You’re braver than you believe, stronger than you seem, and smarter than you think. -   A.A. Milne

Optimization

Why Optimization?

Optimization is important in many businesses to find the ***best set of decisions*** for a particular performance. For example…

Optimize operational efficiency: capital, personnel, equipment, vehicles, facilities.

Create measurable return on investment: ***Optimize costs, earnings and service***.

There are applications of optimization in most industries including: manufacturing, transportation, logistics, financial services, utilities, energy, telecommunications, government, defense and retail.

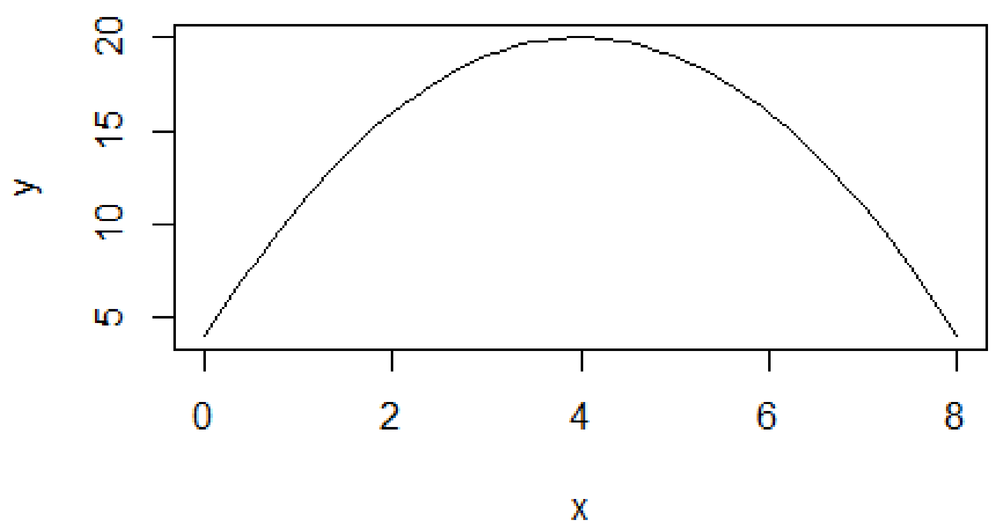
What is optimization in the mathematical sciences?

Optimizing a function is finding the maximum (or minimum) a function can take.

What is optimization in the mathematical sciences?

Optimizing a function is finding the maximum (or minimum) a function can take.

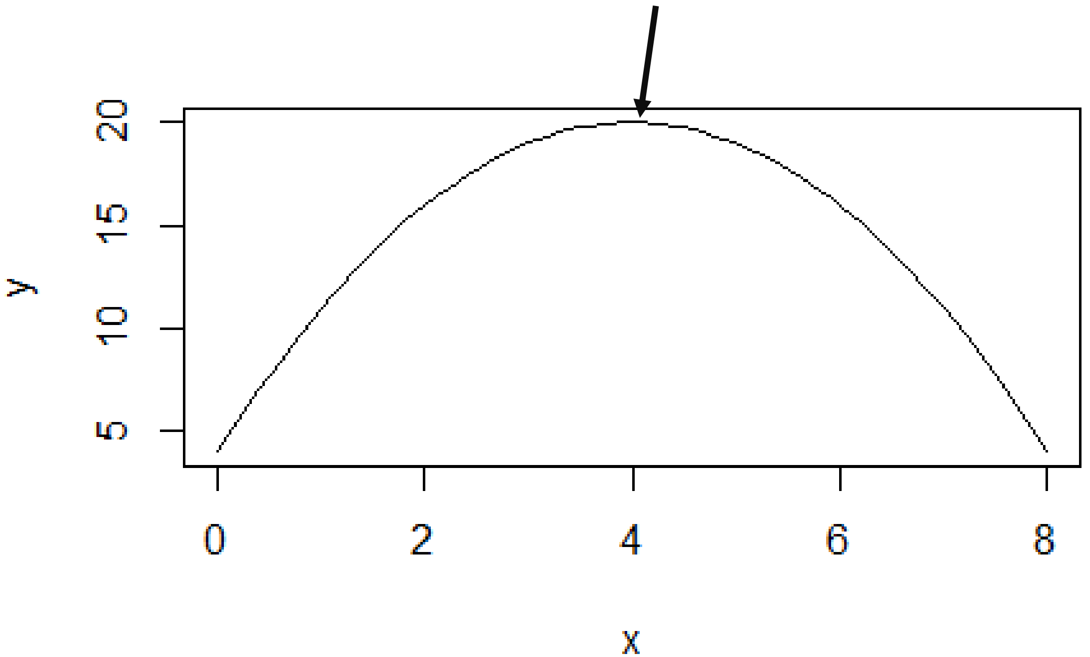
Goal is find out what values of the "decision variables" (or input variables) optimize this function



What is optimization in the mathematical sciences?

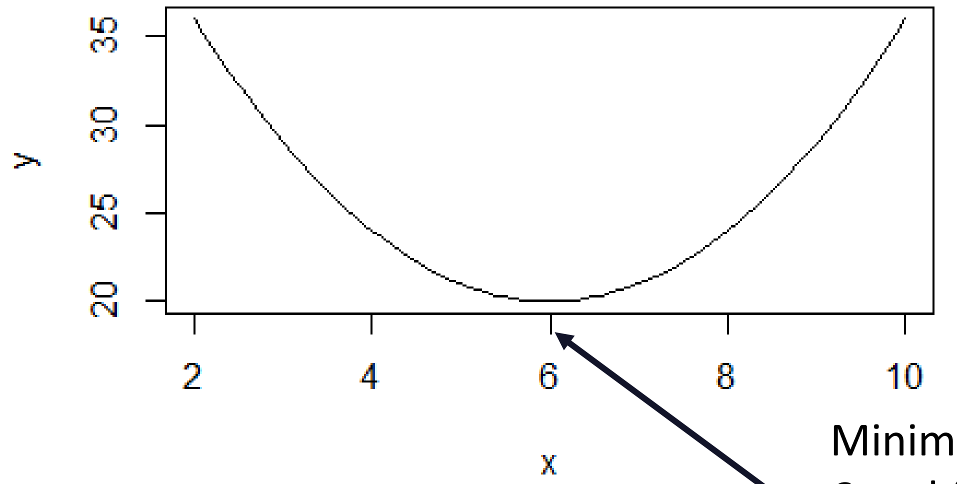
In example below, the x-variable is the "decision variable" and the y is the function we are trying to optimize…

Maximum of function



The maximum occurs when the decision or "input" variable is 4 (the optimal value of the "output" is 20).

Minimize a function



Minimum occurs at "input" of 6 and "output"(function) at this value is 20.

Terminology

The “input” variables are the variables in which we can change to optimize a function. These variables are referred to as the **decision variables**.

The **objective function** is the function in which we are trying to optimize (either maximize or minimize). This function is a function of the decision variables. We are trying to find the best values of the decision variables that optimize this function.

The **constraints** are functions of the decision variables that define the constraints of the problem.

Parameters are values inherent in the problem that we are not able to control

Optimization layout

Decision variables: x_1, x_2,…x_k

Objective function:

$$\Sigma a_i x_i = a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots a_k x_k$$

a_1, a_2, …a_k are the coefficients in the objective function

Constraints:

$$\Sigma b_{1i} x_i \leq c_1$$

$$\Sigma b_{2i} x_i \geq c_2$$

$$\Sigma b_{3i} x_i = c_3$$

…

b_{ji} are the constraint coefficients; c_1,c_2….are **parameters** defined in the model

Optimization layout

Decision variables: $\boxed{x_1, x_2, \ldots x_k}$

Objective function:

$$\Sigma a_i x_i = a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots a_k x_k$$

$a_1, a_2, \ldots a_k$ are the coefficients in the objective function

Constraints:

$$\Sigma b_{1i} x_i \leq c_1$$

$$\Sigma b_{2i} x_i \geq c_2$$

$$\Sigma b_{3i} x_i = c_3$$

…

b_{ji} are the constraint coefficients; $c_1, c_2 \ldots$ are **parameters** defined in the model

Optimization layout

Decision variables: $x_1, x_2,\ldots x_k$

Objective function:

$$\boxed{\Sigma a_i x_i = a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots a_k x_k}$$

$a_1, a_2, \ldots a_k$ are the coefficients in the objective function

Constraints:

$$\Sigma b_{1i} x_i \leq c_1$$

$$\Sigma b_{2i} x_i \geq c_2$$

$$\Sigma b_{3i} x_i = c_3$$

…

b_{ji} are the constraint coefficients; $c_1, c_2 \ldots$ are **parameters** defined in the model

Optimization layout

Decision variables: $x_1, x_2, \ldots x_k$

Objective function:

$$\Sigma a_i x_i = a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots a_k x_k$$

$a_1, a_2, \ldots a_k$ are the coefficients in the objective function

Constraints:

$$\Sigma b_{1i} x_i \leq c_1$$

$$\Sigma b_{2i} x_i \geq c_2$$

$$\Sigma b_{3i} x_i = c_3$$

…

b_{ji} are the constraint coefficients; $c_1, c_2 \ldots$ are **parameters** defined in the model

Optimization layout

Decision variables: $x_1, x_2, \ldots x_k$

Objective function:

$$\Sigma a_i x_i = a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots a_k x_k$$

$a_1, a_2, \ldots a_k$ are the coefficients in the objective function

Constraints:

$$\Sigma b_{1i} x_i \leq c_1$$

$$\Sigma b_{2i} x_i \geq c_2$$

$$\Sigma b_{3i} x_i = c_3$$

…

b_{ji} are the constraint coefficients; c_1, c_2….are **parameters** defined in the model

