COMS21103: Linear Programming

Dima Damen
Dima.Damen@bristol.ac.uk

Bristol University, Department of Computer Science
Bristol BS8 1UB, UK

December 3, 2016

What is Linear Programming?

Definition

In general, linear Programming is optimising a linear function subject to a set of linear inequalities

What is Linear Programming?

Definition

In general, linear Programming is optimising a linear function subject to a set of linear inequalities

- Optimising a function is finding the *minimum* or *maximum* value of a function

What is Linear Programming?

Definition

In general, linear Programming is optimising a linear function subject to a set of linear inequalities

- Optimising a function is finding the *minimum* or *maximum* value of a function
- A linear function... e.g. $5 + 3x + 4y$

What is Linear Programming?

Definition

In general, linear Programming is optimising a linear function subject to a set of linear inequalities

- Optimising a function is finding the *minimum* or *maximum* value of a function
- A linear function... e.g. $5 + 3x + 4y$
- A linear inequality... e.g. $5x + 3y \leq 10$, $3 + 5y \geq 4z$

Linear Programming - Example

Linear Programs arise in a variety of practical applications...

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

- What do you want to optimise?

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

- What do you want to optimise?
  - cost

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

- What do you want to optimise?
  - cost
  - *minimise*

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

- What is the linear function?

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

- What is the linear function?
  - Define variables
    x: # of items shipped from B to b
    y: # of items shipped from L to b
    w: # of items shipped from B to l
    z: # of items shipped from L to l

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

- What is the linear function?
  - Define variables
    x: # of items shipped from B to b
    y: # of items shipped from L to b
    w: # of items shipped from B to l
    z: # of items shipped from L to l
  - $f(x, y, w, z) = 5x + 10y + 4w + 8z$

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

- Subject to what linear inequalities?

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

- Subject to what linear inequalities?
  - Required shipments
    $x + y \geq 400$
    $w + z \geq 600$

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

- Subject to what linear inequalities?
  - Required shipments
    $x + y \geq 400$
    $w + z \geq 600$
  - Available Stock
    $x + w \leq 700$
    $y + z \leq 800$

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

- Subject to what linear inequalities?
  - Required shipments
    $x + y \geq 400$
    $w + z \geq 600$
  - Available Stock
    $x + w \leq 700$
    $y + z \leq 800$
  - Non-negative Inequalities
    $x \geq 0$, $y \geq 0$, $w \geq 0$, $z \geq 0$

Linear Programming - Example

Example

A publisher has orders for 400 copies of a certain text from Bristol (b) and 600 copies from Leeds(l). The company has 700 copies in a warehouse in Birmingham (B) and 800 copies in a warehouse in London (L). It costs £5 to ship a text from Birmingham to Bristol, but it costs £4 to ship it to Leeds. It costs £10 to ship a text from London to Bristol, but it costs £8 to ship it from London to Leeds. How many copies should the company ship from each warehouse to Bristol and Leeds to fill the order at the least cost?

Linear Program - Non-standard form

minimise $5x + 10y + 4w + 8z$

subject to
$$x + y \geq 400$$
$$w + z \geq 600$$
$$x + w \leq 700$$
$$y + z \leq 800$$
$$x \geq 0,\ y \geq 0,\ w \geq 0,\ z \geq 0$$

Linear Programming - Example

Linear Program - Matrix Representation

minimise $5x + 10y + 4w + 8z$

subject to

$$
\begin{aligned}
1x + 1y + 0w + 0z &\geq 400 \\
0x + 0y + 1w + 1z &\geq 600 \\
1x + 0y + 1w + 0z &\leq 700 \\
0x + 1y + 0w + 1z &\leq 800 \\
x \geq 0,\ y \geq 0,\ w &\geq 0,\ z \geq 0
\end{aligned}
$$

Linear Programming - Example

Linear Program - Matrix Representation

$$
\begin{array}{ll}
\textbf{minimise} & 5x + 10y + 4w + 8z \\
\textit{subject to} & -1x - 1y + 0w + 0z \leq -400 \\
& 0x + 0y - 1w - 1z \leq -600 \\
& 1x + 0y + 1w + 0z \leq 700 \\
& 0x + 1y + 0w + 1z \leq 800 \\
& x \geq 0,\ y \geq 0,\ w \geq 0,\ z \geq 0
\end{array}
$$

Linear Programming - