0
swim2,1: 1
swim2,2: 0
swim2,3: -0
swim3,0: 1
swim3,1: -0
swim3,2: 0
swim3,3: -0
Obj: 173

Transshipment Model

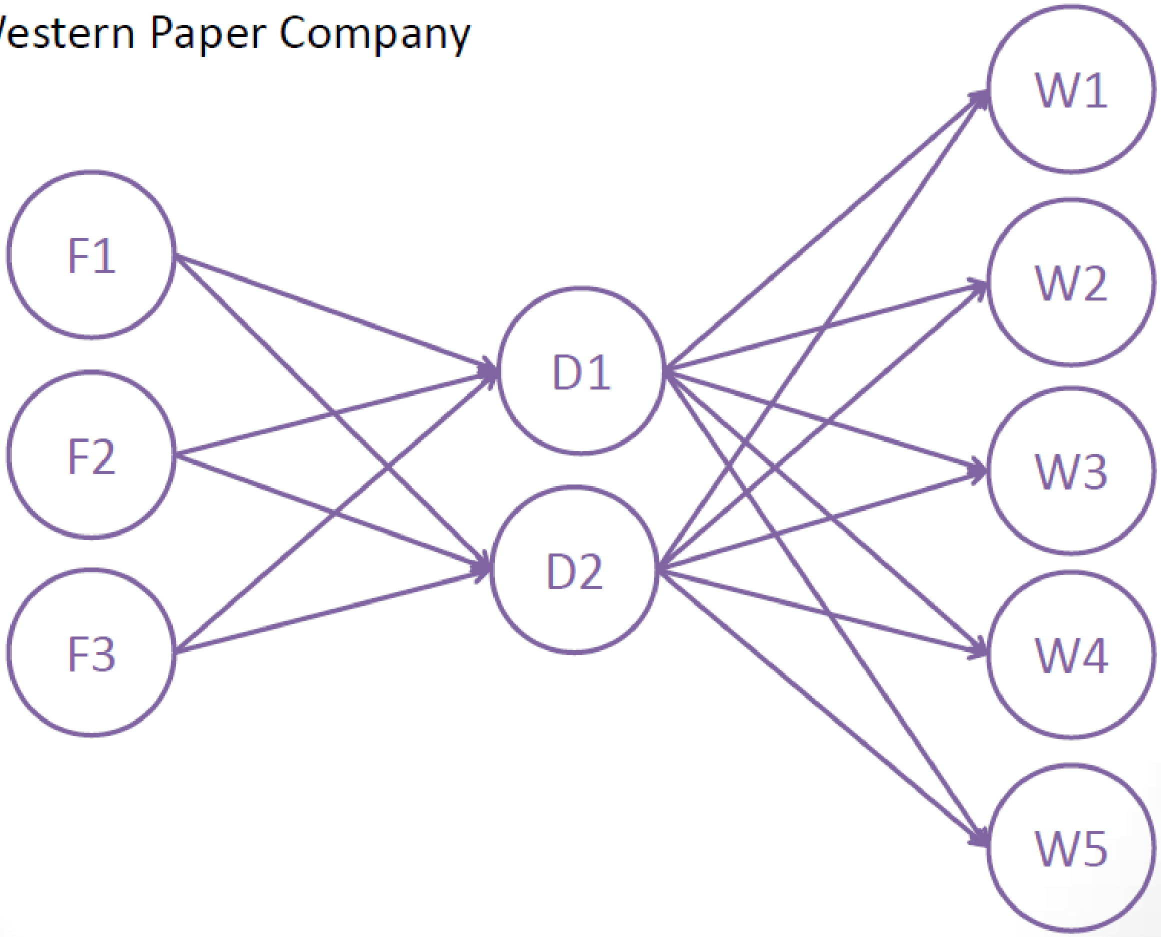
Transshipment Model

- The transshipment model is a more complex version of the transportation model.
- Transshipment models have more than one stage in the system.