Optimization layout

Decision variables: $x_1, x_2, \ldots x_k$

Objective function:

$$\Sigma a_i x_i = a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots a_k x_k$$

$a_1, a_2, \ldots a_k$ are the coefficients in the objective function

Constraints:

$$\Sigma b_{1i} x_i \leq c_1$$

$$\Sigma b_{2i} x_i \geq c_2$$

$$\Sigma b_{3i} x_i = c_3$$

…

b_{ji} are the constraint coefficients; $c_1, c_2 \ldots$ are **parameters** defined in the model

Optimization layout

Decision variables: x_1, x_2,…x_k

Objective function:

$$\Sigma a_i x_i = a_1 x_1 + a_2 x_2 + a_3 x_3 + \ldots a_k x_k$$

a_1, a_2, …a_k are the coefficients in the objective function

Constraints:

$$\Sigma b_{1i} x_i \leq c_1$$

$$\Sigma b_{2i} x_i \geq c_2$$

$$\Sigma b_{3i} x_i = c_3$$

…

b_{ji} are the constraint coefficients; c_1,c_2….are **parameters** defined in the model

Outputs from an optimization

No optimization exists

More than one solution exists

There exists one unique solution to the problem

Other possibilities, but these are the most recognized

4 main types of optimization problems

Linear programming – objective function and constraints are linear (LP)

Integer linear programming – objective function and constraints are linear, but decision variables MUST be integers (ILP)

Mixed integer linear programming (MILP)– same as ILP with only some decision variables restricted to integers

Non-linear programming – objective function and constraints are continuous, but not all are linear

Linear Programming

The **feasible region** is the region defined by the constraints (need to find the optimal solution to the objective function over this region)

One of the easiest solutions for solving a linear programming problem is the ***Simplex*** method

- Strategy is to move along the edges of the feasible region until the optimal value of the objective function is found

Example

Decision variables: x_1, x_2

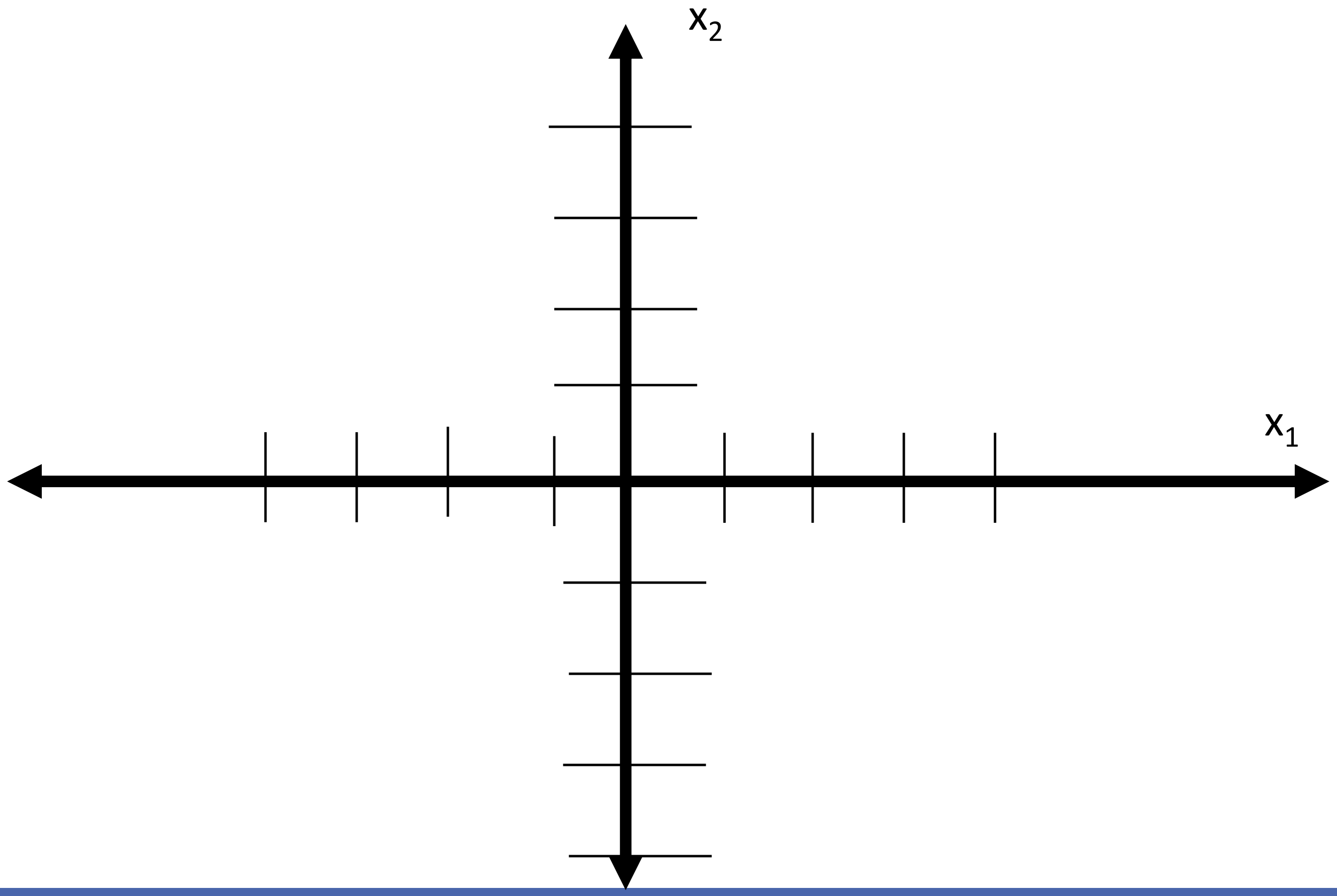
Find maximum of $2x_1 + 3x_2$ (this is the objective function)

Constraints:

