Christopher Lum
lum@uw.edu

Lecture 10h
Linear Progamming

Outline

-Linear Program
-Simplex Algorithm

Linear Program

Recall the general form of a constrained optimization problem is given as

$$(\wp) \text{ minimize } f(x) \qquad \textbf{(Eq.1)}$$

such that $f_i(x) \leq 0 \qquad i = 1, 2, \ldots, m \qquad$ (m inequality constraints)
$f_i(x) = 0 \qquad i = m+1, m+2, \ldots, m+p$ (p equality constraints)

In general, the cost function, $f(x)$, and the constraint functions, $f_i(x)$, are nonlinear.

If the cost function and constraints are all linear, then the optimization problem is called a **linear program (LP)**. In other words, the cost function takes the form of

$$f(x) = a_0 + a_1 x_1 + a_2 x_2 + \ldots + a_n x_n$$

And all the constraints have the form of

$$f_i(x) = b_{i,0} + b_{i,1} x_1 + b_{i,2} x_2 + \ldots + b_{i,n} x_n \qquad i = 1, 2, \ldots, m, m+1, \ldots, m+p$$

Note that we can write this in matrix form as

Objective Function

$$f(x) = c^T x \qquad \textbf{(Eq.1)}$$

where $\quad c^T = \begin{pmatrix} a_1 & a_2 & \ldots & a_n \end{pmatrix}$ (note a_0 can be omitted as it only provides a constant offset to the cost functino)

$$x = \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix}$$

Inequality constraints

$$\begin{pmatrix} b_{1,0} \\ b_{2,0} \\ \ldots \\ b_{m,0} \end{pmatrix} + \begin{pmatrix} b_{1,1} & b_{1,2} & \ldots & b_{1,n} \\ b_{2,1} & b_{2,2} & \ldots & b_{2,n} \\ \ldots & \ldots & \ldots & \ldots \\ b_{m,1} & b_{m,2} & \ldots & b_{m,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix} \leq \begin{pmatrix} 0 \\ 0 \\ \ldots \\ 0 \end{pmatrix}$$

$$b + A\,x \leq 0$$

Equality Constraints

$$\begin{pmatrix} b_{m+1,0} \\ b_{m+2,0} \\ \ldots \\ b_{m+p,0} \end{pmatrix} + \begin{pmatrix} b_{m+1,1} & b_{m+1,2} & \ldots & b_{m+1,n} \\ b_{m+2,1} & b_{m+2,2} & \ldots & b_{m+2,n} \\ \ldots & \ldots & \ldots & \ldots \\ b_{m+p,1} & b_{m+p,2} & \ldots & b_{m+p,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ \ldots \\ 0 \end{pmatrix}$$

$$b_{\text{eq}} + A_{\text{eq}}\,x = 0$$

Note that the objective function and constraints are technically affine functions, but this falls under the definition of a linear program.

Example: Simple Linear Program

Reference: The following example is taken from the excellent article, “An Introduction to Linear Programming and the Simplex Algorithm” by Spyros Reveliotis.

<u>Decision Vector</u>

Consider the scenario where we have a factory which is in charge of producing two types of products, P_1 and P_2. A product needs to be processed by two separate workstations, W_1 and W_2, before being completed (for example, one assembles and the other paints the product).

The market value of a unit of P_1 is \$200 and a unit of P_2 is \$400. Assuming the company can sell everything it produces and there are no overhead associated with switching production of different products, how much of P_1 and P_2 should the factory product?

We see that the decision vector is

$$x = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} \text{units of } P_1 \text{ to produce daily} \\ \text{units of } P_2 \text{ to produce daily} \end{pmatrix}$$

<u>Cost Function</u>

Therefore, we can define a reward function (aka objective function) which is the revenue generated by the factory

$g(x) = 200\, x_1 + 400\, x_2$

At this point, we can easily formulate this as a optimization problem of attempting to maximize the revenue

$(\tilde{\wp})$ maximize $g(x)$

Note that we can turn this into our standard minimization problem by defining the cost function to be the negative of the reward function.