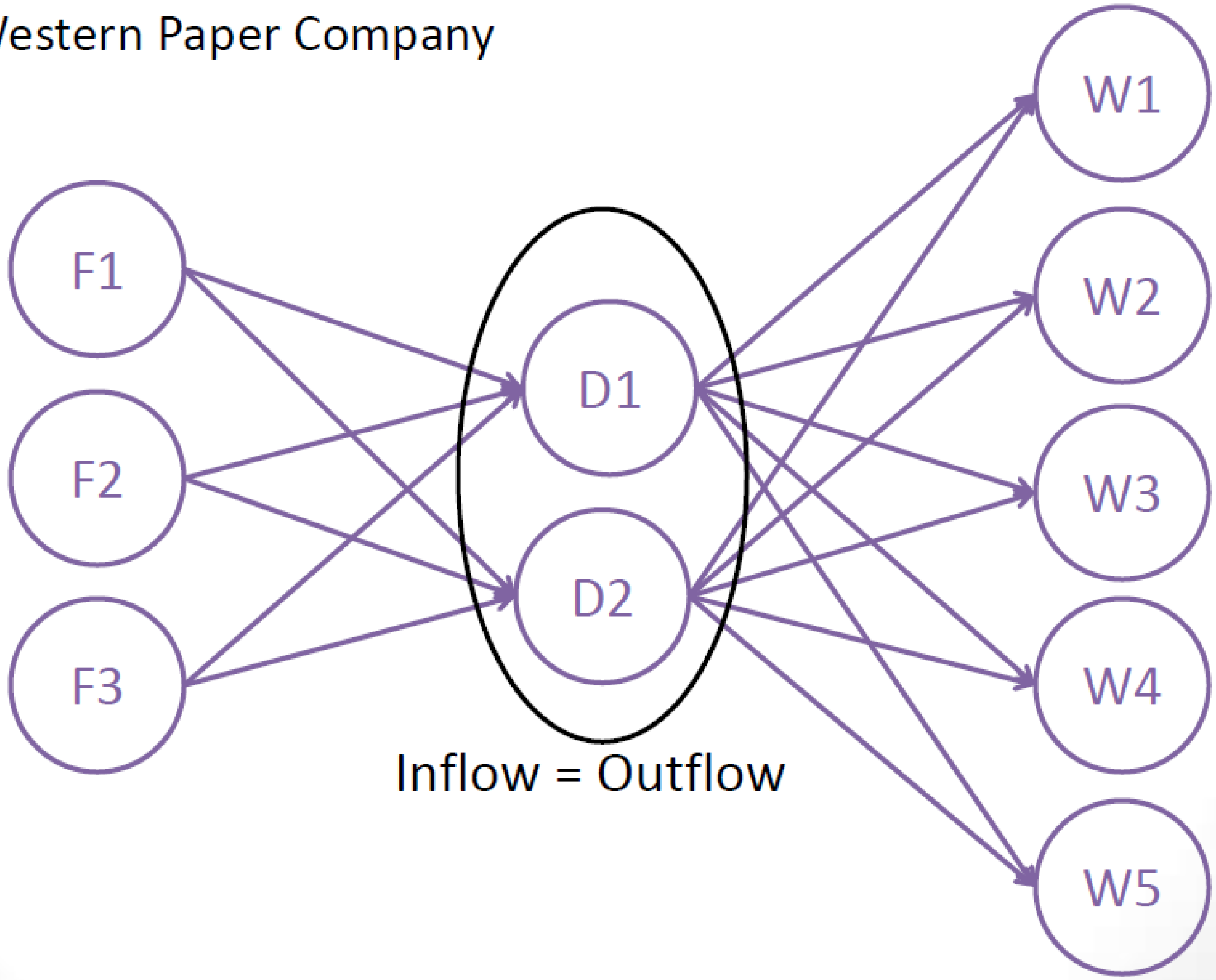
Transshipment Model Example

- Western Paper Company



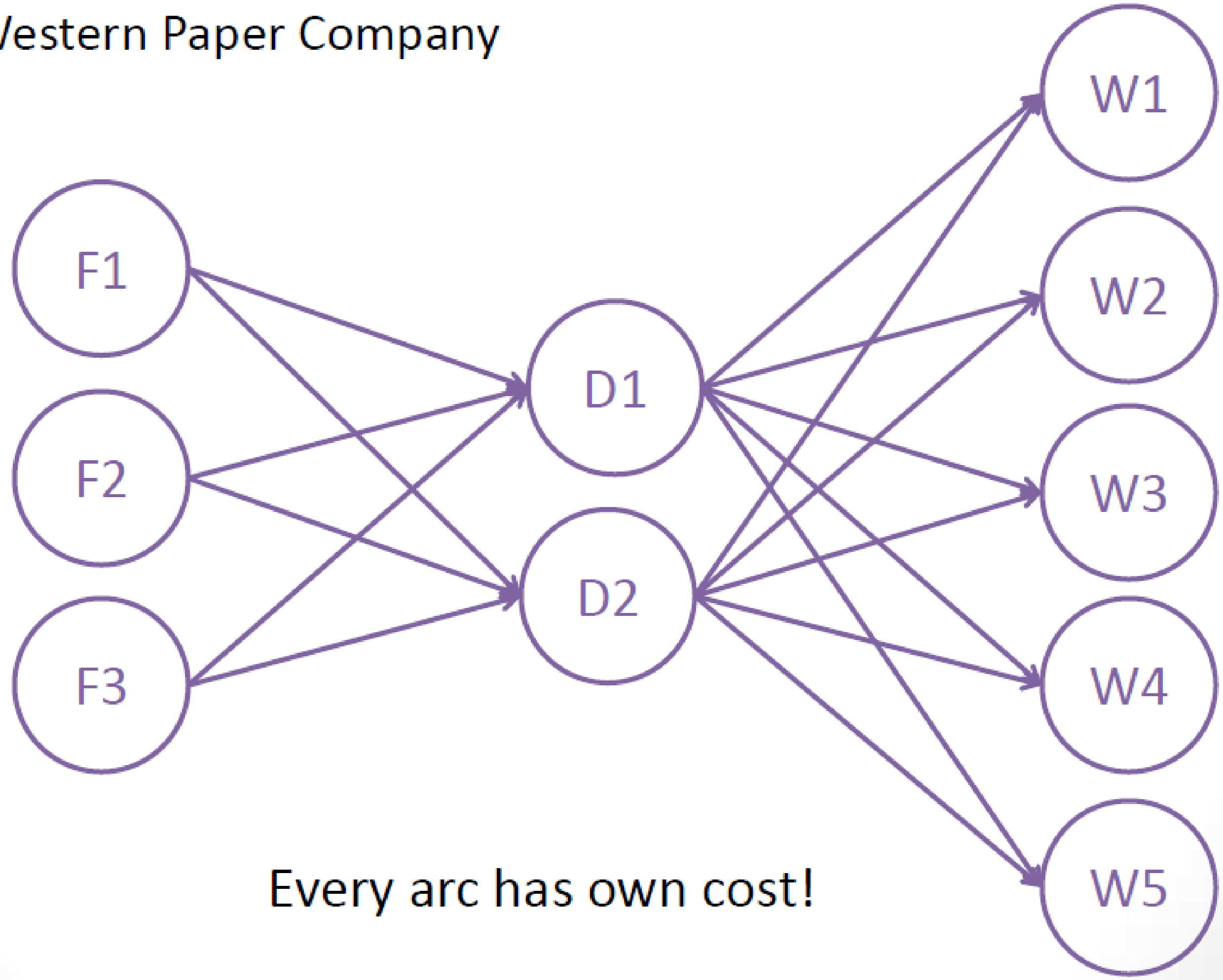
Transshipment Model Example

- Western Paper Company



Transshipment Model Example

- Western Paper Company



Transportation Cost

TRANSPORTATION COST

	Dist1	Dist2	Cap
Factory1	1.28	1.36	2500
Factory2	1.33	1.38	2500
Factory3	1.68	1.55	2500

TRANSPORTATION COST

	W1	W2	W3	W4	W5
Dist1	0.6	0.42	0.32	0.44	0.68
Dist2	0.57	0.3	0.4	0.38	0.72
Dem:	1200	1300	1400	1500	1600

Optimization set-up

There are 14 decision variables (6 deciding how much to ship from each factory to distribution center…and 8 from each distribution center to warehouse): $FtoD_{i,j}$ and $DtoW_{j,k}$

Objective function: Minimize Cost:
$\sum_i \sum_j C1_{i,j} FtoD_{i,j} + \sum_j \sum_k C2_{j,k} DtoW_{j,k}$

Constraints:

Capacity: For every i (Factory): $\sum_j FtoD_{i,j} \leq Capacity_i$

Demand: For every k (Warehouse): $\sum_j DtoW_{j,k} \geq Demand_k$

Inflow=Outflow: For every j (Distribution Center): $\sum_i FtoD_{i,j} = \sum_k DtoW_{j,k}$

```
m=Model()
C1 = [[1.28, 1.36],
        [1.33, 1.38],
        [1.68, 1.55]]
C2 = [[0.60, 0.42, 0.32, 0.44, 0.68],
        [0.57, 0.30, 0.4, 0.38, 0.72]]