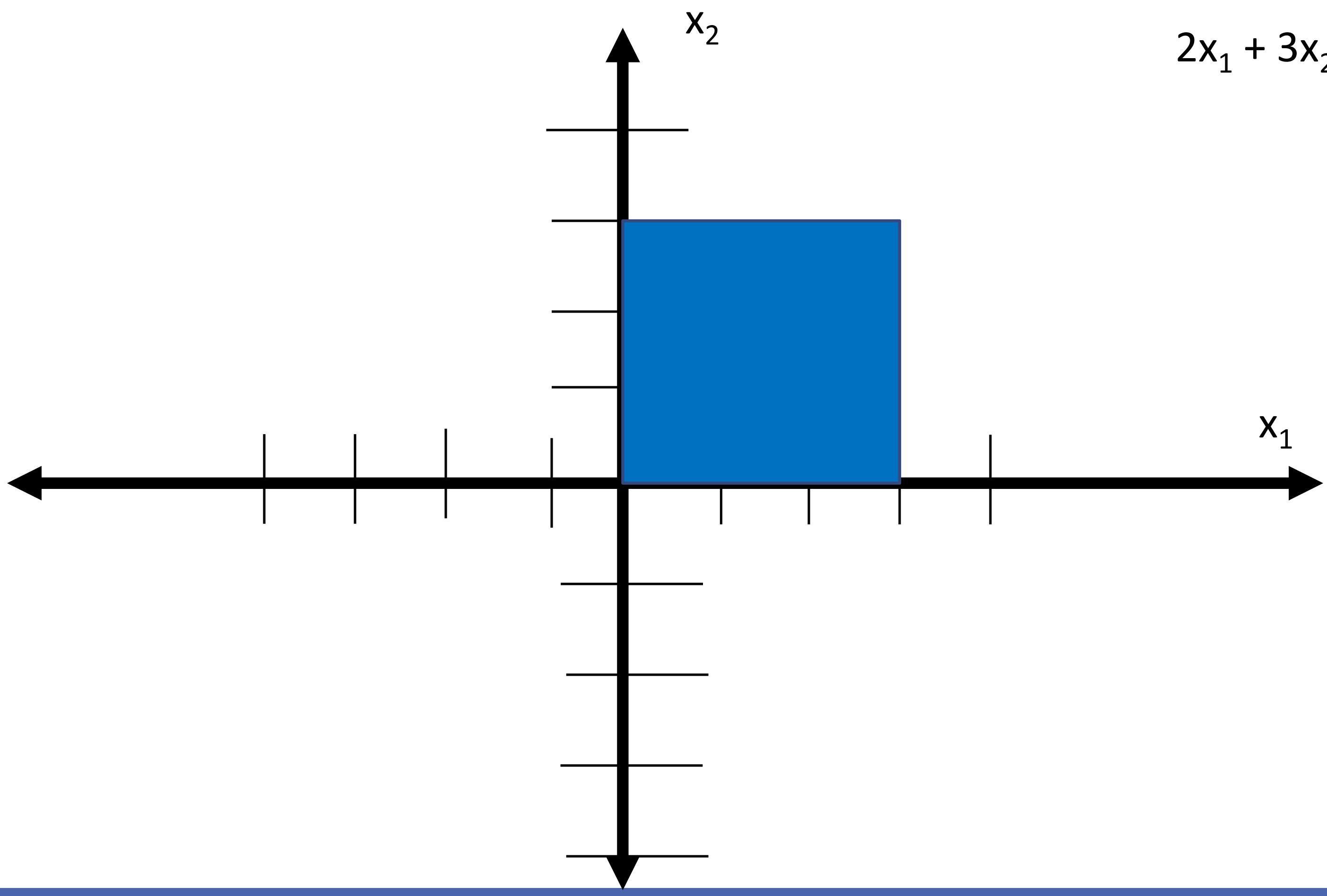
$0 \leq x_1 \leq 3$

$0 \leq x_2 \leq 3$

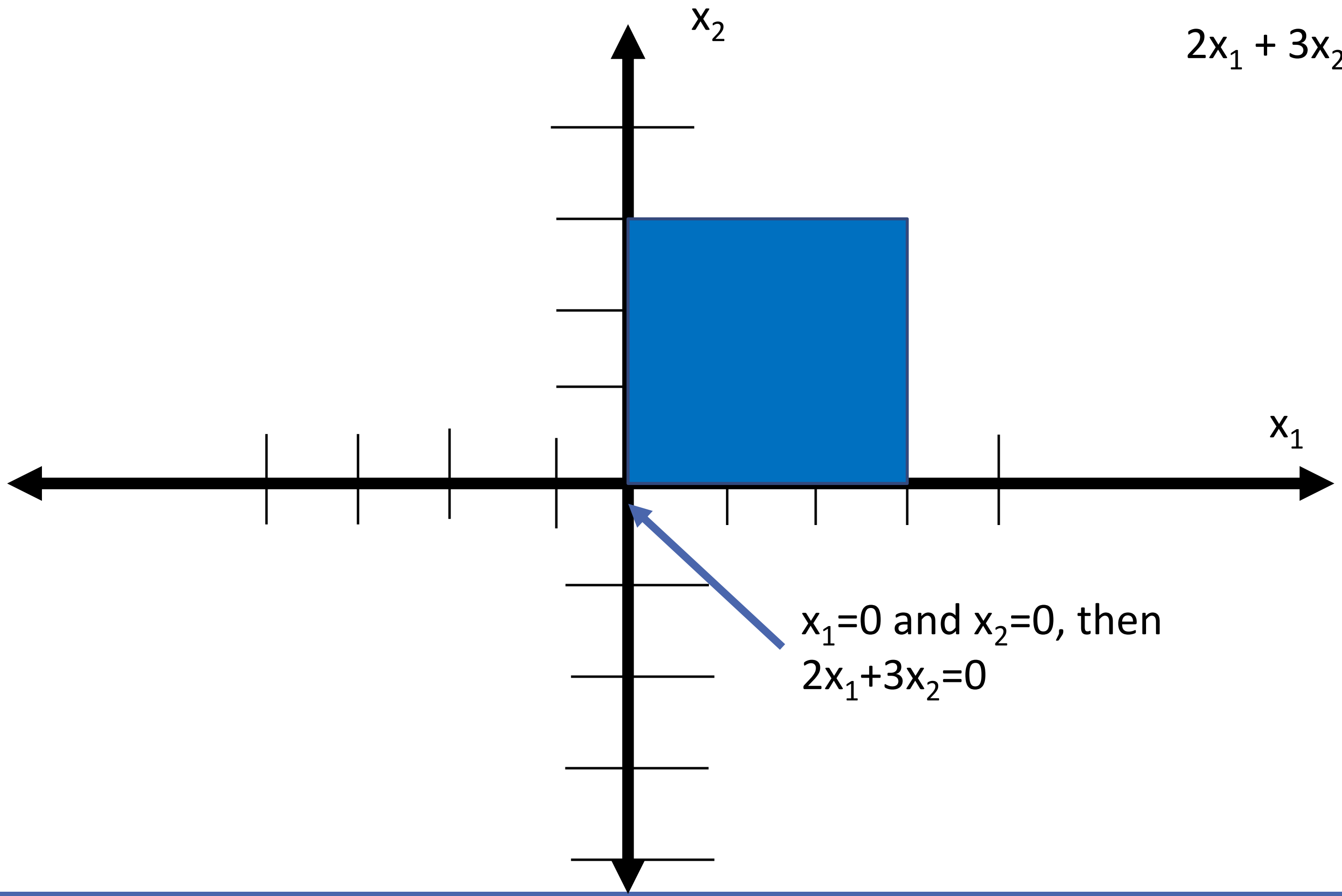
Feasible Region?



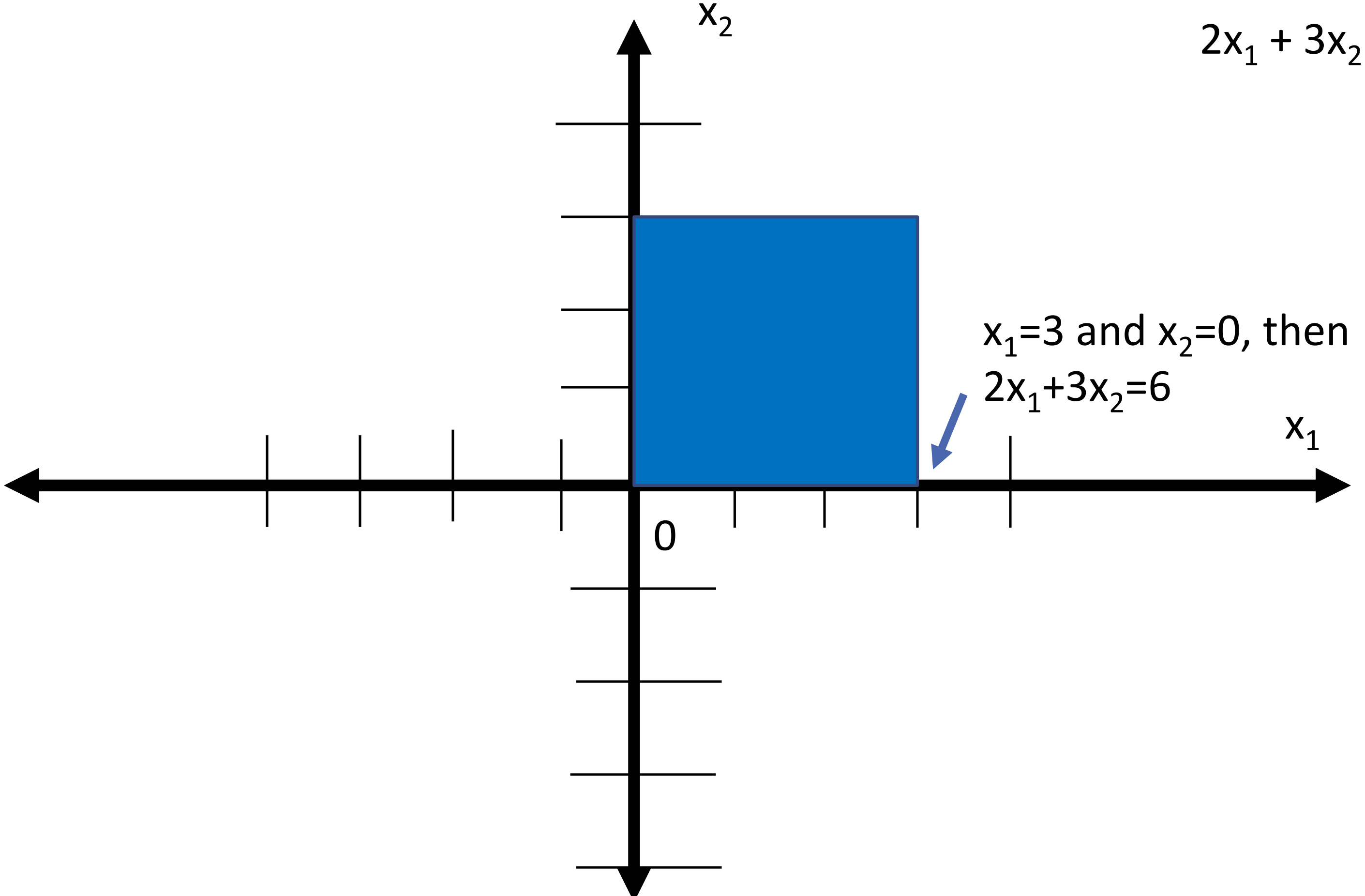
Feasible Region?



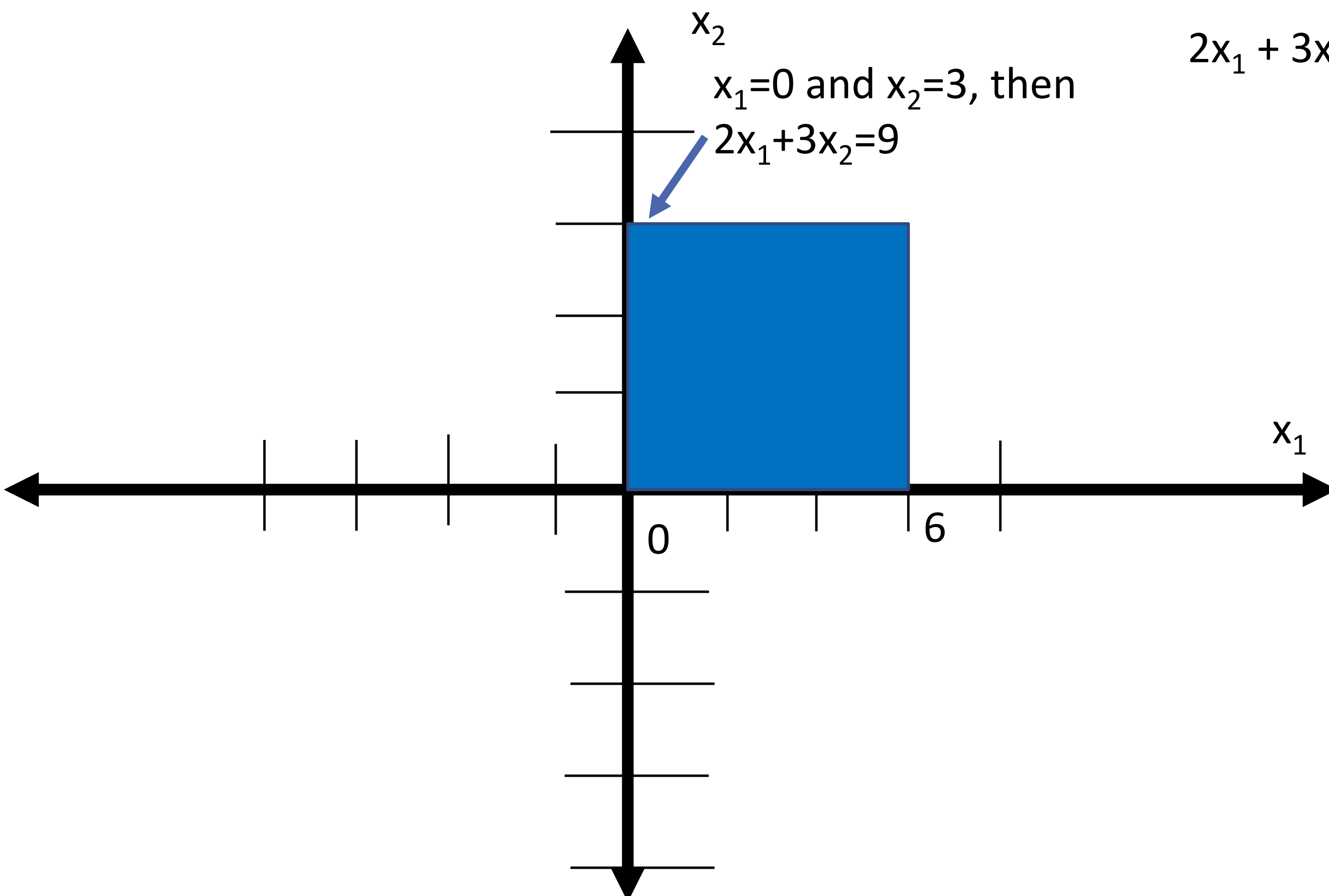
Feasible Region?



Feasible Region?

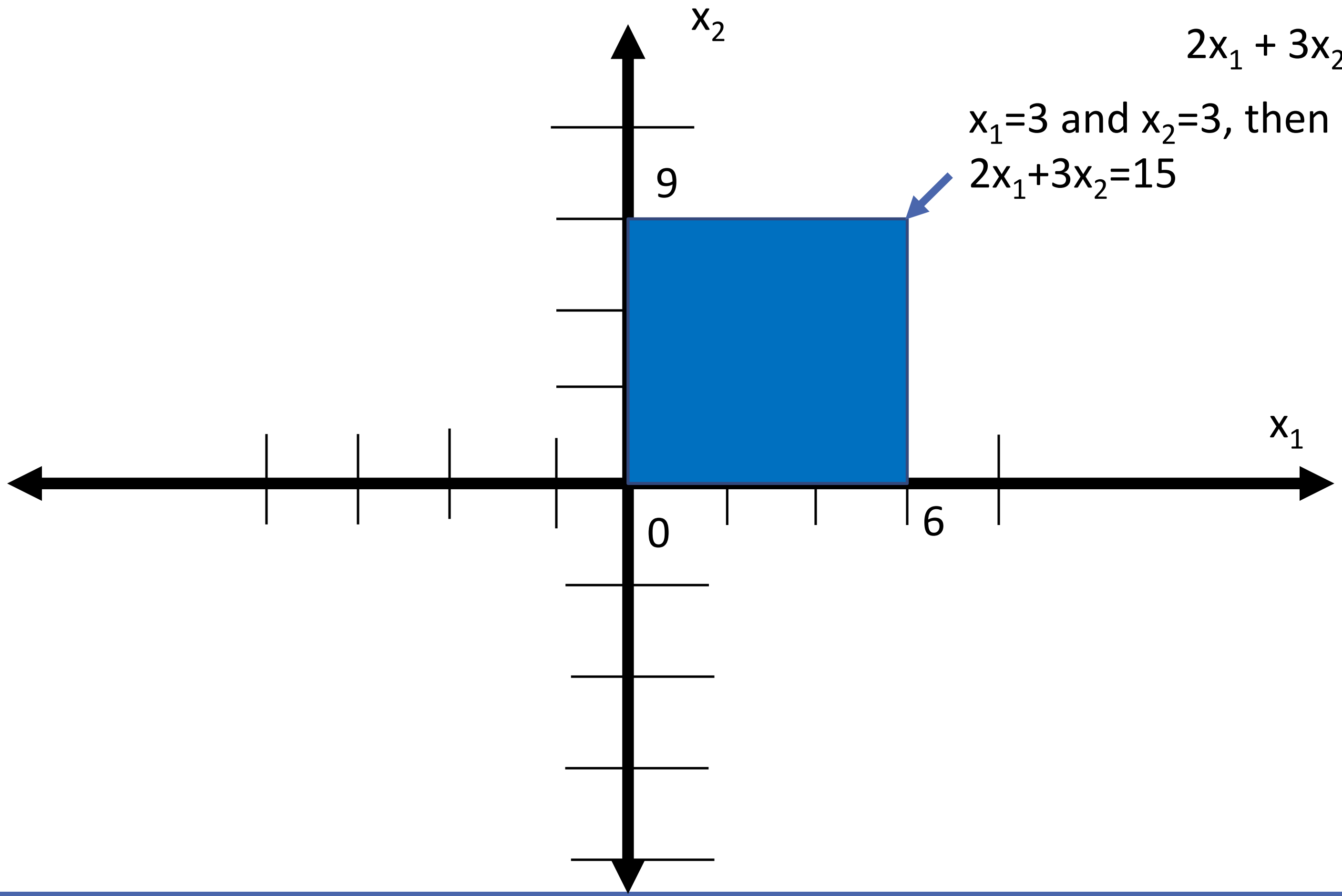


Feasible Region?

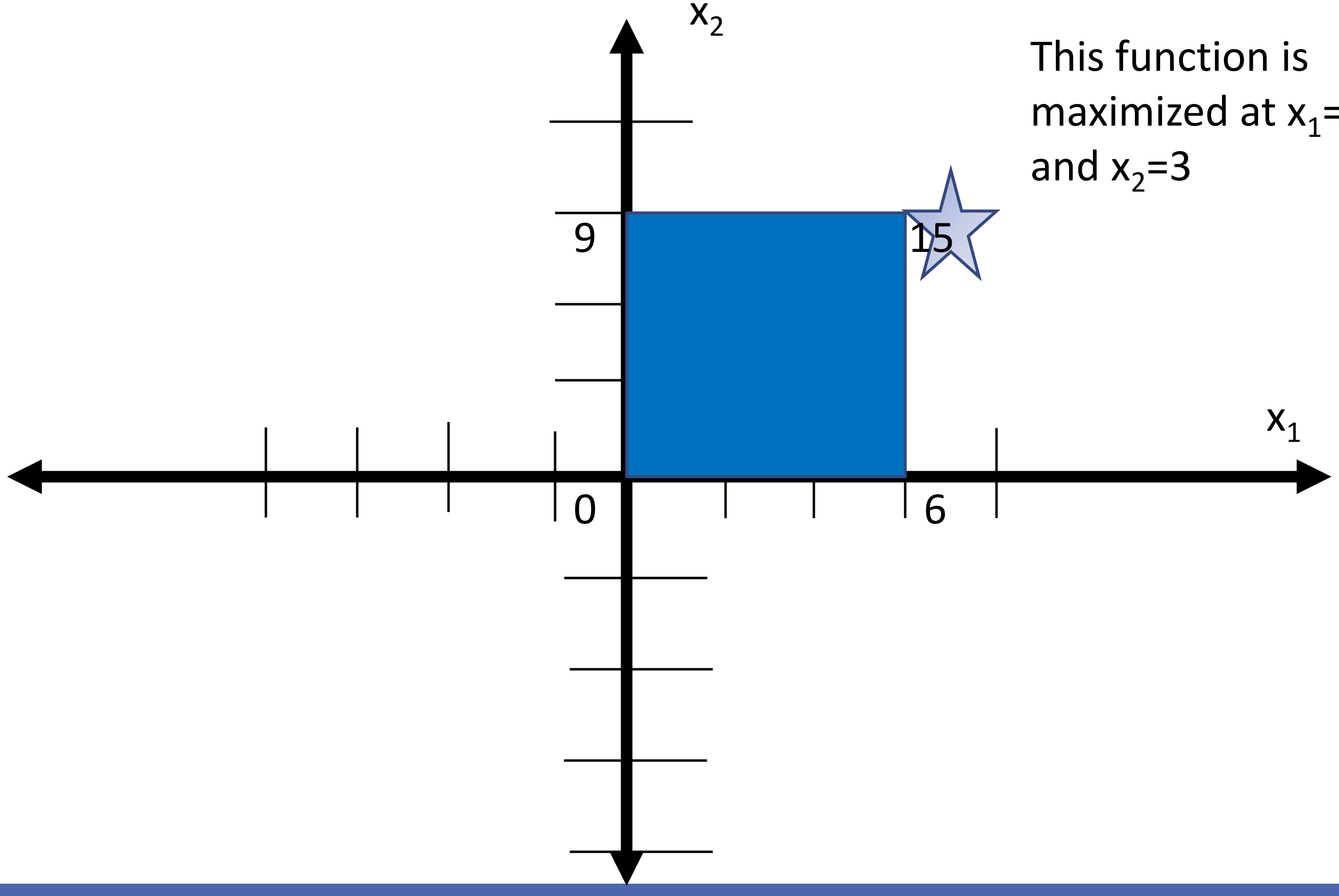


$2x_1 + 3x_2$

Feasible Region?



Feasible Region?



