Network models

Linear Programming and Optimization

Recall Types of Linear Programming

1. Allocation Models
2. Covering Models
3. Blending Models
4. Network Models

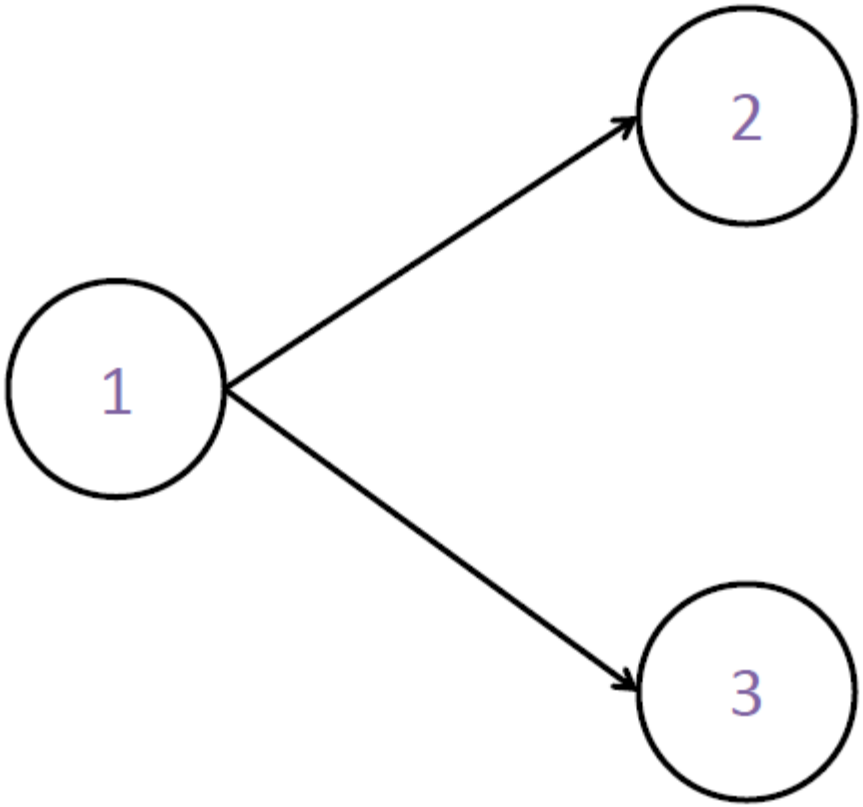
Recall Types of Linear Programming

1. Allocation Models
2. Covering Models
3. Blending Models
4. Network Models

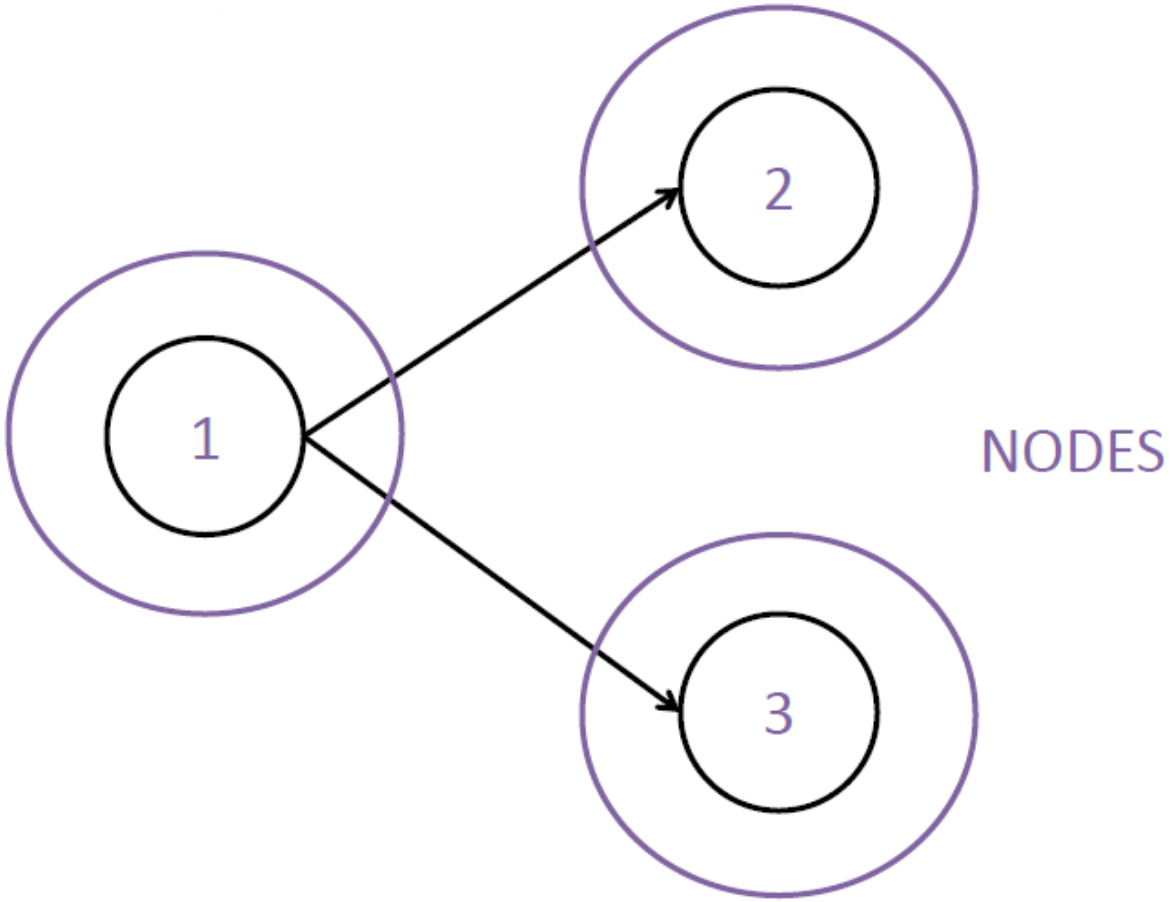
Network Models

- **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

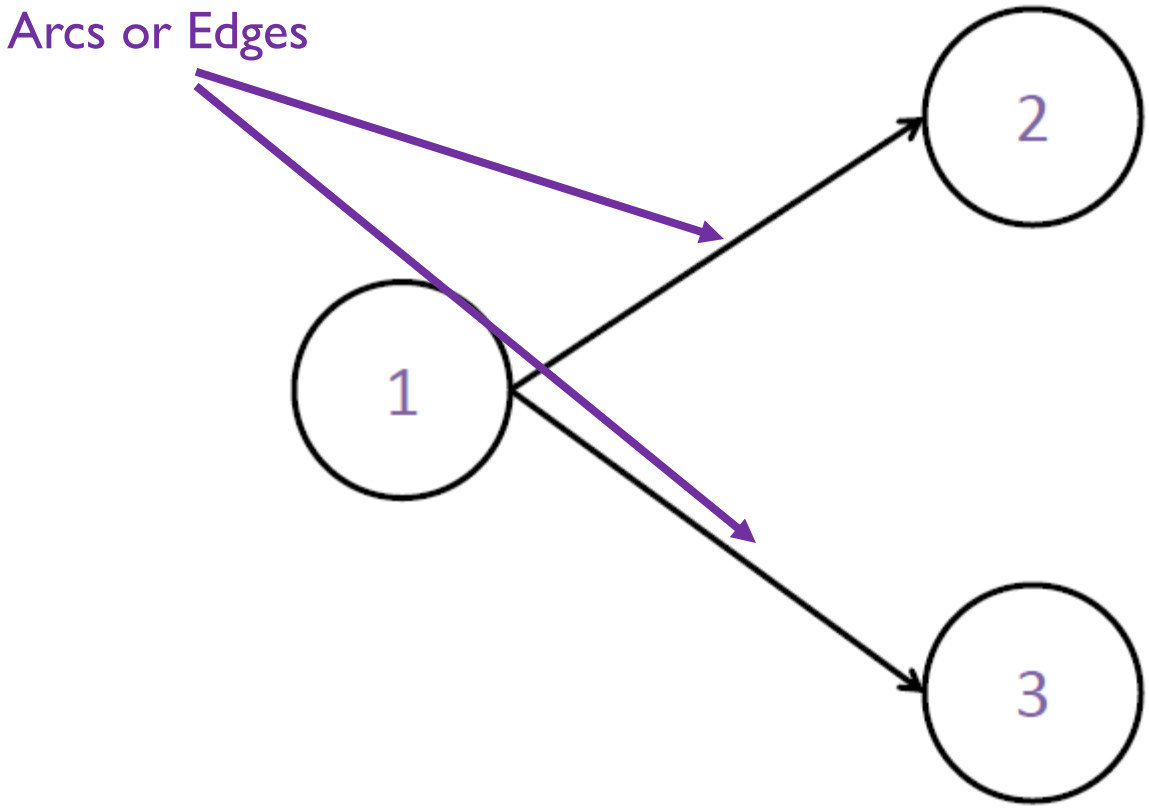
Example of a Network



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!!