$(\wp)$ minimize $f(x) = -g(x) = -200\, x_1 - 400\, x_2$ ***(Eq.E.1)***

```
In[3]:= f[x1_, x2_] = -200 x1 - 400 x2;
```

At this point we have an unconstrained linear program and we see that the optimal thing to do would be to have $x_1 = +\infty$ and $x_2 = +\infty$ (produce an infinite amount of P_1 and P_2). Obviously, this is not realistic so we need to impose some constraints on the system.

<u>Constraints</u>

The machines can only produce a certain amount of each product each day.

If W_1 is dedicated to processing only P_1, it can process 40 units per day. Similarly, if it is dedicated to only processing P_2 it can process 60 units per day. Mathematically, this can be written as

$\frac{1}{40}\, x_1 + \frac{1}{60}\, x_2 \leq 1$ (constraint from machine 1) ***(Eq.E.2)***

If W_2 is dedicated to processing only P_1, it can process 50 units per day. Similarly, if it is dedicated to only processing P_2 it can process 50 units per day. Mathematically, this can be written as

$\frac{1}{50}\, x_1 + \frac{1}{50}\, x_2 \leq 1$ (constraint from machine 2) ***(Eq.E.3)***

In[4]:= **f1[x1_, x2_] =** $\frac{1}{40}$ **x1 +** $\frac{1}{60}$ **x2;**

f2[x1_, x2_] = $\frac{1}{50}$ **x1 +** $\frac{1}{50}$ **x2;**

Finally, the factory cannot produce a negative amount of items, so

$x_i \geq 0$ $\quad i = 1, 2$ ***(Eq.E.4)***

So finally, the formal, constrained optimization problem is

$(\wp)$ minimize $f(x) = -200\, x_1 - 400\, x_2$ ***(Eq.E.5)*** such that $\frac{1}{40} x_1 + \frac{1}{60} x_2 \leq 1$