Demand = [1200, 1300, 1400, 1500, 1600]
FtoD={}
DtoW={}
for i in range(3):
    for j in range(2):
        FtoD[(i,j)] = m.addVar(vtype=GRB.CONTINUOUS, lb=0, name='F2D%d,%d' % (i,j))
for j in range(2):
    for k in range(5):
        DtoW[(i,j)]=m.addVar(vtype=GRB.CONTINUOUS, lb=0,  name='D2W%d,%d' % (i,j))
m.setObjective(quicksum(quicksum(C1[i][j]*FtoD[(i,j)] for i in range(3)) for j in range(2)) +
            quicksum(quicksum(C2[j][k]*DtoW[(j,k)] for j in range(2)) for k in range(5)), GRB.MINIMIZE)
for i in range(3):
    m.addConstr(quicksum(FtoD[(i,j)] for j in range(2)) <= 2500)
for k in range(5):
    m.addConstr(quicksum(DtoW[(j,k)] for j in range(2)) >= Demand[k])
for j in range(2):
    m.addConstr(quicksum(DtoW[(j,k)] for k in range(5)) == quicksum(FtoD[(i,j)] for i in range(3)))
m.optimize()
for v in m.getVars():
    print('%s: %g' % (v.varName, v.x))
print('Obj: %g' % m.objVal)
```

Output

Solved in 11 iterations and 0.02 seconds
Optimal objective 1.288100000e+04
F2D0,0: 2500
F2D0,1: 0
F2D1,0: 1700
F2D1,1: 800
F2D2,0: 0
F2D2,1: 2000
D2W0,0: 1200
D2W0,1: 0
D2W0,2: 1400
D2W0,3: 0
D2W0,4: 1600
D2W1,0: 0
D2W1,1: 1300
D2W1,2: 0
D2W1,3: 1500
D2W1,4: 0
Obj: 12881

Transshipment Model Example