Mixed and Integer linear programming

Types of Optimization

- Linear Programming – objective function and constraints are linear.
- Integer Linear Programming – objective function and constraints are linear but decision variables must be integers.
- Mixed Integer Linear Programming – same as ILP with only some decision variables restricted to integers.
- Non-linear Programming – objective function and constraints continuous but not all linear

Types of Optimization

- ~~Linear Programming – objective function and constraints are linear.~~
- Integer Linear Programming – objective function and constraints are linear but decision variables must be integers.
- Mixed Integer Linear Programming – same as ILP with only some decision variables restricted to integers.
- Non-linear Programming – objective function and constraints continuous but not all linear

ILP

- Sometimes linear programming is used to estimate ILP problems (if arrive with integers like chair example, this is optimal! Sometimes we can round)
- However, this is not always best
  - Can produce suboptimal solutions
- Moving from LP to ILP is a big step
  - Need different algorithm (common algorithm: Branch and Cut….which starts with LP)
  - More constraints (only integers)

Example: Veerman Furniture Company

	Hours per unit			
Department	Chairs	Desks	Tables	Hours Avail
Fabrication	4	6	2	1850
Assembly	3	5	7	2400
Shipping	3	2	4	1500
Demand potential	360	300	100	
Profit	$15	$24	$18	

Let's change this to be ILP

```
proc optmodel;
/* declare variables */
var Chairs>=0 integer, Desks>=0 integer, Tables>=0 integer;
/* maximize objective function (profit) */
max Profit = 15*Chairs + 24*Desks + 18*Tables;
/* subject to constraints */
con Assembly: 3*Chairs + 5*Desks +7*Tables<=2400;
con Shipping: 3*Chairs + 2*Desks + 4*Tables<=1500;
con Fabrication: 4*Chairs+6*Desks+2*Tables<=1850;
con DemandC: Chairs <=360;
con DemandD: Desks<=300;
con DemandT: Tables<=100;
solve ;
/* display solution */
print Chairs Desks Tables;
quit;
```

In R

```
library(lpSolve)
f.names=c('Chairs','Desks','Tables')
f.obj=c(15,24,18)
f.con=matrix(c(4,6,2,3,5,7,3,2,4,1,0,0,0,1,0,0,0,1),nrow=6,byrow=T)
f.dir=c("<=","<=","<=","<=","<=","<=")
f.rhs = c(1850,2400,1500,360,300,100)
lp.model=lp (direction="max", f.obj, f.con, f.dir, f.rhs,compute.sens =
1, all.int=T)
lp.model$status
names(lp.model$solution)=f.names

 lp.model$solution
 lp.model$objval
```

In Gurobi

```
library(gurobi)
library(prioritizr)
model <- list()
f.names=c('Chairs','Desks','Tables')
model$obj=c(15,24,18)
model$modelsense =“max”
model$A=matrix(c(4,6,2,3,5,7,3,2,4,1,0,0,0,1,0,0,0,1),nrow=6,by
row=T)
model$sense=c("<=","<=","<=","<=","<=","<=")
model$rhs = c(1850,2400,1500,360,300,100)
model$vtype   = “I“
result = gurobi(model, list())
print(result$status)
names(result$x)=f.names
print(result$x)
print(result$objval)
```

Example: Veerman Furniture Company

	Hours per unit			
Department	Chairs	Desks	Tables	Hours Avail
Fabrication	4	6	2	*1800*
Assembly	3	5	7	2400
Shipping	3	2	4	1500
Demand potential	360	300	100	
Profit	$15	$24	$18	

Chairs	Desks	Tables
0	266.67	100

Objective function is $8200

Note: if we "round", we will get an infeasible solution (4*0 + 6*267+2*100 = 1802)!!

Binary Choice Models

- Binary Choice Models are a form of ILP
- Further restrict variables to be binary (0 or 1)
- 2 Common Binary Choice Models:
  - Capital Budget Problem
  - Set Covering Problem

Binary Choice Models

- Binary Choice Models are a form of ILP
- Further restrict variables to be binary (0 or 1)
- 2 Common Binary Choice Models:
  - *Capital Budget Problem*
  - Set Covering Problem

Companies that want to have projects within a given year, but there is only a certain allocated budget to do a subset of these projects. How do we choose the most optimal subset of projects?