$\frac{1}{50} x_1 + \frac{1}{50} x_2 \leq 1$

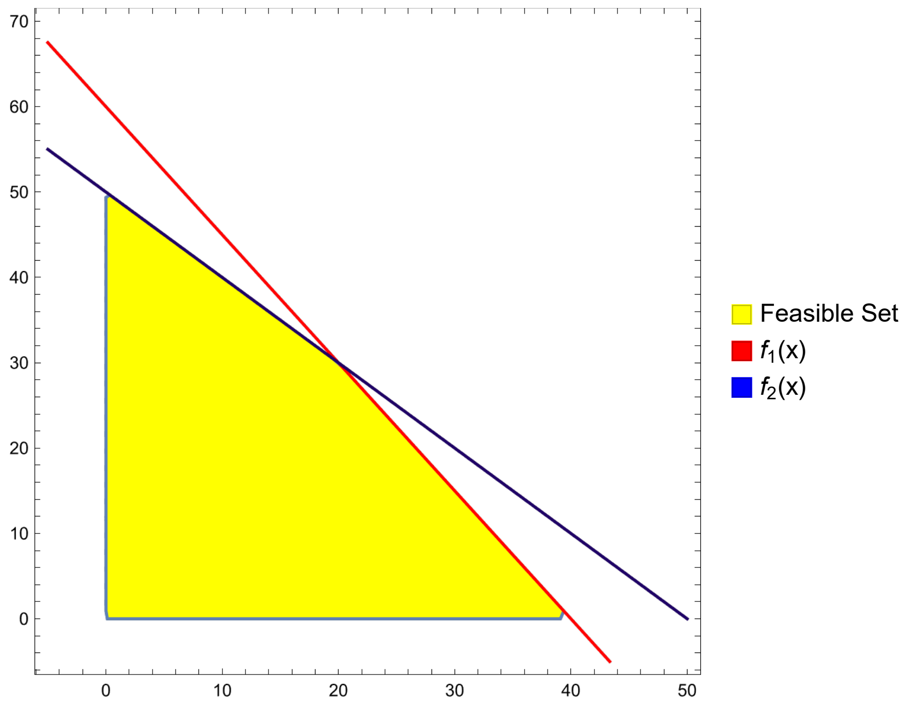
$x_1 \geq 0$

$x_2 \geq 0$

These constraints define the feasible set which we can graphically visualize in 2D

```
In[6]:= x1min = -5;
        x1max = 50;

        x2min = -5;
        x2max = 70;

        (*Plot the individual regions*)
        figureFeasibleSet = Legended[
            Show[
              (*Plot the feasible set*)
              RegionPlot[
                f1[x1, x2] ≤ 1 &&
                  f2[x1, x2] ≤ 1 &&
                  x1 ≥ 0 &&
                  x2 ≥ 0,
                {x1, x1min, x1max}, {x2, x2min, x2max},
                PlotStyle → Yellow
              ],

              (*Plot the individual lines*)
              ContourPlot[f1[x1, x2] == 1,
                {x1, x1min, x1max}, {x2, x2min, x2max},
                ColorFunction → Hue],

              (*Plot the individual lines*)
              ContourPlot[f2[x1, x2] == 1,
                {x1, x1min, x1max}, {x2, x2min, x2max},
                ColorFunction → "BlueGreenYellow"],

              (*Plot options*)
              AxesLabel → {"x₁", "x₂"}
            ],

            SwatchLegend[{Yellow, Red, Blue}, {"Feasible Set", "f₁(x)", "f₂(x)"}]
          ]
```

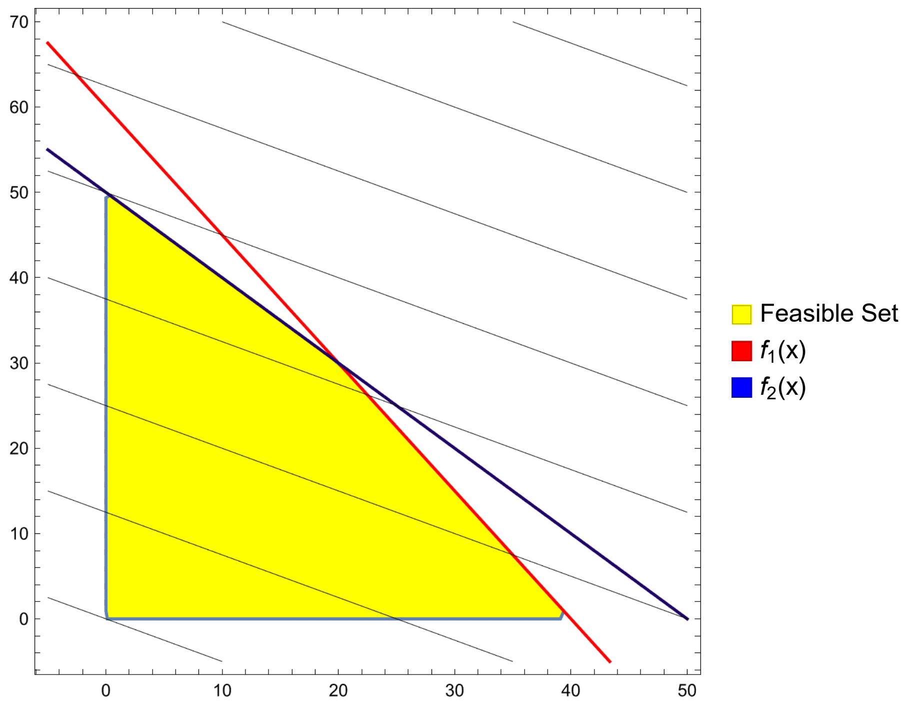
Out[10]=



We can overlay the iso-cost lines on this plot

```
In[11]:= Show[
           (*plot feasible set*)
           figureFeasibleSet,

           (*plot the iso-costs*)
           ContourPlot[f[x1, x2], {x1, x1min, x1max}, {x2, x2min, x2max},
             ContourShading → None]
         ]
```

Out[11]=



From this graphic, we see that the optimal solution is at the point

$$x^* = \begin{pmatrix} 0 \\ 50 \end{pmatrix}$$

The physical interpretation of this result is that we should only produce P_2.