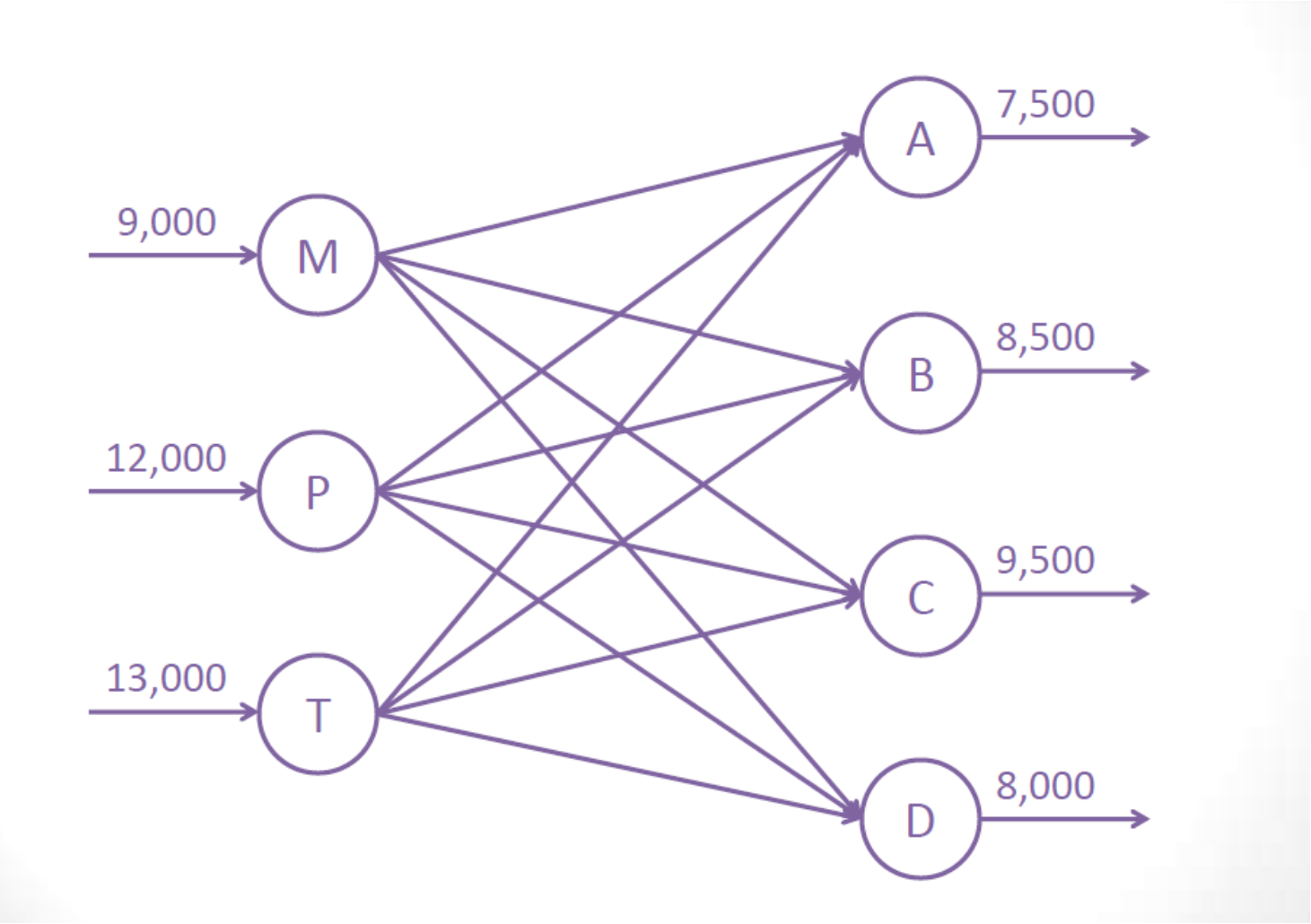
Transportation Model Example

- 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



SAS Code

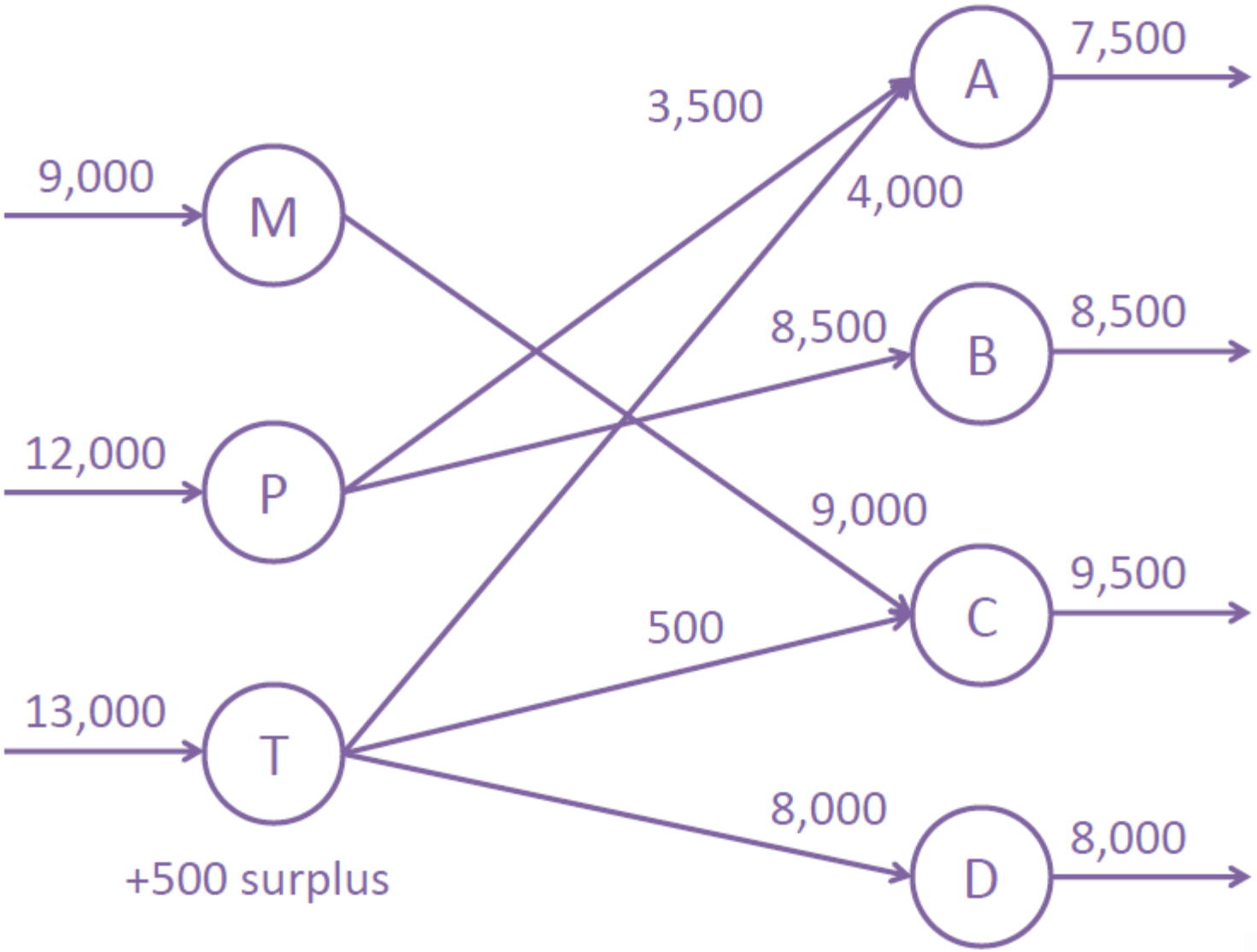
```
proc optmodel;
set Warehouse = /Atlanta Boston Chicago Denver/;
set Plant = /Minneapolis Pittsburgh Tuscon/;
number ShipCost{Plant,Warehouse} = [0.60 0.56 0.22 0.40
                                    0.36 0.30 0.28 0.58
                                    0.65 0.68 0.55 0.42];
number Capacity{Plant} = [9000 12000 13000];
number Demand{Warehouse} = [7500 8500 9500 8000];
var x{Plant,Warehouse}>=0;
min Cost = sum{i in Plant}(sum{j in Warehouse}
(ShipCost[i,j]*x[i,j]));
con Cap {i in Plant}: sum{j in Warehouse} x[i,j] <= Capacity[i];
con Dem {j in Warehouse}: sum{i in Plant} x[i,j] >= Demand[j];
solve;
print x;
print x.rc;
print Cap.dual Dem.dual;
quit;
```

Output

x				
	Atlanta	Boston	Chicago	Denver
Minneapolis	0	0	9000	0
Pittsburgh	3500	8500	0	0
Tuscon	4000	0	500	8000

x.RC				
	Atlanta	Boston	Chicago	Denver
Minneapolis	0.28	0.30	0.00	0.31
Pittsburgh	0.00	0.00	0.02	0.45
Tuscon	0.00	0.09	0.00	0.00

[1]	Cap.DUAL	Dem.DUAL
Atlanta		0.65
Boston		0.59
Chicago		0.55
Denver		0.42
Minneapolis	-0.33	
Pittsburgh	-0.29	
Tuscon	0.00	