Example: Marr Corporation

- Has $160 Million for capital projects
- Five new projects to consider

1. New Information System
2. License New Technology from Other Firms
3. Build State-of-Art Recycling Facility
4. Install an Automated Machining Center in Production
5. Move Receiving Department to New Facility

- Each Project has a cost and a projected Net Present Value (NPV) over the life of the project
- Either a project is selected or not selected (no partial projects), therefore this is a binary choice model

Information

	Project 1	Project 2	Project 3	Project 4	Project 5
NPV	10	17	16	8	14
Expenditure	48	96	80	32	64

We want to maximize NPV.

Information

	Project 1	Project 2	Project 3	Project 4	Project 5
NPV	10	17	16	8	14
Expenditure	48	96	80	32	64

So, we are trying to Maximize NPV!

Maximize: $NPV = 10P_1 + 17P_2 + 16P_3 + 8P_4 + 14P_5$
Constraint: $48P_1 + 96P_2 + 80P_3 + 32P_4 + 64P_5 \leq 160$

$P_1,\ldots,P_5$ are binary --- 1 if project is done, 0 if it is not done

In SAS

```
proc optmodel;
var p1>=0 binary, p2>=0 binary, p3>=0 binary, p4>=0 binary, p5>=0 binary;
max NPV = 10*p1 + 17*p2 + 16*p3 + 8*p4 + 14*p5;
con cost: 48*p1 + 96*p2 + 80*p3 + 32*p4 + 64*p5 <=160;
solve;
print p1 p2 p3 p4 p5;
quit;
```

SAS Output

Solution Summary	
Solver	MILP
Algorithm	Branch and Cut
Objective Function	NPV
Solution Status	Optimal
Objective Value	34
Relative Gap	0
Absolute Gap	0
Primal Infeasibility	0
Bound Infeasibility	0

Integer Infeasibility	0
Best Bound	34
Nodes	1
Iterations	0
Presolve Time	0.00
Solution Time	0.00

p1	p2	p3	p4	p5
1	0	1	1	0

In R

```
f.names=c('p1','p2','p3','p4','p5')
f.obj=c(10,17,16,8,14)
f.con=matrix(c(48,96,80,32,64),nrow=1)
f.dir=("<=")
f.rhs = 160
lp.model=lp (direction="max", f.obj, f.con, f.dir, f.rhs,
all.bin=T)
lp.model$status
names(lp.model$solution)=f.names

lp.model$solution
lp.model$objval
```

Output

```
> lp.model$solution
p1 p2 p3 p4 p5
 1  0  1  1  0
> lp.model$objval
[1] 34
> sum(lp.model$solution*f.con)
[1] 160
```

In Gurobi

```
model <- list()
model$obj        <- c(10,17,16,8,14)
model$modelsense <- "max"
model$rhs        <- c(160)
model$sense       <- c("<=")
model$vtype       <- "B"
model$A          <-
matrix(c(48,96,80,32,64),nrow=1,byrow=T)
result <- gurobi(model, list())
print(result$status)
print(result$x)
print(result$objval)
```

Output

```
> print(result$status)
[1] "OPTIMAL"
> print(result$x)
[1] 1 0 1 1 0
> print(result$objval)
[1] 34
```

Binary Decision Variables

- Binary Choice Models
  - Capital Budget problem
  - Set Covering problem

Binary Decision Variables

- Binary Choice Models
  - ~~Capital Budget problem~~
  - Set Covering problem

Need to make sure that an area is "covered" by available units. For example, how many EMS stations are needed to cover Wake County?

Set Covering Example

- Emergency Coverage in Metropolis
  - Metropolis city is divided into 9 districts
  - 7 potential sites for emergency vehicles
  - Sites can reach some districts, but not others, in the required 3 minutes response time
  - Location of these sites MUST cover all districts (would like to have the least amount of sites that can accomplish this)

Set Covering Information

District	Site 1	Site 2	Site 3	Site 4	Site 5	Site 6	Site 7
1	0	1	0	1	0	0	1
2	1	0	0	0	0	1	1
3	0	1	0	0	0	1	1
4	0	1	1	0	1	1	0
5	1	0	1	0	1	0	0
6	1	0	0	1	0	1	0
7	1	0	0	0	0	0	1
8	0	0	1	1	1	0	0
9	1	0	0	0	1	0	0

Want to minimize the number of sites while making sure all districts are covered.