This function is maximized at x_1=3 and x_2=3

Second Example of Simplex algorithm (Chairs=C, Desks=D)

MAXIMIZE Profit: $15C + 24D$

CONSTRAINTS:

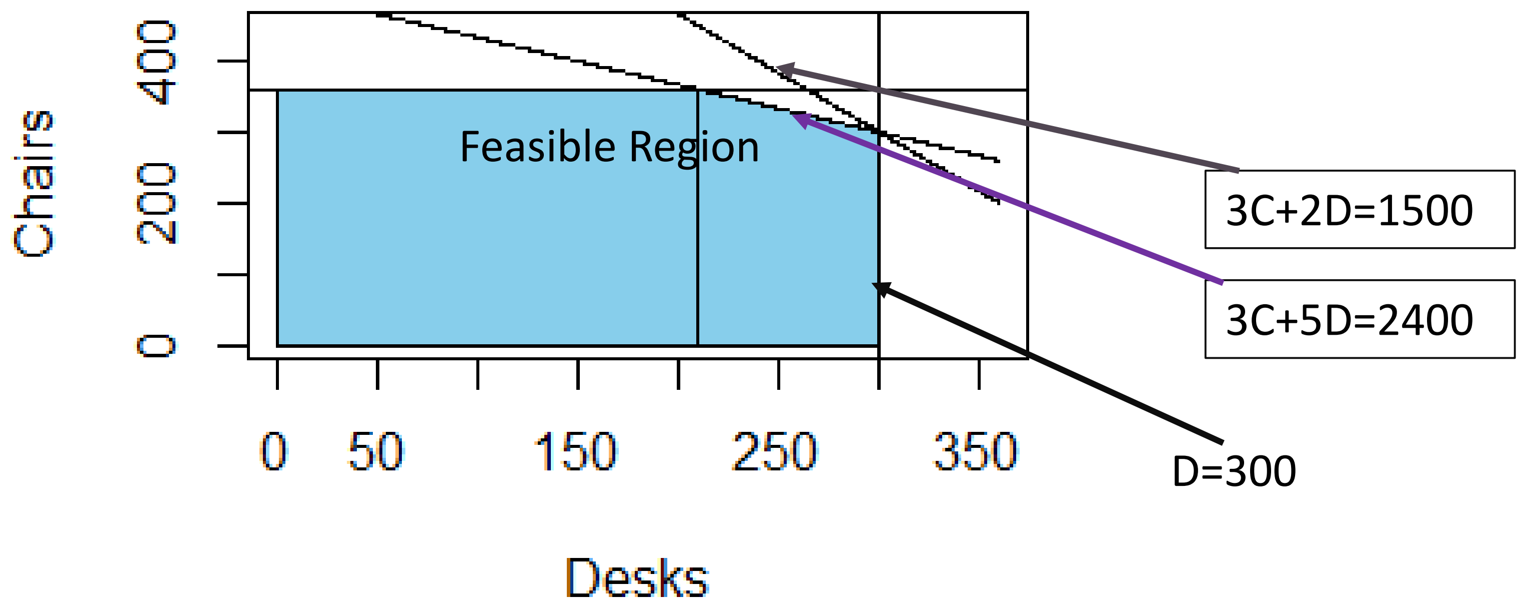
$3C + 5D \leq 2400$

$3C + 2D \leq 1500$

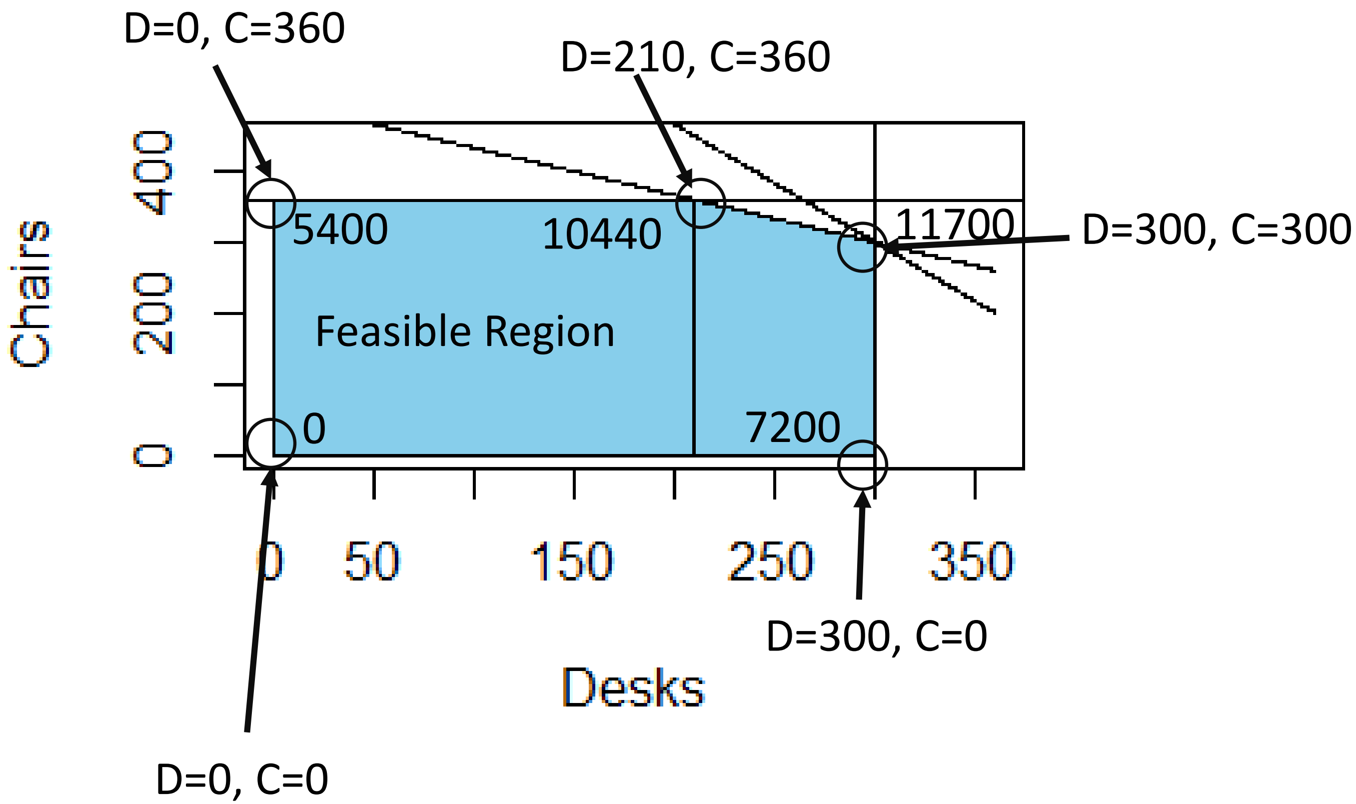