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

Rcode

```
library(lpSolve)
cost=matrix(c(.6,.36,.65,.56,.3,.68,.22,.28,.55,.4,.58,.42), ncol=4, byrow=F)
capacity=c(9000,12000,13000)
demand=c(7500,8500,9500,8000)
f.obj=as.numeric(cost)
f.con=matrix(0,nrow=7,ncol=12)
temp=c(1,4,7,10)
for (i in 1:3)
{f.con[i,(i-1+temp)]=1}
temp2=c(1,2,3)
for (i in 1:4)
{f.con[i+3,(3*(i-1)+temp2)]=1}
f.rhs=c(capacity,demand)
f.dir=c(rep('<=',3),rep('>=',4))
lp.model=lp (direction="min", f.obj, f.con, f.dir, f.rhs,compute.sens=1)
matrix(lp.model$solution,nrow=3,byrow=F)
matrix(lp.model$duals[8:19],nrow=3,byrow=F)
```

Output

```
> matrix(lp.model$solution,nrow=3,byrow=F)
     [,1] [,2] [,3] [,4]
[1,]    0    0 9000    0
[2,] 3500 8500    0    0
[3,] 4000    0  500 8000
> matrix(lp.model$duals[8:19],nrow=3,byrow=F)
     [,1] [,2] [,3] [,4]
[1,] 0.28 0.30 0.00 0.31
[2,] 0.00 0.00 0.02 0.45
[3,] 0.00 0.09 0.00 0.00
```

Gurobi

```
library(gurobi)
library(prioritizr)
cost=matrix(c(.6,.36,.65,.56,.3,.68,.22,.28,.55,.4,.58,.42), ncol=4, byrow=F)
capacity=c(9000,12000,13000)
demand=c(7500,8500,9500,8000)
model=list()
model$obj=as.numeric(cost)
model$A=matrix(0,nrow=7,ncol=12)
temp=c(1,4,7,10)
for (i in 1:3)
{model$A[i,(i-1+temp)]=1}
temp2=c(1,2,3)
for (i in 1:4)
{model$A[i+3,(3*(i-1)+temp2)]=1}
model$rhs=c(capacity,demand)
model$sense=c(rep('<=',3),rep('>=',4))
model$modelsense="min"
result=gurobi(model,list())
matrix(result$x,nrow=3,byrow=F)
matrix(result$rc,nrow=3,byrow=F)
result$pi
```

Output

```
> matrix(result$x,nrow=3,byrow=F)
     [,1] [,2] [,3] [,4]
[1,]    0    0 9000    0
[2,] 3500 8500    0    0
[3,] 4000    0  500 8000
> matrix(result$rc,nrow=3,byrow=F)
     [,1] [,2] [,3] [,4]
[1,] 0.28 0.30 0.00 0.31
[2,] 0.00 0.00 0.02 0.45
[3,] 0.00 0.09 0.00 0.00
> result$pi
[1] -0.33 -0.29  0.00  0.65  0.59  0.55  0.42
```

Sensitivity Analysis

- Shadow Prices and Reduced Costs are very informational in a transportation network model:
  - Shadow Price for Demand – marginal cost to ship one more product to that location
  - Shadow Price for Capacity – marginal cost saving to have one more product capacity at that location
  - Reduced Cost – how much to reduce cost to make that path feasible to ship along

Assignment Model

Assignment Model Example

- Buchanan Swim Club
  - Ned 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

SAS Code

```
proc optmodel;
set Stroke = /Butterfly Breaststroke Backstroke Freestyle/;
set Swimmer = /Todd Betsy Lee Carly/;
number Time{Swimmer,Stroke} = [38 75 44 27
                               34 76 43 25
                               41 71 41 26
                               33 80 45 30];
var x{Swimmer,Stroke}>=0 binary;
min RelayTime = sum{i in Swimmer}(sum{j in Stroke}
(Time[i,j]*x[i,j]));
con MaxSwimmer {i in Swimmer}: sum{j in Stroke} x[i,j] = 1;
con MaxStroke {j in Stroke}: sum{i in Swimmer} x[i,j] = 1;
solve;
print x;
quit;
```

Output

	x			
	Backstroke	Breaststroke	Butterfly	Freestyle
Betsy	0	0	0	1
Carly	0	0	1	0
Lee	0	1	0	0
Todd	1	0	0	0

Rcode

```
time=matrix(c(38,34,41,33,75,76,71,80,44,43,41,45,27,25,26,30),nrow=4,b
yrow=F)
f.obj=as.numeric(time)
f.con=matrix(0,nrow=8,ncol=16)
temp=c(1,5,9,13)
for (i in 1:4)
{f.con[i,(i-1+temp)]=1}
temp2=c(1,2,3,4)
for (i in 1:4)
{f.con[i+4,(4*(i-1)+temp2)]=1}
f.rhs=rep(1,8)
f.dir=rep('=',8)
lp.model=lp (direction="min", f.obj, f.con, f.dir, f.rhs)
matrix(lp.model$solution,nrow=4,byrow=F)
lp.model$objval
```

Output

```
> matrix(lp.model$solution,nrow=4,byrow=F)
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    0
[2,]    0    0    0    1
[3,]    0    1    0    0
[4,]    1    0    0    0
> lp.model$objval
[1] 173
```

Gurobi

```
time=matrix(c(38,34,41,33,75,76,71,80,44,43,41,45,27,25,26,30),nrow=4,byrow=F)
model=list()
model$obj=as.numeric(time)
model$A=matrix(0,nrow=8,ncol=16)
temp=c(1,5,9,13)
for (i in 1:4)
{model$A[i,(i-1+temp)]=1}
temp2=c(1,2,3,4)
for (i in 1:4)
{model$A[i+4,(4*(i-1)+temp2)]=1}
model$rhs=rep(1,8)
model$sense=rep('=',8)
model$modelsense="min"
result=gurobi(model,list())
matrix(result$x,nrow=4,byrow=F)
result$objval
```