Set Covering example setup

Min: Sites = $S_1 + S_2 + S_3 + S_4 + S_5 + S_6 + S_7$

Subject to:

$$S_2 + S_4 + S_7 \geq 1$$
$$S_1 + S_6 + S_7 \geq 1$$
$$S_2 + S_6 + S_7 \geq 1$$
$$S_2 + S_3 + S_5 + S_6 \geq 1$$
$$S_1 + S_3 + S_5 \geq 1$$
$$S_1 + S_4 + S_6 \geq 1$$
$$S_1 + S_7 \geq 1$$
$$S_3 + S_4 + S_5 \geq 1$$
$$S_1 + S_5 \geq 1$$

In SAS

```
proc optmodel;
var s1 >=0 binary, s2 >=0 binary, s3 >=0 binary, s4 >=0 binary, s5
>=0 binary, s6 >=0 binary, s7 >=0 binary;
Min Sites = s1 +s2 +s3+s4+s5+s6+s7;
con District1: S2 + S4 + S7 >=1;
con District2: S1 + S6 + S7 >=1;
con District3: S2 + S6 + S7 >=1;
con District4: S2 + S3 + S5 + S6 >=1;
con District5: S1 + S3 + S5 >=1;
con District6: S1 + S4 + S6 >=1;
con District7: S1 + S7 >=1;
con District8: S3 + S4 + S5 >=1;
con District9:S1 + S5 >=1;
solve;
print s1 s2 s3 s4 s5 s6 s7;
quit;
```

The OPTMODEL Procedure

Solution Summary	
Solver	MILP
Algorithm	Branch and Cut
Objective Function	Sites
Solution Status	Optimal
Objective Value	3
Relative Gap	0
Absolute Gap	0
Primal Infeasibility	0
Bound Infeasibility	0
Integer Infeasibility	0
Best Bound	3
Nodes	1
Iterations	12
Presolve Time	0.00
Solution Time	0.02

s1	s2	s3	s4	s5	s6	s7
0	0	0	0	1	1	1

However, there is more than one optimal solution!!

```
proc optmodel;
var s1 >=0 binary, s2 >=0 binary, s3 >=0 binary, s4 >=0 binary, s5 >=0 binary,
s6 >=0 binary, s7 >=0 binary;
Min Sites = s1 +s2 +s3+s4+s5+s6+s7;
con District1: S2 + S4 + S7 >=1;
con District2: S1 + S6 + S7 >=1;
con District3: S2 + S6 + S7 >=1;
con District4: S2 + S3 + S5 + S6 >=1;
con District5: S1 + S3 + S5 >=1;
con District6: S1 + S4 + S6 >=1;
con District7: S1 + S7 >=1;
con District8: S3 + S4 + S5 >=1;
con District9:S1 + S5 >=1;
solve with CLP / FINDALLSOLNS;
print s1 s2 s3 s4 s5 s6 s7;
quit;
```

Output

The OPTMODEL Procedure

Solution Summary	
Solver	CLP
Objective Function	Sites
Solution Status	Optimal
Objective Value	3
Solutions Found	8
Presolve Time	0.00
Solution Time	0.00

s1	s2	s3	s4	s5	s6	s7
0	0	0	0	1	1	1

In R

```
library(lpSolve)
f.names=c('s1','s2','s3','s4','s5', 's6','s7')
f.obj=c(1,1,1,1,1,1,1)
f.con=matrix(c(0,1,0,1,0,0,1,1,0,0,0,0,1,1,0,1,0,0,0,1,1,0,1,1,0,1,1,0,1,0,1,0,1,0,0,1
,0,0,1,0,1,0,1,0,0,0,0,0,1,0,0,1,1,1,0,0,1,0,0,0,1,0,0),nrow=9,byrow=T)
f.dir=c(">=",">=",">=",">=",">=",">=",">=",">=",">=")
f.rhs = rep(1,9)
lp.model=lp (direction="min", f.obj, f.con, f.dir, f.rhs, all.bin=T,num.bin.solns=100)
```

Output

```
> lp.model$status
[1] 0
> numcol=length(f.obj)
> numsol=lp.model$num.bin.solns
>
solution=matrix(head(lp.model$solution,numcol*numsol),nrow
=numsol,byrow=T)
> colnames(solution)=f.names
> solution
     s1 s2 s3 s4 s5 s6 s7
[1,]  1  1  1  0  0  0  0
[2,]  1  0  1  0  0  0  1
[3,]  1  1  0  1  0  0  0
[4,]  1  1  0  0  1  0  0
[5,]  1  0  0  0  1  0  1
[6,]  1  0  0  1  0  1  0
[7,]  0  0  0  1  1  0  1
[8,]  0  0  0  0  1  1  1
```