$0 \leq C \leq 360$

$0 \leq D \leq 300$

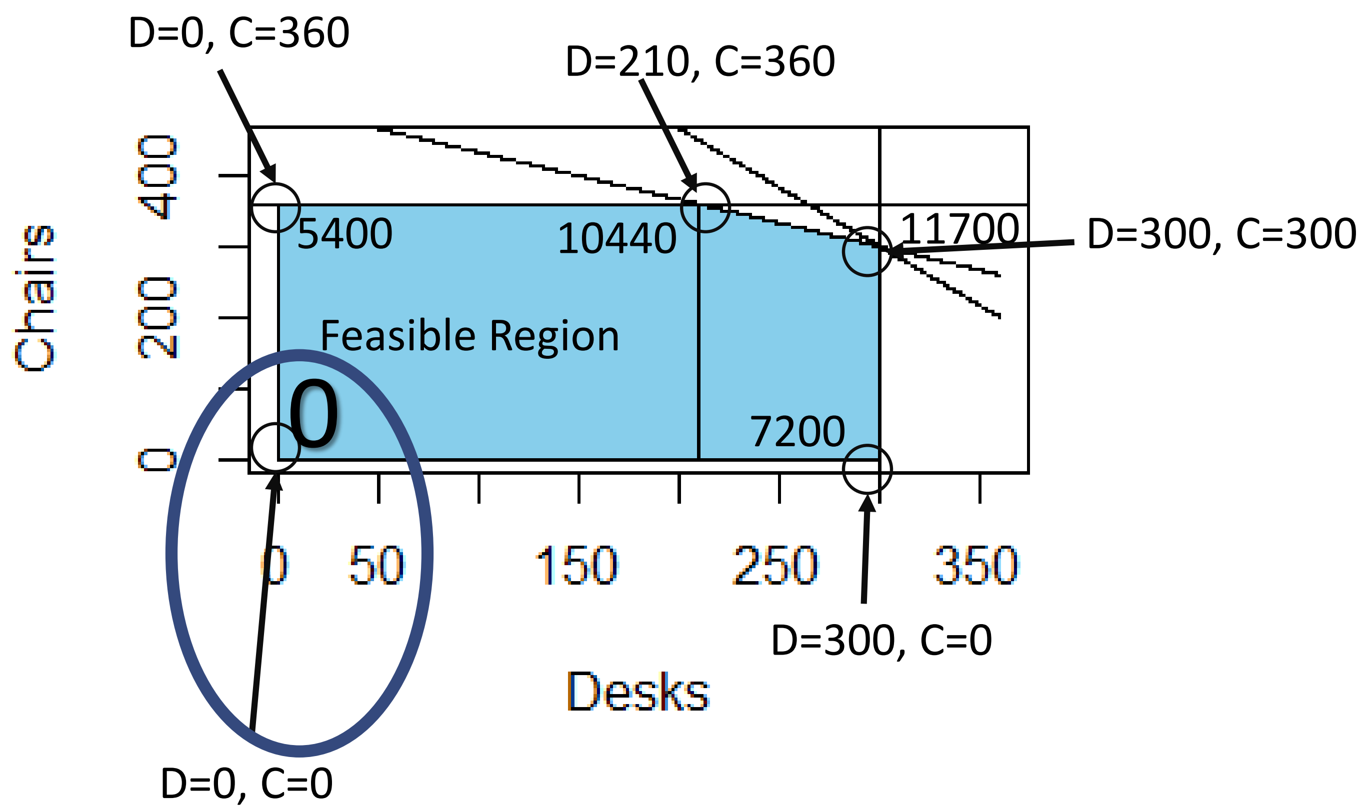
Feasible region



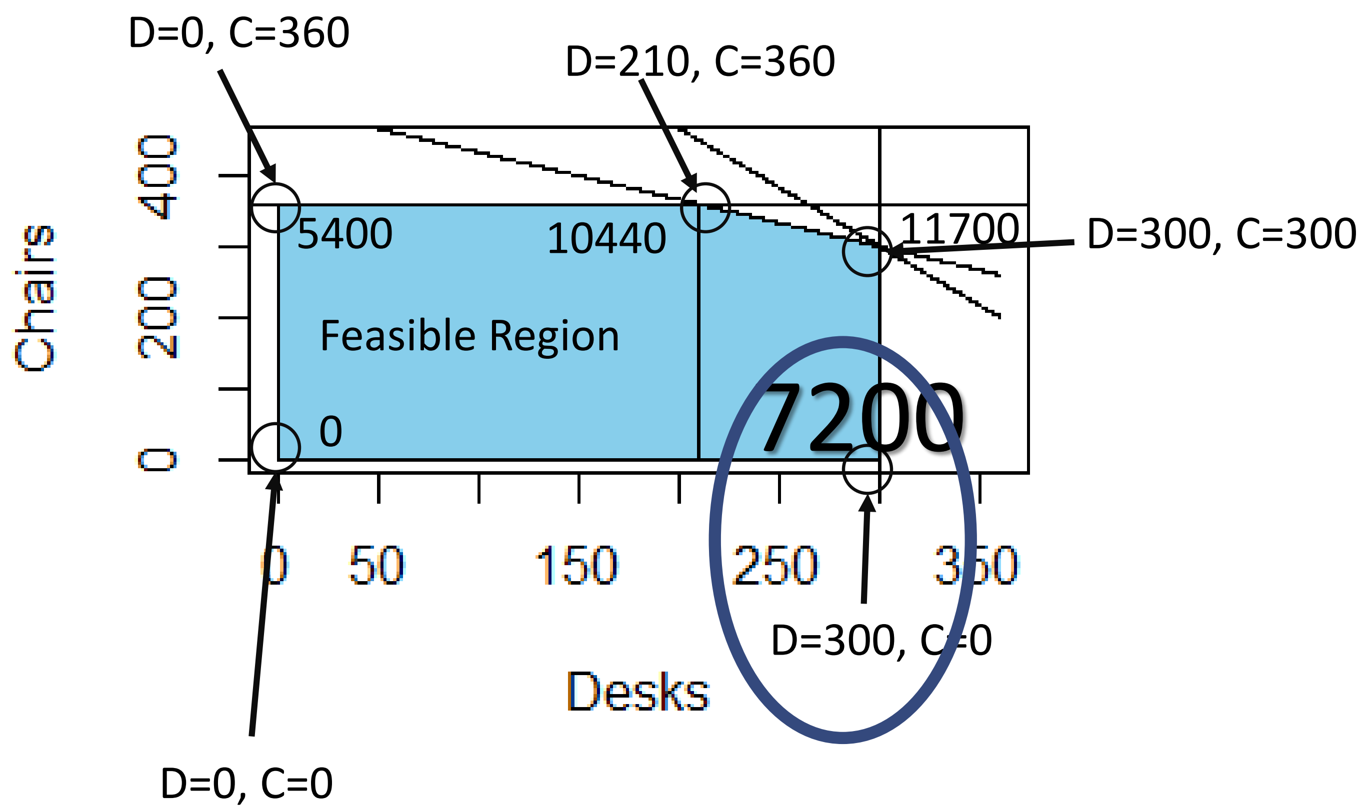
Simplex method



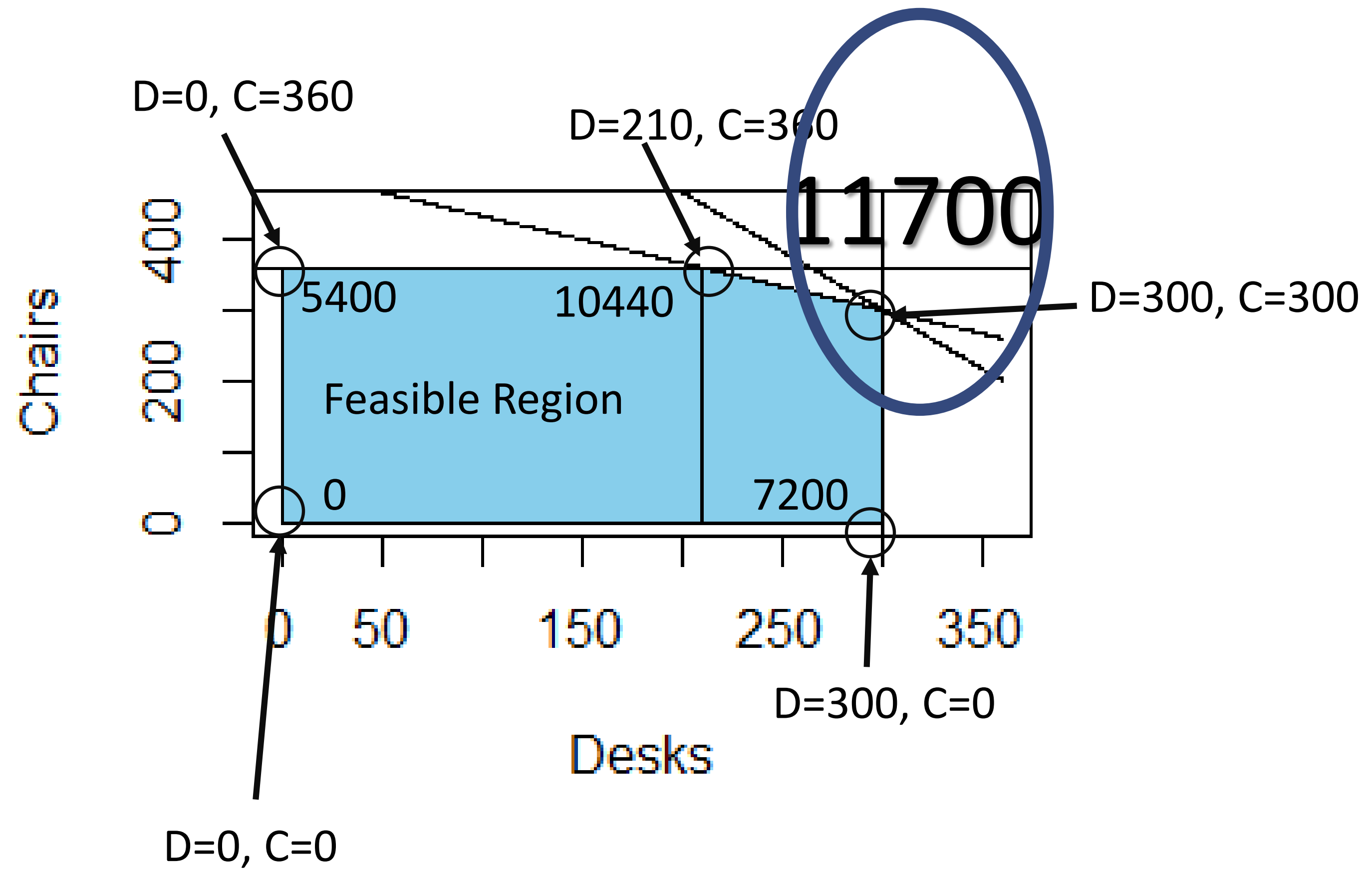
Simplex method



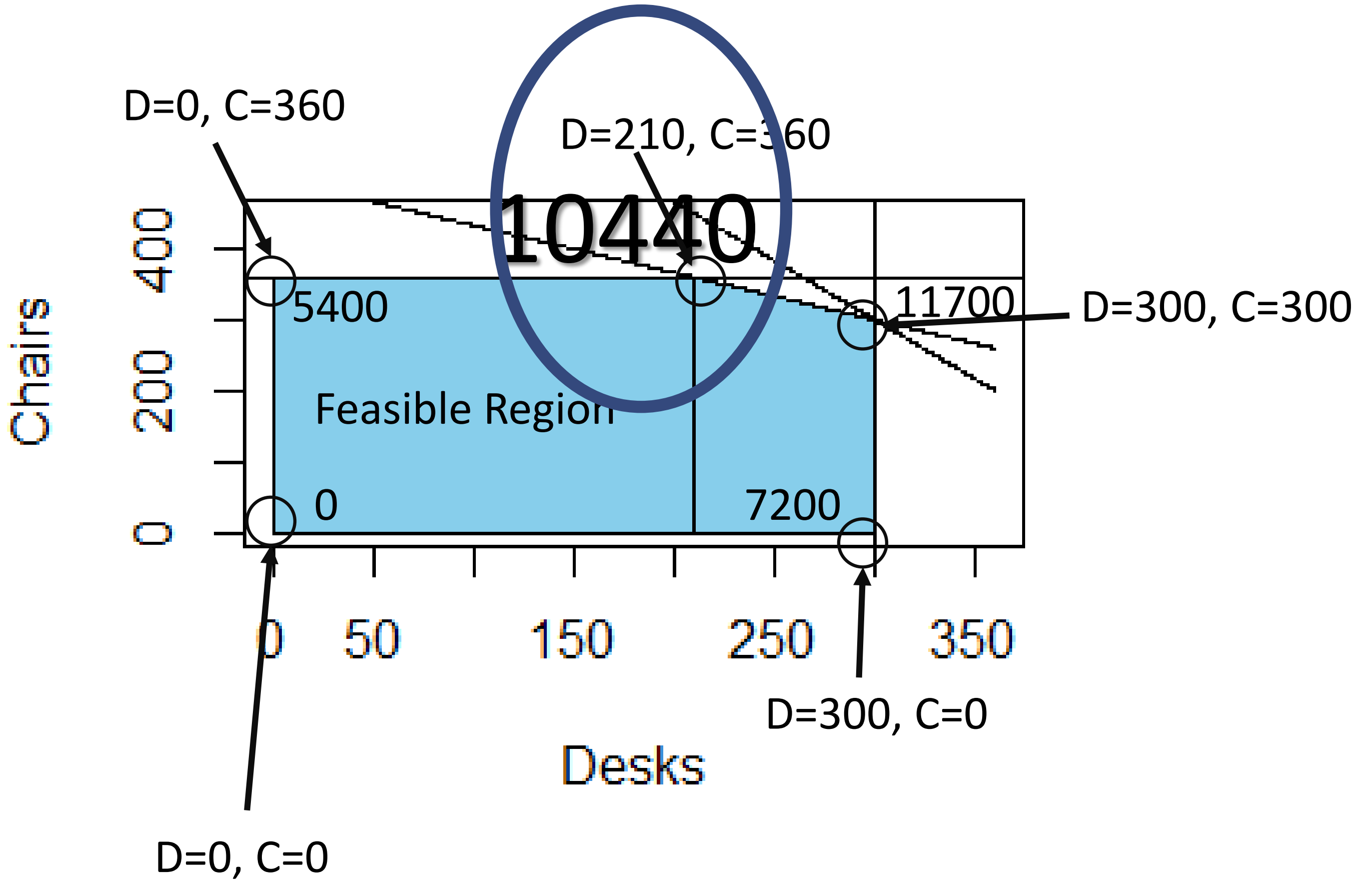