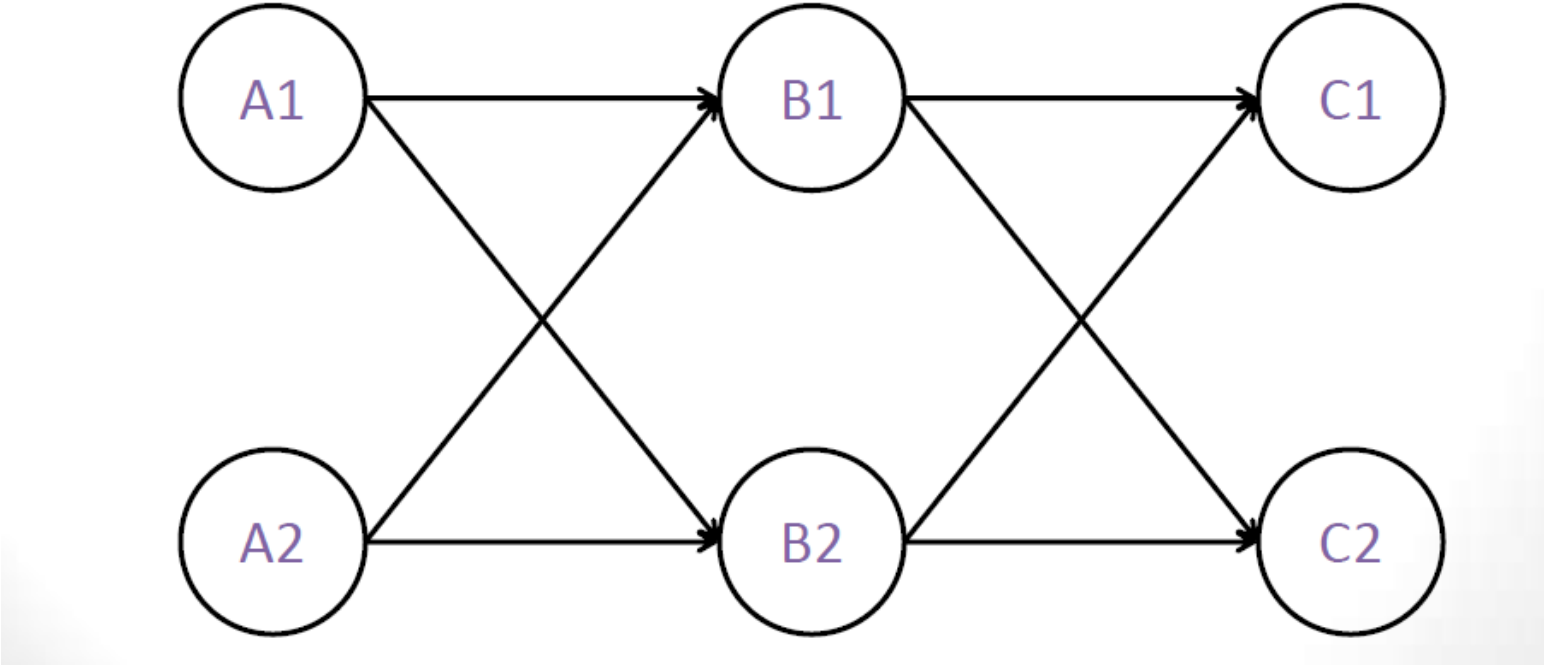
Output

```
> matrix(result$x,nrow=4,byrow=F)
     [,1] [,2] [,3] [,4]
[1,]    0    0    1    0
[2,]    0    0    0    1
[3,]    0    1    0    0
[4,]    1    0    0    0
> result$objval
[1] 173
```