In Gurobi

```
model <- list()
model$obj        <- c(1,1,1,1,1,1,1)
model$modelsense <- "min"
model$rhs        <- rep(1,9)
model$sense       <- c(">=",">=",">=",">=",">=",">=",">=",">=",">=")
model$vtype       <- "B"
model$A           <-
matrix(c(0,1,0,1,0,0,1,1,0,0,0,0,1,1,0,1,0,0,0,1,1,0,1,1,0,1,1,0,1,0,1,0,1,0,0,1,0,0,1,0,1,
0,1,0,0,0,0,0,1,0,0,1,1,1,0,0,1,0,0,0,1,0,0),nrow=9,byrow=T)
result <- gurobi(model, list())
params                 <- list()
params$PoolSolutions  <- 1024
params$PoolGap        <- 0.10
params$PoolSearchMode <- 2
gurobi_write(model, 'poolsearch_R.lp')
result <- gurobi(model, params, list())
result$pool
result.pool.matrix=matrix(nrow=8,ncol=7)
for (i in 1:8)
{result.pool.matrix[i,]=result$pool[[i]]$xn}
round(result.pool.matrix)
```

Output

round(result.pool.matrix)

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7]
[1,]    0    0    0    0    1    1    1
[2,]    1    0    0    0    1    0    1
[3,]    0    0    0    1    1    0    1
[4,]    1    0    0    1    0    1    0
[5,]    1    1    0    0    1    0    0
[6,]    1    0    1    0    0    0    1
[7,]    1    1    1    0    0    0    0
[8,]    1    1    0    1    0    0    0
```

Logical Relationships

- Binary variables can also be used to model if-then-else situations.
- For example, here are 5 common relationships:

1. At least m items
2. At most n items
3. Exactly k items
4. Mutually exclusive items
5. Contingency based items

For example:

- Marr Corporation
  - Have $160 million for capital projects over the year.
  - Five new projects to consider.
  - Each project has **cost** and projected **NPV** over life of project. Still trying to maximize NPV.
  - Add 4 Restrictions:
    1. Must have at least one international project ($P2$ or $P5$).
    2. Only have the staff to support 2 projects total.
    3. Projects 4 & 5 have the same resources, so can't have both.
    4. Project 5 requires project 3 be selected.

Set up

MAX:

$$NPV = 10P_1 + 17P_2 + 16P_3 + 8P_4 + 14P_5$$

Subject to:

$48P_1 + 96P_2 + 80P_3 + 32P_4 + 64P_5 \leq 160$

$P_2 + P_5 \geq 1$

$P_1 + P_2 + P_3 + P_4 + P_5 = 2$

$P_4 + P_5 \leq 1$

$P_3 - P_5 \geq 0$

P_1, P_2, P_3, P_4, P_5 are all binary variables.

SAS

```
proc optmodel;
var P1 >=0 binary, P2 >=0 binary, P3 >=0 binary, P4 >=0 binary,
P5 >=0 binary;
max NPV=10*P1 + 17*P2 + 16*P3 + 8*P4 + 14*P5;
con Cost: 48*P1 + 96*P2 + 80*P3 + 32*P4 + 64*P5 <= 160;
con International: P2 + P5 >= 1;
con ShortStaff: P1 + P2 + P3 + P4 + P5 = 2;
con CommonResources: P4 + P5 <= 1;
con ProjectDependent: P3 - P5 >= 0;
solve;
print P1 P2 P3 P4 P5;
quit;
```

Solution Summary	
Solver	MILP
Algorithm	Branch and Cut
Objective Function	NPV
Solution Status	Optimal
Objective Value	30
Relative Gap	0
Absolute Gap	0
Primal Infeasibility	0
Bound Infeasibility	0
Integer Infeasibility	0
Best Bound	30
Nodes	1
Iterations	6
Presolve Time	0.00
Solution Time	0.00

P1	P2	P3	P4	P5
0	0	1	0	1

In R