Simplex method



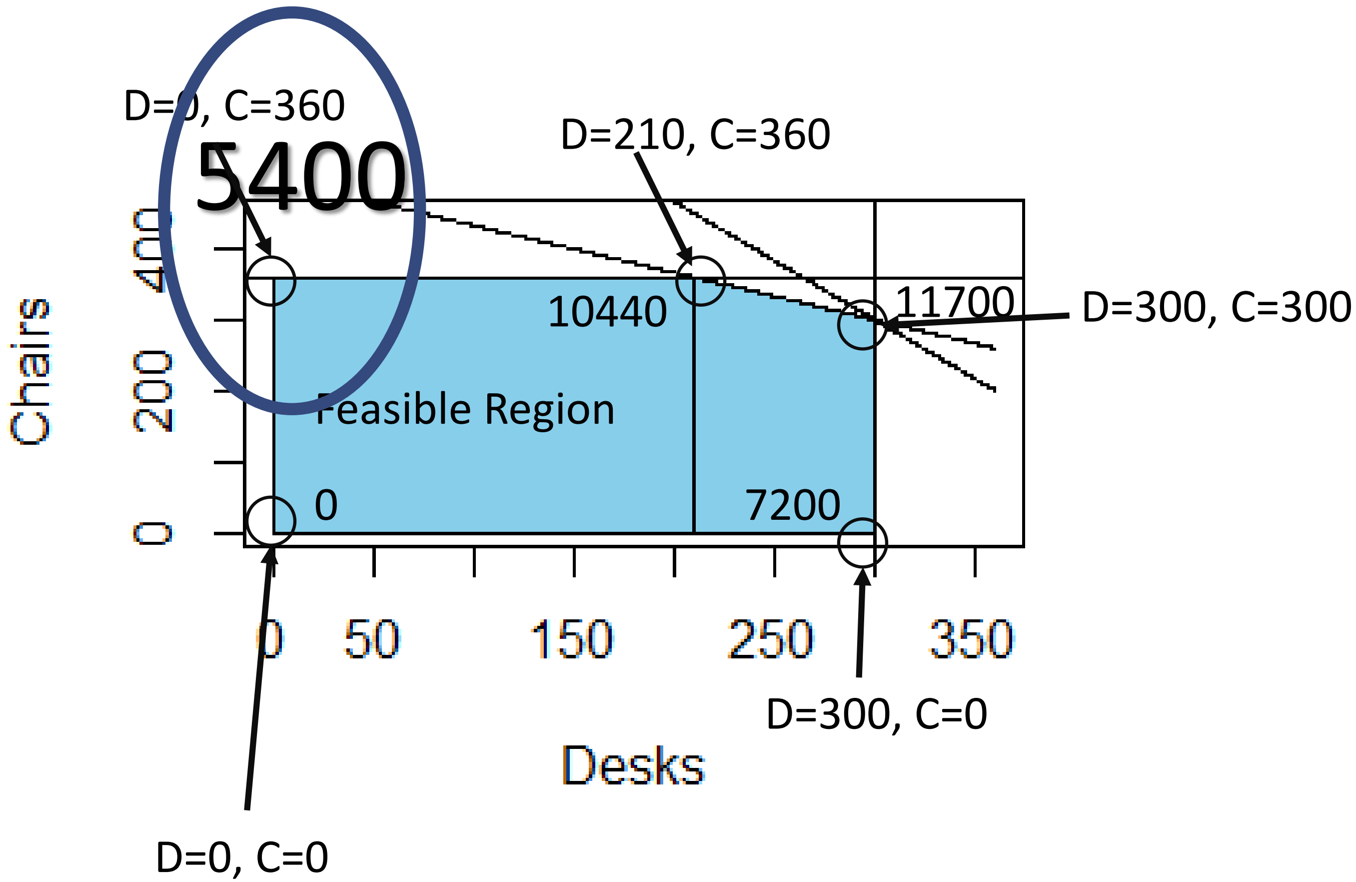
Simplex method



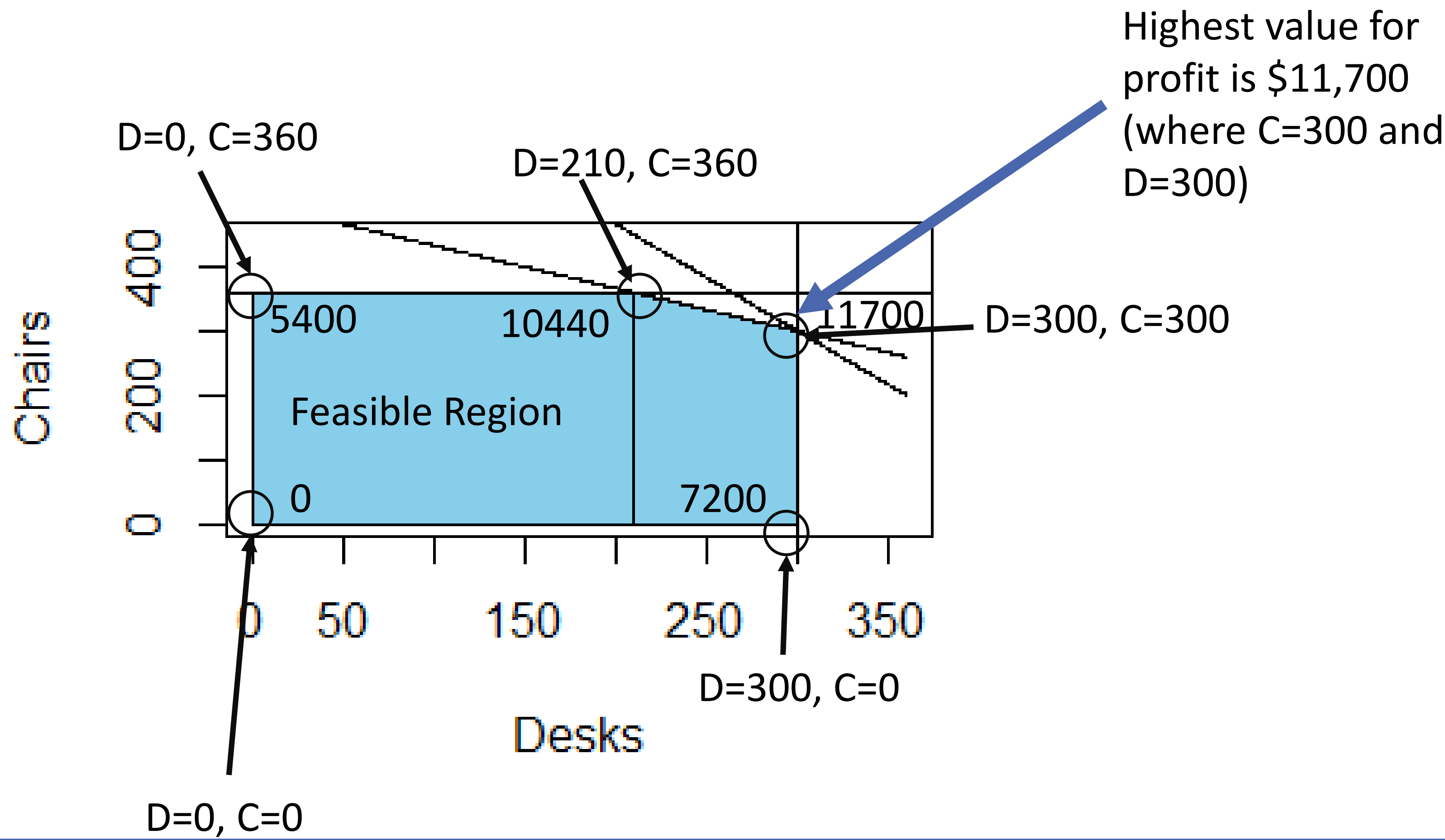
Simplex method



Simplex method



Simplex method



Example

Veerman Furniture Company makes chairs, desks and tables with the following information