Transshipment Model

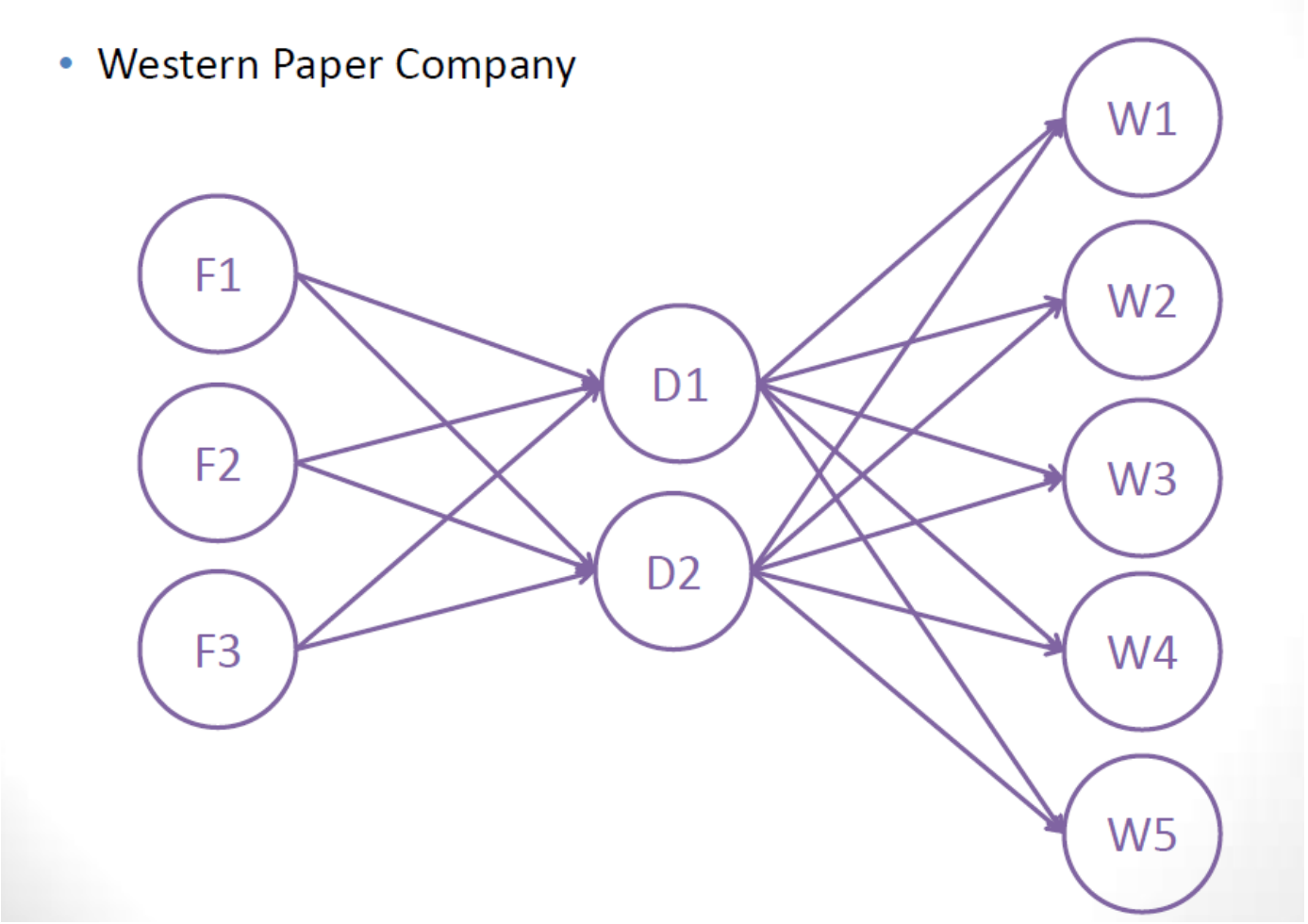
Transshipment Model

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



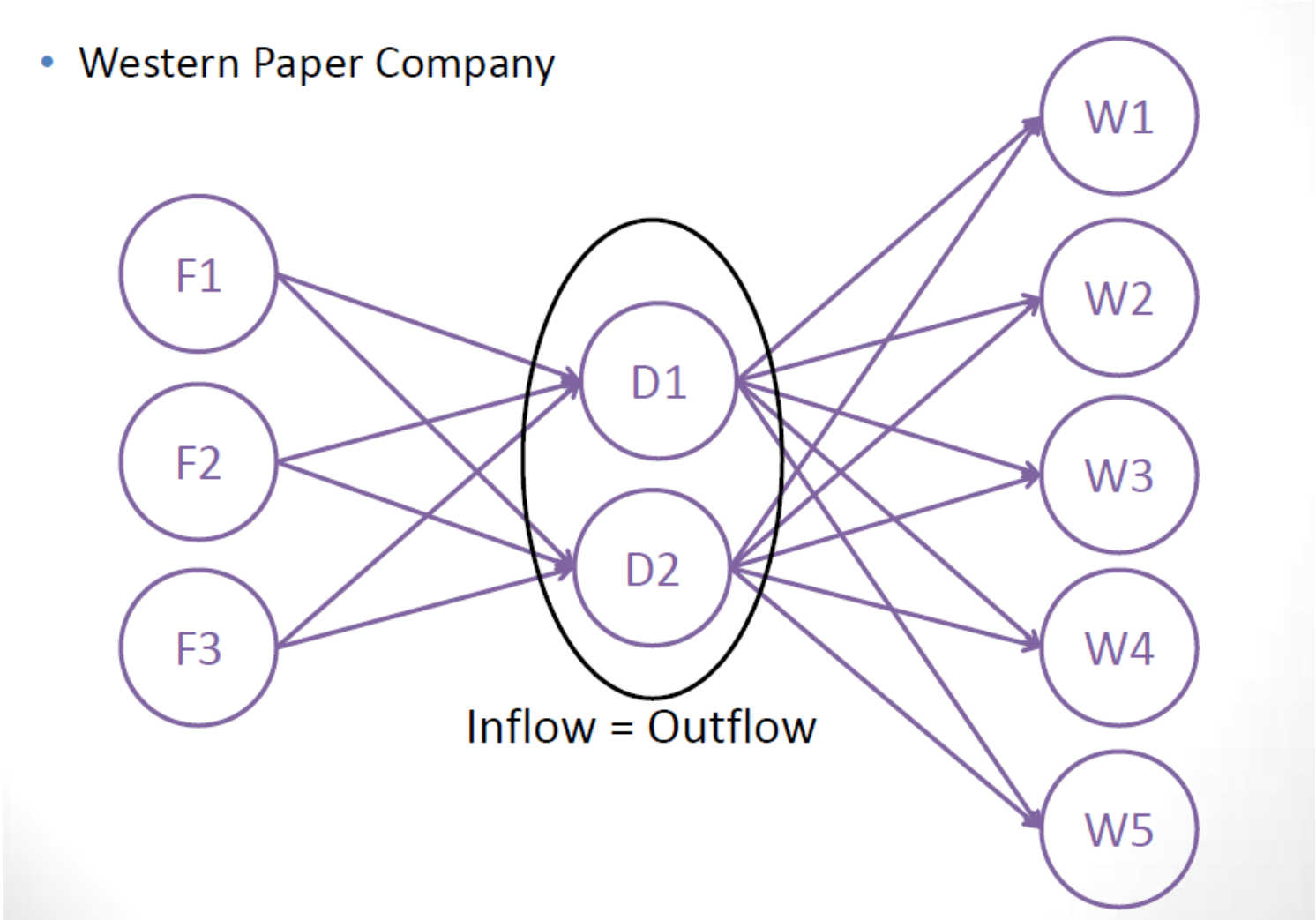
Transshipment Model Example

- Western Paper Company



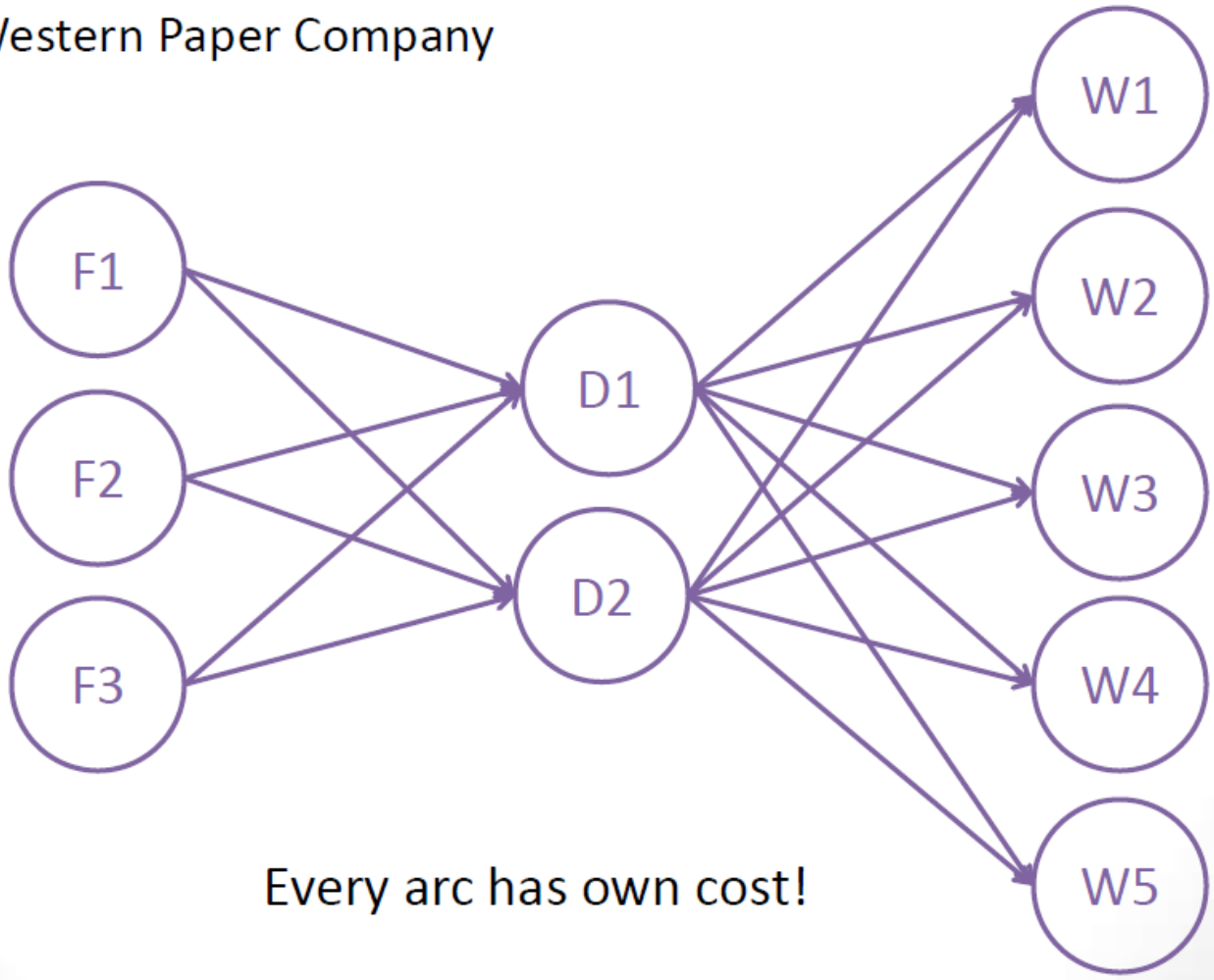
Transshipment Model Example

- Western Paper Company



Transshipment Model Example

- Western Paper Company



Every arc has own cost!

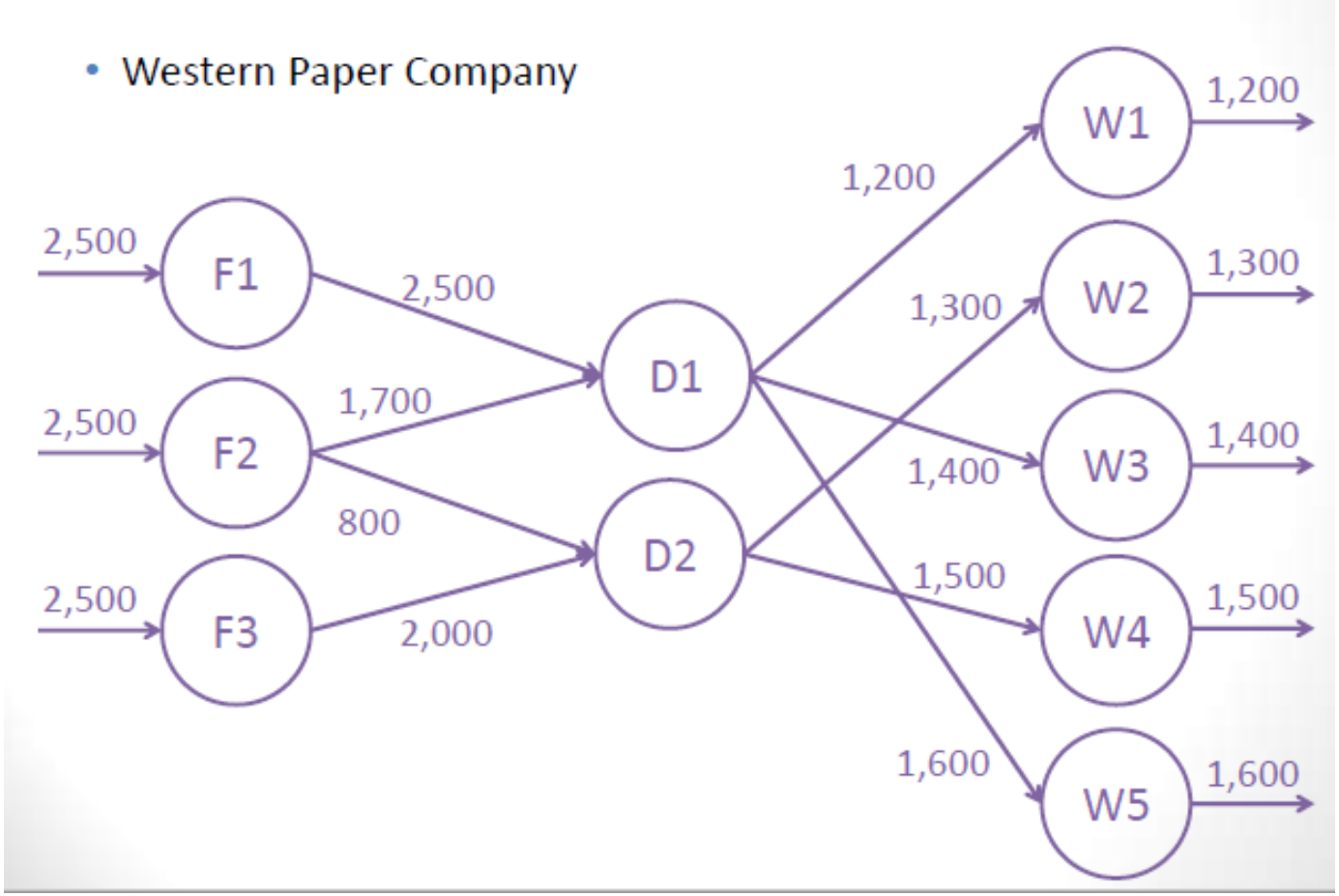
```
proc optmodel;
set Factory = /Fac1 Fac2 Fac3/;
set DistCenter = /Dist1 Dist2/;
set Warehouse = /Ware1 Ware2 Ware3 Ware4 Ware5/;
number F2DCost{Factory,DistCenter} = [1.28 1.36
                                      1.33 1.38
                                      1.68 1.55];
number D2WCost{DistCenter,Warehouse} = [0.60 0.42 0.32 0.44 0.68
                                        0.57 0.30 0.4 0.38 0.72];
number Demand{Warehouse} = [1200 1300 1400 1500 1600];
var F2D{Factory,DistCenter}>=0,
D2W{DistCenter,Warehouse}>=0;
min Cost = sum{i in Factory}(sum{j in DistCenter}(F2DCost[i,j]*F2D[i,j])) +
sum{k in DistCenter}(sum{l in Warehouse}(D2WCost[k,l]*D2W[k,l]));
con Cap {i in Factory}: sum{j in DistCenter} F2D[i,j] <= 2500;
con Dem {j in Warehouse}: sum{i in DistCenter}
D2W[i,j] >= Demand[j];
con Cons {j in DistCenter}: sum{k in Warehouse}D2W[j,k] -
sum{i in Factory}F2D[i,j] = 0;
solve;
print F2D D2W;
print F2D.rc D2W.rc;
quit;
```