Note that if we change the cost function, the solution changes. For example, supposed that the market changes and the price of a unit of P_1 and P_2 changes as follows

$$P_1 : \$200 \to \$320$$
$$P_2 : \$400 \to \$280$$

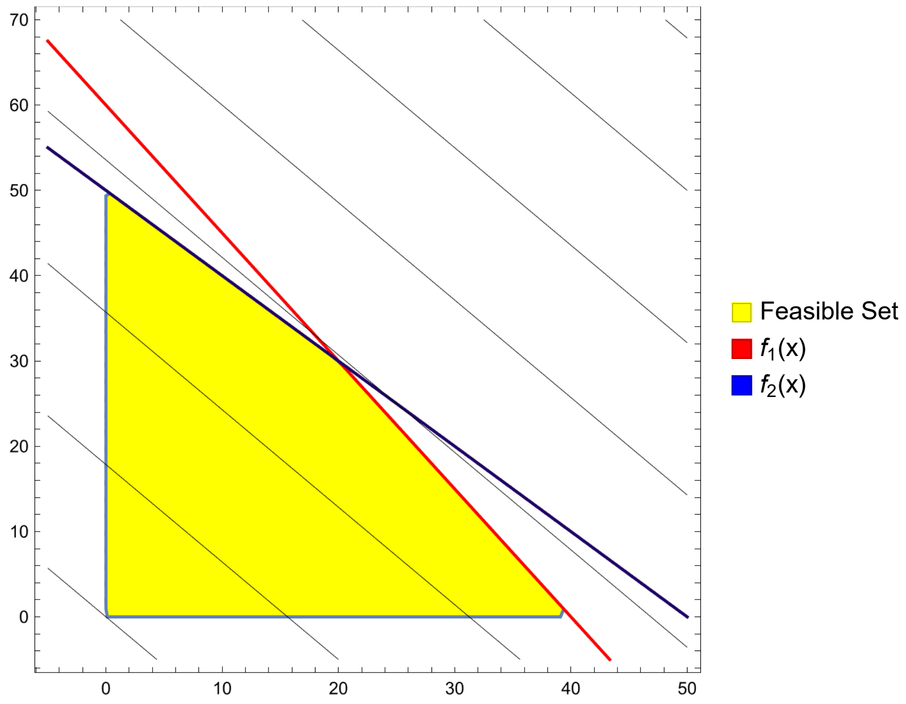
A new cost function would be of the form

$$h(x) = -320\, x_1 - 280\, x_2$$

Using the same iso-cost analysis as before we obtain

In[23]:=
```
h[x1_, x2_] = -320 x1 - 280 x2;

Show[
  (*plot feasible set*)
  figureFeasibleSet,

  (*plot the iso-costs*)
  ContourPlot[h[x1, x2], {x1, x1min, x1max}, {x2, x2min, x2max},
    ContourShading → None]
]
```

Out[24]=



In this case, the optimal solution is

$$x^* = \begin{pmatrix} 20 \\ 30 \end{pmatrix}$$

In both cases, we notice that the optimal solution actually is on a vertex of the feasible set. This observa tion leads to the following theorem

Theorem: Fundamental Theorem of Linear Programming
If a linear program has a bounded optimal solution, then there exists an extreme point of the feasible region which is optimal.

Proof: See http://www2.isye.gatech.edu/~spyros/LP/node16.html.

This observation has led to the development of one of the most important optimization algorithms for solving linear programs, the **simplex method**.

Simplex Algorithm

Reference:
An Introduction to Linear Programming and the Simplex Method
http://www2.isye.gatech.edu/~spyros/LP/LP.html.

The simplex algorithm (aka the simplex method) is a popular algorithm for linear programming. It was listed as one of the top 10 algorithms of the twentieth century.