	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	

Want to maximize profit

Example:

Objective function:

- MAXIMIZE: $15C + 24D + 18T$

Constraints:

- Fabri: $4C + 6D + 2T \leq 1850$
- Assem: $3C + 5D + 7T \leq 2400$
- Shipp: $3C + 2D + 4T \leq 1500$
- $C \leq 360$
- $D \leq 300$
- $T \leq 100$
- $C \geq 0$, $D \geq 0$, $T \geq 0$

SAS Code

```
proc optmodel;
var Chairs>=0, Desks>=0, Tables>=0;
max Profit = 15*Chairs + 24*Desks + 18*Tables;
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 with lp;
print Chairs Desks Tables;
quit;
```

SAS Code

```
proc optmodel;
var Chairs>=0, Desks>=0, Tables>=0;
max Profit = 15*Chairs + 24*Desks + 18*Tables;
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 with lp;
print Chairs Desks Tables;
quit;
```

Define decision variables

SAS Code

```
proc optmodel;
var Chairs>=0, Desks>=0, Tables>=0;
max Profit = 15*Chairs + 24*Desks + 18*Tables;
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 with lp;
print Chairs Desks Tables;
quit;
```

Define objective function

SAS Code

```
proc optmodel;
var Chairs>=0, Desks>=0, Tables>=0;
max Profit = 15*Chairs + 24*Desks + 18*Tables;
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 with lp;
print Chairs Desks Tables;
quit;
```

Define constraints

SAS Code

```
proc optmodel;
var Chairs>=0, Desks>=0, Tables>=0;
max Profit = 15*Chairs + 24*Desks + 18*Tables;
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 with lp;
print Chairs Desks Tables;
quit;
```

Call the solver

SAS Code

```
proc optmodel;
var Chairs>=0, Desks>=0, Tables>=0;
max Profit = 15*Chairs + 24*Desks + 18*Tables;
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 with lp;
print Chairs Desks Tables;
quit;
```

Print the results

Output

The SAS System

The OPTMODEL Procedure

Problem Summary	
Objective Sense	Maximization
Objective Function	Profit
Objective Type	Linear
Number of Variables	3
Bounded Above	0
Bounded Below	3
Bounded Below and Above	0
Free	0
Fixed	0
Number of Constraints	6
Linear LE (<=)	6

Output cont

Chairs	Desks	Tables
0	275	100

The OPTMODEL Procedure

Solution Summary	
Solver	LP
Algorithm	Dual Simplex
Objective Function	Profit
Solution Status	Optimal
Objective Value	8400
Primal Infeasibility	0
Dual Infeasibility	0
Bound Infeasibility	0
Iterations	2
Presolve Time	0.00
Solution Time	0.00

Gurobi in Python

```
from gurobipy import *

# Create a new model
m = Model(“Veerman")

# Create variables
c = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Chair")
d = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Desk")
t = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Table")

# Set objective
m.setObjective(15*c+24*d+18*t, GRB.MAXIMIZE)
```

```
# Add constraints
m.addConstr(4*c + 6*d + 2*t <=
1850, "c0")
m.addConstr(3*c + 5*d + 7*t <=
2400, "c1")
m.addConstr(3*c + 2*d + 4*t <=
1500, "c2")

m.addConstr(c <= 360, "c3")
m.addConstr(d <= 300, "c4")
m.addConstr(t <= 100, "c5")

m.optimize()

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

Gurobi in Python

```
from gurobipy import *

# Create a new model
m = Model("Veerman")
```

DEFINE MODEL

```
# Create variables
c = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Chair")
d = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Desk")
t = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Table")

# Set objective
m.setObjective(15*c+24*d+18*t, GRB.MAXIMIZE)
```

```
# Add constraints
m.addConstr(4*c + 6*d + 2*t <=
1850, "c0")
m.addConstr(3*c + 5*d + 7*t <=
2400, "c1")
m.addConstr(3*c + 2*d + 4*t <=
1500, "c2")

m.addConstr(c <= 360, "c3")
m.addConstr(d <= 300, "c4")
m.addConstr(t <= 100, "c5")

m.optimize()

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

Gurobi in Python

```
from gurobipy import *

# Create a new model
m = Model("Veerman")

# Create variables
c = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Chair")
d = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Desk")
t = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Table")

# Set objective
m.setObjective(15*c+24*d+18*t, GRB.MAXIMIZE)
```

```
# Add constraints
m.addConstr(4*c + 6*d + 2*t <=
1850, "c0")
m.addConstr(3*c + 5*d + 7*t <=
2400, "c1")
m.addConstr(3*c + 2*d + 4*t <=
1500, "c2")
```

DEFINE DECISION VARIABLES

```
m.addConstr(c <= 360, "c3")
m.addConstr(d <= 300, "c4")
m.addConstr(t <= 100, "c5")

m.optimize()

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

Gurobi in Python

```
from gurobipy import *

# Create a new model
m = Model("Veerman")

# Create variables
c = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Chair")
d = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Desk")
t = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Table")

# Set objective
m.setObjective(15*c+24*d+18*t, GRB.MAXIMIZE)
```

```
# Add constraints
m.addConstr(4*c + 6*d + 2*t <=
1850, "c0")
m.addConstr(3*c + 5*d + 7*t <=
2400, "c1")
m.addConstr(3*c + 2*d + 4*t <=
1500, "c2")

m.addConstr(c <= 360, "c3")
m.addConstr(d <= 300, "c4")
m.addConstr(t <= 100, "c5")

m.optimize()
```

DEFINE OBJECTIVE FUNCTION

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

Gurobi in Python

```
from gurobipy import *

# Create a new model
m = Model(“Veerman")

# Create variables
c = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Chair")
d = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Desk")
t = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Table")

# Set objective
m.setObjective(15*c+24*d+18*t, GRB.MAXIMIZE)
```

```
# Add constraints
m.addConstr(4*c + 6*d + 2*t <=
1850, "c0")
m.addConstr(3*c + 5*d + 7*t <=
2400, "c1")
m.addConstr(3*c + 2*d + 4*t <=
1500, "c2")

m.addConstr(c <= 360, "c3")
m.addConstr(d <= 300, "c4")
m.addConstr(t <= 100, "c5")

m.optimize()
```

DEFINE CONSTRAINTS

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

Gurobi in Python

```
from gurobipy import *

# Create a new model
m = Model("Veerman")

# Create variables
c = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Chair")
d = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Desk")
t = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Table")

# Set objective
m.setObjective(15*c+24*d+18*t, GRB.MAXIMIZE)
```

```
# Add constraints
m.addConstr(4*c + 6*d + 2*t <=
1850, "c0")
m.addConstr(3*c + 5*d + 7*t <=
2400, "c1")
m.addConstr(3*c + 2*d + 4*t <=
1500, "c2")

m.addConstr(c <= 360, "c3")
m.addConstr(d <= 300, "c4")
m.addConstr(t <= 100, "c5")

m.optimize()
```

TELL IT TO OPTIMIZE

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

Gurobi in Python

```
from gurobipy import *

# Create a new model
m = Model(“Veerman")

# Create variables
c = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Chair")
d = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Desk")
t = m.addVar(vtype=GRB.CONTINUOUS, lb=0,
name="Table")

# Set objective
m.setObjective(15*c+24*d+18*t, GRB.MAXIMIZE)
```

```
# Add constraints
m.addConstr(4*c + 6*d + 2*t <=
1850, "c0")
m.addConstr(3*c + 5*d + 7*t <=
2400, "c1")
m.addConstr(3*c + 2*d + 4*t <=
1500, "c2")

m.addConstr(c <= 360, "c3")
m.addConstr(d <= 300, "c4")
m.addConstr(t <= 100, "c5")

m.optimize()
```

PRINT RESULTS

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

Output from Gurobi (Python)

Solved in 1 iterations and 0.02 seconds
Optimal objective 8.400000000e+03
Chair 0
Desk 275
Table 100
Obj: 8400

Example (recall..):

Objective function:

- MAXIMIZE: $15C + 24D + 18T$

Constraints:

- Fabri: $4C + 6D + 2T \leq 1850$
- Assem: $3C + 5D + 7T \leq 2400$
- Shipp: $3C + 2D + 4T \leq 1500$
- $C \leq 360$
- $D \leq 300$
- $T \leq 100$
- $C \geq 0$, $D \geq 0$, $T \geq 0$

Gurobi in R

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

Gurobi in R

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

Need to define all parts of optimization in a ‘list’ (we will call it ‘model’)

Gurobi in R

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

The objective function..only put in the coefficients…be sure to note order and MUST keep same order throughout!!

Gurobi in R

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

This is where you define if you want to maximize or minimize

Gurobi in R

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

These are the values on the right hand side of the constraints (parameters)

Gurobi in R

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

The inequalities for each line of the constraints (the length of the rhs vector should equal length of this vector)

Gurobi in R

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

This indicates decision variables are continuous

Gurobi in R

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

These are the coefficients for the left hand side of the constraints….

Writing out ‘left-hand’ side

model$A <-
matrix(c(4,6,2,3,5,7,3,2,4,1,0,0,0,1,0,0,0,1),nrow=6,byrow=T)

4C + 6D +2T

3C + 5D +7T

3C + 2D +4T

C

D

T

byrow=T tells R that the numbers should be read ‘by rows’

Gurobi in R

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

Runs the model

Gurobi in R

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

Print the output (f.names is just to make it look nicer and to label the output)

Output from Gurobi (R)

```
print(result$status)
[1] "OPTIMAL"
> f.names=c('Chairs','Desks','Tables')
> names(result$x)=f.names
> print(result$x)
Chairs Desks Tables 0 275 100
> print(result$objval) [1] 8400
```