Output

[1]	[2]	F2D	D2W
Dist1	Ware1		1200
Dist1	Ware2		0
Dist1	Ware3		1400
Dist1	Ware4		0
Dist1	Ware5		1600
Dist2	Ware1		0
Dist2	Ware2		1300
Dist2	Ware3		0
Dist2	Ware4		1500
Dist2	Ware5		0
Fac1	Dist1	2500	
Fac1	Dist2	0	
Fac2	Dist1	1700	
Fac2	Dist2	800	
Fac3	Dist1	0	
Fac3	Dist2	2000	

Transshipment Model Example

- Western Paper Company



R Code

```
f2dcost=matrix(c(1.28,1.36,1.33,1.38,1.68,1.55),nrow=3,byrow=T)
d2wcost=matrix(c(0.6,0.42,0.32,0.44,0.68,0.57,0.3,0.4,0.38,0.72),nrow=2,byrow=T)
demand=c(1200,1300,1400,1500,1600)
f.obj=c(as.numeric(f2dcost),as.numeric(d2wcost))
f.con=matrix(0,nrow=10,ncol=16)
for (i in 1:3)
{f.con[i,i]=1
 f.con[i,i+3]=1}
for (i in 1:5)
{f.con[i+3,6+2*i-1]=1
 f.con[i+3,6+2*i]=1}
temp=c(7,9,11,13,15)
temp2=c(8,10,12,14,16)
f.con[9,temp]=1
f.con[9,1:3]=-1
f.con[10,temp2]=1
f.con[10,4:6]=-1
f.rhs=c(rep(2500,3),demand,0,0)
f.dir=c(rep('<=',3),rep('>=',5),'=','=')
lp.model=lp (direction="min", f.obj, f.con, f.dir, f.rhs)

first.lp=matrix(result$x[1:6],nrow=3,byrow=F)
rownames(first.lp)=c('Fac1','Fac2','Fac3')
colnames(first.lp)=c('Dist1','Dist2')
first.lp
second.lp=matrix(result$x[7:16],nrow=2,byrow=F)
rownames(second.lp)=c('Dist1','Dist2')
colnames(second.lp)=c('Ware1','Ware2','Ware3','Ware4','Ware5')
second.lp
```

Output

```
> first.lp
      Dist1  Dist2
Fac1   2500      0
Fac2   1700    800
Fac3      0   2000
> second.lp
      Ware1 Ware2 Ware3 Ware4 Ware5
Dist1   1200     0   1400     0   1600
Dist2      0  1300     0   1500     0
```

Gurobi

```
f2dcost=matrix(c(1.28,1.36,1.33,1.38,1.68,1.55),nrow=3,byrow=T)
d2wcost=matrix(c(0.6,0.42,0.32,0.44,0.68,0.57,0.3,0.4,0.38,0.72),nrow=2,byrow=T)
demand=c(1200,1300,1400,1500,1600)
model=list()
model$obj=c(as.numeric(f2dcost),as.numeric(d2wcost))
model$A=matrix(0,nrow=10,ncol=16)
for (i in 1:3)
{model$A[i,i]=1
model$A[i,i+3]=1}
for (i in 1:5)
{model$A[i+3,6+2*i-1]=1
model$A[i+3,6+2*i]=1}
temp=c(7,9,11,13,15)
temp2=c(8,10,12,14,16)
model$A[9,temp]=1
model$A[9,1:3]=-1
model$A[10,temp2]=1
model$A[10,4:6]=-1
model$rhs=c(rep(2500,3),demand,0,0)
model$sense=c(rep('<=',3),rep('>=',5),'=','=')
model$modelsense="min"
result=gurobi(model,list())
first.result=matrix(result$x[1:6],nrow=3,byrow=F)
rownames(first.result)=c('Fac1','Fac2','Fac3')
colnames(first.result)=c('Dist1','Dist2')
first.result
second.result=matrix(result$x[7:16],nrow=2,byrow=F)
rownames(second.result)=c('Dist1','Dist2')
colnames(second.result)=c('Ware1','Ware2','Ware3','Ware4','Ware5')
second.result
result$objval
```

Output

```
> first.result
      Dist1  Dist2
Fac1   2500      0
Fac2   1700    800
Fac3      0   2000
> second.result
      Ware1 Ware2 Ware3 Ware4 Ware5
Dist1   1200     0   1400     0   1600
Dist2      0  1300     0  1500     0
```