In order to implement the simplex algorithm, the LP is transformed to "standard form". An LP is said to be in standard form if

i. There are non-negativity constraints for all decision variables
ii. All remaining constraints are expressed as equality constraints ($A x = b$)
iii. The right hand side vector, b, is non-negative

$(\wp)$ minimize $f(x) = c \cdot x$ ***(Eq.1)***

such that $A x = b$

$x_i \geq 0$

$b_k \geq 0 \quad \forall k$

Note that there depending on the author, the name of "standard form" may refer to different formulations. The definition above derives from Dimitris Bertsimas and John Tsitsiklis "Introduction to Linear Optimization" (not to be confused with Dimitri P. Bertsekas, "Nonlinear Programming"). Confusingly, both Bertsimas and Bertsekas work in the field of optimization at MIT, and both have co-authored a book with John Tsitsiklis (Introduction to Probability).

https://ocw.mit.edu/courses/civil-and-environmental-engineering/1-224j-carrier-systems-fall-2003/lecture-notes/lecture2_3.pdf
https://ocw.mit.edu/courses/sloan-school-of-management/15-053-optimization-methods-in-management-science-spring-2013/tutorials/MIT15_053S13_tut06.pdf

Others authors may refer to a standard form LP to be of othre forms

Alternate Standard Form 1

$(\wp)$ maximize $f(x) = c \cdot x$ ***(Eq.1)*** such that $A x \leq b$

$x_i \geq 0$

https://www.matem.unam.mx/~omar/math340/std-form.html
https://sites.math.washington.edu/~burke/crs/407/lectures/L4-lp_standard_form.pdf

Every LP can be transformed into standard form using slack variables and other techniques.

Example: Slack Variables with a 1D Linear Program

Consider the cost associated with driving your car. The decision vector is defined as how much fuel to expend on the trip

$x = (x_1)$ = gallons of fuel burned

So the optimization problem in non-standard form is

$$(\wp_{\text{non-standard}}) \text{ minimize } f(x) = c\, x_1$$

such that $x_1 \leq 10$ (car's maximum fuel capacity)

$x_1 \geq 0$ (cannot have burn negative fuel)

Here, we can consider c to be the price per gallon of gas so the cost function measures the amount of money spent for the trip.

We note that this is not in standard form. Namely, the constraint $x_1 \leq 10$ needs to be transformed to an equality constraint. We can transform this into standard form by introducing a slack variable, s_1.

s_1 = slack variable = amount of unburned fuel (should be non-negative)

So now the constraint of $x_1 \leq 10$ (the car cannot burn more than 10 gallons of fuel), can be written as

$x_1 + s_1 = 10$ (fuel burnt plus fuel unburnt must equal 10)

So the LP in standard form can be stated as

$$(\wp_{\text{standard}}) \text{ minimize } f(x) = c\, x_1$$

such that $x_1 + s_1 = 10$ (car's maximum fuel capacity)

$x_1 \geq 0$ (cannot have burn negative fuel)

$s_1 \geq 0$ (unburnt fuel cannot be negative)

And in this case, the decision vector becomes

$$x = \begin{pmatrix} x_1 \\ s_1 \end{pmatrix} = \begin{pmatrix} \text{original decision variable} \\ \text{slack variable} \end{pmatrix}$$

This process can be extended to high dimension LPs.

Interestingly (and perhaps obviously) the solution to this simple LP is $x^* = 0$ (sit in the driveway doing nothing, thereby expending no money on gas).

Based on the Fundamental Theorem of Linear Programming, if the linear program has a bounded optimal solution, this solution exists on an extreme point of the feasible region. Therefore, one option might be to simply check all of the extreme points. The set of extreme points for an LP with *m* constraints and *n* variables has at most

$$\binom{n}{m} = \frac{n!}{m!\,(n-m)!} \qquad \text{(binomial coefficient of } n \text{ choose } m \text{ combinations)}$$