```
library(lpSolve)
f.names=c('p1','p2','p3','p4','p5')
f.obj=c(10,17,16,8,14)
f.con=matrix(c(48,96,80,32,64,0,1,0,0,1,1,1,1,1,1,0,0,0,1,1,0,0,1,0,-1),nrow=5,
byrow=T)
f.dir=c("<=",">=","=","<=",">=")
f.rhs = c(160,1,2,1,0)
lp.model=lp (direction="max", f.obj, f.con, f.dir, f.rhs, all.bin=T)
```

Output

```
> lp.model$status
[1] 0
> names(lp.model$solution)=f.names
> lp.model$solution
p1 p2 p3 p4 p5
 0  0  1  0  1
> lp.model$objval
[1] 30
```

In Gurobi

```
model <- list()
model$obj        <- c(10,17,16,8,14)
model$modelsense <- "max"
model$rhs        <- c(160,1,2,1,0)
model$sense       <- c("<=",">=","=","<=",">=")
model$vtype       <- "B"
model$A           <- 
matrix(c(48,96,80,32,64,0,1,0,0,1,1,1,1,1,1,0,0,0,1,1,0,0,1,0,-1),nrow=5,
byrow=T)
result <- gurobi(model, list())
print(result$status)
print(result$x)
print(result$objval)
```

Output

```
> print(result$status)
[1] "OPTIMAL"
> print(result$x)
[1] 0 0 1 0 1
> print(result$objval)
[1] 30
```

Linking Constraints & Fixed Costs

- Up until this point we have discussed variable costs, but some decisions also have **fixed costs** that have to be accounted for.
- Fixed Cost
  - One time cost that is incurred when any amount of an item is made.
  - Binary variables are used to account for fixed costs.

Linking Constraints & Fixed Costs

- Fixed Cost – binary variable
- Variable Cost – continuous variable

x = level of production
y = 0 if x = 0
y =1 if x >1

$$Cost = F * y + C * x$$

Fixed Cost

Variable Cost

Linking Constraints & Fixed Costs

- Fixed Cost – binary variable
- Variable Cost – continuous variable

$$Cost = F * y + C * x$$

Binary Variable

Continuous Variable

Linking Constraints & Fixed Costs

- How do we invoke the restriction that if y = 1 then x > 0, but if y = 0 then x = 0?

same logic without the if statement

$$x \leq Max * y$$

if y=1, x<=2000

if y=0, x<=0

x>=0 (x can only be 0)

Upper bound on x

Fixed Cost Example

Mayhugh Manufacturing Company

- Produce 3 product families.
- Each product family requires production hours in 3 different departments.
- Each product family requires its own sales force **no matter how large the sales volume**.

Fixed Cost Example

Department	Product Family 1	Product Family 2	Product Family 3	Hours Available
A	3	4	8	2,000
B	3	5	6	2,000
C	2	3	9	2,000
Sales Force Cost	60	200	100	
Maximum Demand	400	300	50	
Profit	1.2	1.8	2.2	

Maximize profit. Note: the Sales force cost is NOT included in the profit (need to take this under consideration in optimization).

Fixed Cost Example

- MAX:

$$Profit = 1.2F_1 + 1.8F_2 + 2.2F_3 - 60FY_1 - 200FY_2 - 100FY_3$$

- Subject to:

$$3F_1 + 4F_2 + 8F_3 \leq 2000$$

$$3F_1 + 5F_2 + 6F_3 \leq 2000$$

$$2F_1 + 3F_2 + 9F_3 \leq 2000$$

$$F_1 - 400FY_1 \leq 0$$

$$F_2 - 300FY_2 \leq 0$$

$$F_3 - 50FY_3 \leq 0$$

- FY_1, FY_2, FY_3 are all binary variables.

SAS Code

```
proc optmodel;
var F1 >=0, F2 >=0, F3 >=0, F1yes >=0 binary, F2yes >=0 binary,
F3yes >=0 binary;
max Profit=1.2*F1 + 1.8*F2 + 2.2*F3 – 60*F1yes - 200*F2yes - 100*F3yes;
con DepartmentA: 3*F1 + 4*F2 + 8*F3 <= 2000;
con DepartmentB: 3*F1 + 5*F2 + 6*F3 <= 2000;
con DepartmentC: 2*F1 + 3*F2 + 9*F3 <= 2000;
con FixedF1: F1 - 400*F1yes <= 0;
con FixedF2: F2 - 300*F2yes <= 0;
con FixedF3: F3 - 50*F3yes <= 0;
solve;
print F1 F2 F3 F1yes F2yes F3yes;
quit;
```

Output

Algorithm	Branch and Cut
Objective Function	Profit
Solution Status	Optimal
Objective Value	508
Relative Gap	0
Absolute Gap	0
Primal Infeasibility	0
Bound Infeasibility	0
Integer Infeasibility	0
Best Bound	508
Nodes	1
Iterations	16
Presolve Time	0.00
Solution Time	0.00

F1	F2	F3	F1yes	F2yes	F3yes
400	160	0	1	1	0

Another SAS code

```
proc optmodel;
var x{1..3}>=0, y{1..3}>=0 binary;
number p{1..3} = [1.2 1.8 2.2];
number A{1..3,1..3} = [3 4 8
                       3 5 6
                       2 3 9];
number b{1..3} = [2000 2000 2000];
number f{1..3} = [60 200 100];
number M{1..3} = [400 300 50];
max Profit = sum{j in 1..3} (p[j]*x[j] - f[j]*y[j]);
con Department {i in 1..3}: sum{j in 1..3}A[i,j]*x[j] <= b[i];
con Demand {j in 1..3}: x[j] - M[j]*y[j] <= 0;
solve;
print x y;
quit;
```

Output

Solution Status	Optimal
Objective Value	508
Relative Gap	0
Absolute Gap	0
Primal Infeasibility	0
Bound Infeasibility	0
Integer Infeasibility	0
Best Bound	508
Nodes	1
Iterations	16
Presolve Time	0.00
Solution Time	0.00

[1]	x	y
1	400	1
2	160	1
3	0	0

Another SAS Code

```
proc optmodel;
set Product = /Family1 Family2 Family3/;
set Department = /DepA DepB DepC/;
number Profit{Product} = [1.2 1.8 2.2];
number HrsReq{Department,Product} = [3 4 8
3 5 6
2 3 9];
number HrsCap{Department} = [2000 2000 2000];
number FixedCost{Product} = [60 200 100];
number Demand{Product} = [400 300 50];
var x{Product}>=0, y{Product}>=0 binary;
max TotalProfit =
sum{j in Product} (Profit[j]*x[j] - FixedCost[j]*y[j]);
con Dep {i in Department}:
sum{j in Product}HrsReq[i,j]*x[j] <= HrsCap[i];
con Dem {j in Product}: x[j] - Demand[j]*y[j] <= 0;
solve;
print x y;
quit;
```

Output

Solution Status	Optimal
Objective Value	508
Relative Gap	0
Absolute Gap	0
Primal Infeasibility	0
Bound Infeasibility	0
Integer Infeasibility	0
Best Bound	508
Nodes	1
Iterations	16
Presolve Time	0.00
Solution Time	0.00

[1]	x	y
Family1	400	1
Family2	160	1
Family3	0	0

In R

```
library(lpSolve)
f.names=c('x1','x2','x3','y1','y2','y3')
f.obj=c(1.2,1.8,2.2,-60,-200,-100)
f.con=matrix(c(3,4,8,0,0,0,3,5,6,0,0,0,2,3,9,0,0,0,1,0,0,-
400,0,0,0,1,0,0,-300,0,0,0,1,0,0,-50),nrow=6, byrow=T)
f.dir=c("<=","<=","<=","<=","<=","<=")
f.rhs = c(2000,2000,2000,0,0,0)
lp.model=lp (direction="max", f.obj, f.con, f.dir, f.rhs,
binary.vec=c(4,5,6))
```

Output

```
> lp.model$status
[1] 0
> names(lp.model$solution)=f.names
> lp.model$solution
 x1  x2  x3  y1  y2  y3
400 160   0   1   1   0
> lp.model$objval
[1] 508
```

In Gurobi

```
#####Fixed cost example
model <- list()
model$obj        <- c(1.2,1.8,2.2,-60,-200,-100)
model$modelsense <- "max"
model$rhs        <- c(2000,2000,2000,0,0,0)
model$sense       <- c("<=","<=","<=","<=","<=","<=")
model$vtype       <- c("C","C","C","B","B","B")
model$A          <- matrix(c(3,4,8,0,0,0,3,5,6,0,0,0,2,3,9,0,0,0,1,0,0,-
400,0,0,0,1,0,0,-300,0,0,0,1,0,0,-50),nrow=6, byrow=T)
result <- gurobi(model, list())
print(result$status)
print(result$x)
print(result$objval)
```

Output