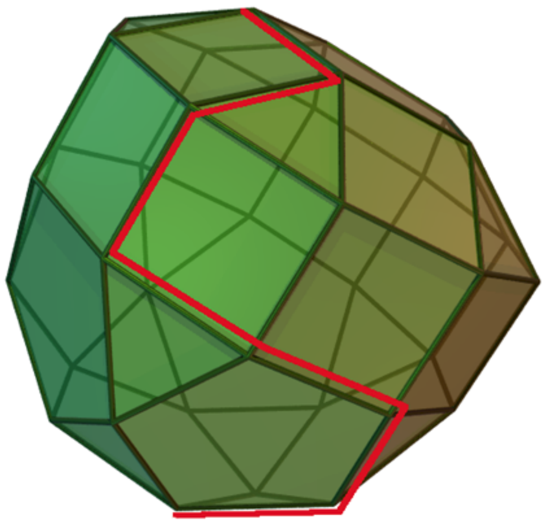
However, this can become large even for moderately sized linear programs. For example, a relatively small LP with 50 variables and 20 constraints can be transformed into a standard LP with at most ~50 trillion extreme points

```
In[14]:= Binomial[50, 20]

Out[14]= 47 129 212 243 960
```

However, it can be shown that for a linear program in standard form, if an extreme point is not a minimum point of the cost function, then there is an edge containing the point so that the cost function is strictly decreasing on the edge moving away from the point. If the edge is finite then one of two things can occur

1. The edge connects to another extreme point where the cost function has a smaller value
2. The cost function is unbounded below on the edge and the linear program has no solution.

The simplex algorithm applies this insight by walking along edges of the polytope to extreme points of increasingly decreasing cost function value until a minimum is obtained.



Matlab provides the 'linprog' function to solve a linear program using a variation of the simplex method. Note that this is part of the ‘Optimization Toolbox’ (which is available on the computers supplied by the AA department via remote desktop). We investigate it’s use in a homework assignment.

For example, code to solve the factory LP is shown below

```
%Solve the linear program
f0 = [-200;-400];        %define the cost function
A = [1/40 1/60;
     1/50 1/50];         %inequality constraints
b = [1;
     1];                 %inequality constraints

Aeq = [];                %equality constraints
beq = [];                %equality constraints

lb = [0;0];              %lower bound for defining X
ub = [];                 %upper bound for defining X

%Optimize using linprog
[xstar,fval,exitflag,output] = ...
    linprog(f0,A,b,Aeq,beq,lb,ub,[],optimset('Display','iter','Diagnostic','on'));

disp('optimal solution')
xstar
```

The output is shown below

```
   Diagnostic Information

Number of variables: 2


 Number of linear inequality constraints:    2
 Number of linear equality constraints:      0
 Number of lower bound constraints:          2
 Number of upper bound constraints:          0

Algorithm selected
   large-scale: interior point



   End diagnostic information

  Residuals:   Primal     Dual     Duality    Total
               Infeas    Infeas      Gap       Rel
               A*x-b    A'*y+z-f    x'*z      Error
  ---------------------------------------------------
  Iter    0:  1.46e+002 9.34e+002 9.80e+004 6.00e+004
  Iter    1:  1.91e+001 4.62e+002 1.20e+003 1.35e+001
  Iter    2:  1.91e+001 4.62e+002 1.98e+003 1.35e+001
  Iter    3:  1.91e+001 4.62e+002 8.93e+002 1.35e+001
  Iter    4:  1.91e+001 4.00e-011 3.06e+005 1.35e+001
  Iter    5:  1.13e-015 5.68e-014 1.90e+004 9.21e-001
  Iter    6:  4.02e-015 1.25e-012 4.99e+004 7.44e-001
  Iter    7:  2.22e-016 4.58e-013 1.57e+003 7.81e-002
  Iter    8:  3.51e-016 3.64e-012 3.90e+000 1.95e-004
  Iter    9:  2.67e-014 3.64e-012 1.95e-003 9.74e-008
  Iter   10:  2.27e-014 2.84e-014 1.95e-010 1.61e-014
Optimization terminated.
optimal solution

xstar =

    0.0000
   50.0000
```

As expected, this gives the optimal location of $x^* = \begin{pmatrix} 0 & 50 \end{pmatrix}^T$

Convex Optimization

See AE512 notes for this discussion.