```
> print(result$status)
[1] "OPTIMAL"
> print(result$x)
[1] 400 160   0   1   1   0
> print(result$objval)
[1] 508
```

Facility Location Model

- Goal is to set up an optimal number of supply locations and shipping schedules.
- We will focus on short-term supply chain models later in the network models section.
- This problem is a long-term version of the problem where we are deciding supply locations as well as shipping schedules.
- Conceivable to think these supply locations must remain for the foreseeable future.

Facility Location Model Example

- Levinson Foods Company
  - 10 distribution centers with monthly volumes.
  - 6 of these locations are able to support warehouses.
  - Warehouses have additional capacity, but monthly operating costs.
  - Variable shipping costs between each location and potential warehouse is calculated.
  - Want to develop the optimal shipping schedule as well as determine which locations to upgrade to warehouses.

Facility Location Model Example

Location	Alb	Boise	Dall	Denv	Hous	Okla	Phoe	Salt	SanA	Wich
Albuquerque	0	47	32	22	42.5	27	23	30	36.5	29.5
Dallas	32	79.5	0	39	12.5	10.5	50	63	13.5	17
Denver	21	42	39	0	51.5	31.5	40.5	24	47.5	26
Houston	42.5	91	12.5	51.5	0	23	58	72	10	31
Pheonix	23	49	50	40.5	58	49	0	32.5	50	52
San Antonio	36.5	83.5	13.5	47.5	10	24	50	66.5	0	32

Facility Location Model Example

Center / Warehouse	Volume	Capacity	Cost
Albuquerque (W)	3,200	16,000	$140,000
Boise	2,500		
Dallas (W)	6,800	20,000	$150,000
Denver (W)	4,000	10,000	$100,000
Houston (W)	9,600	10,000	$110,000
Oklahoma City	3,500		
Phoenix (W)	5,000	12,000	$125,000
Salt Lake City	1,800		
San Antonio (W)	7,400	10,000	$120,000
Wichita	2,700		

Facility Location Model Example

- MIN: $Cost = Variable\ cost + Fixed\ Cost$
- Subject to:
  - Warehouse capacity – can't send more than it can store
  - Distribution center volume – send each distribution center pre-specified volume
  - If warehouse isn't used then don't build!
  - Each warehouse/distribution center combination is a decision variable!

SAS Code

```
proc optmodel;
var x{1..6,1..10}>=0, y{1..6}>=0 binary;
number c{1..6,1..10}=[0 47 32 22 42.5 27 23 30 36.5 29.5
32 79.5 0 39 12.5 10.5 50 63 13.5 17
21 42 39 0 51.5 31.5 40.5 24 47.5 26
42.5 91 12.5 51.5 0 23 58 72 10 31
23 49 50 40.5 58 49 0 32.5 50 52
36.5 83.5 13.5 47.5 10 24 50 66.5 0 32];
number Capacity{1..6} = [16000 20000 10000
10000 12000 12000];
number Demand{1..10} = [3200 2500 6800 4000 9600
3500 5000 1800 7400 2700];
number FixedCost{1..6} = [140000 150000 100000
110000 125000 120000];
min Cost = sum{i in 1..6}(sum{j in 1..10}(c[i,j]*x[i,j]) +
FixedCost[i]*y[i]);
con Cap {i in 1..6}: sum{j in 1..10} x[i,j] –
Capacity[i]*y[i] <= 0;
con Dem {j in 1..10}: sum{i in 1..6}x[i,j] = Demand[j];
solve;
print x y;
quit;
```

Output

Solution Summary	
Solver	MILP
Algorithm	Branch and Cut
Objective Function	Cost
Solution Status	Optimal
Objective Value	884550
Relative Gap	0
Absolute Gap	0
Primal Infeasibility	2.734629E-13
Bound Infeasibility	9.094947E-13
Integer Infeasibility	2.220446E-16
Best Bound	884550
Nodes	1
Iterations	66
Presolve Time	0.00
Solution Time	0.00

	x									
	1	2	3	4	5	6	7	8	9	10
1	0	0	0	0	0	0	0	0	0	0
2	0	0	6800	0	0	3500	0	0	7000	2700
3	1700	2500	0	4000	0	0	0	1800	0	0
4	0	0	0	0	9600	0	0	0	400	0
5	1500	0	0	0	0	0	5000	0	0	0
6	0	0	-0	0	0	0	0	0	0	0

[1]	y
1	0
2	1
3	1
4	1
5	1
6	0

Rcode

```
library(lpSolve)
cost=matrix(c(0,47,32,22,42.5,27,23,30,36.5,29.5,32,79.5,0,39,12.5,10.5,50,63,13.5,1
7,21,42,39,0,51.5,31.5,40.5,24,47.5,26,42.5,91,12.5,51.5,0,23,58,72,10,31,23,49,50,40.
5,58,49,0,32.5,50,52,36.5,83.5,13.5,47.5,10,24,50,66.5,0,32), nrow=6,byrow=T)
capacity=c(16000, 20000, 10000,10000, 12000, 12000)
demand=c(3200, 2500, 6800, 4000, 9600, 3500, 5000, 1800, 7400, 2700)
fixed.cost=c(140000, 150000, 100000, 110000, 125000, 120000)
f.obj=c(as.numeric(t(cost)),fixed.cost)
f.con=matrix(0,nrow=16,ncol=66)
for (i in 1:6)
{f.con[i,]=c(rep(0,10*(i-1)),rep(1,10),rep(0,10*(6-i)),rep(0,i-1),-capacity[i],rep(0,6-i))
}
for (j in 1:10)
{for (i in 1:6)
{f.con[(j+6),(10*(i-1)+j)] = 1
}}
f.dir=c(rep('<=',6),rep('=',10))
f.rhs=c(rep(0,6),demand)
lp.model=lp (direction="min", f.obj, f.con, f.dir, f.rhs, binary.vec=c(61,62,63,64,65,66))
```

Output

```
> x.solution=matrix(lp.model$solution[1:60],nrow=6,byrow=T)
> y.solution=lp.model$solution[61:66]
> lp.model$status
[1] 0
> x.solution
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    0    0    0    0    0    0    0    0    0     0
[2,]    0    0 6800    0    0 3500    0    0 7000  2700
[3,] 1700 2500    0 4000    0    0    0 1800    0     0
[4,]    0    0    0    0 9600    0    0    0  400     0
[5,] 1500    0    0    0    0    0 5000    0    0     0
[6,]    0    0    0    0    0    0    0    0    0     0
> y.solution
[1] 0 1 1 1 1 0
```

Gurobi

```
cost=matrix(c(0,47,32,22,42.5,27,23,30,36.5,29.5,32,79.5,0,39,12.5,10.5,50,63,13.5
,17,21,42,39,0,51.5,31.5,40.5,24,47.5,26,42.5,91,12.5,51.5,0,23,58,72,10,31,23,49,5
0,40.5,58,49,0,32.5,50,52,36.5,83.5,13.5,47.5,10,24,50,66.5,0,32),
nrow=6,byrow=T)
capacity=c(16000, 20000, 10000,10000, 12000, 12000)
demand=c(3200, 2500, 6800, 4000, 9600, 3500, 5000, 1800, 7400, 2700)
fixed.cost=c(140000, 150000, 100000, 110000, 125000, 120000)
model <- list()
model$obj        <- c(as.numeric(t(cost)),fixed.cost)
model$modelsense <- "min"
model$rhs        <- c(rep(0,6),demand)
model$sense       <- c(rep('<=',6),rep('=',10))
model$vtype       <- c(rep("C",60),rep("B",6))
model$A          <- matrix(0,nrow=16,ncol=66)
for (i in 1:6)
{model$A[i,]=c(rep(0,10*(i-1)),rep(1,10),rep(0,10*(6-i)),rep(0,i-1),-
capacity[i],rep(0,6-i))}
for (j in 1:10)
{for (i in 1:6)
{model$A[(j+6),(10*(i-1)+j)] = 1}}
result <- gurobi(model, list())
```

Output

```
> print(result$objval)
[1] 884550
> matrix(result$x[1:60],nrow=6,byrow=T)
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    0    0    0    0    0    0    0    0    0     0
[2,]    0    0 6800    0    0 3500    0    0 7000  2700
[3,] 1700 2500    0 4000    0    0    0 1800    0     0
[4,]    0    0    0    0 9600    0    0    0  400     0
[5,] 1500    0    0    0    0    0 5000    0    0     0
[6,]    0    0    0    0    0    0    0    0    0     0
> result$x[61:66]
[1] 0 1 1 1 1 0
```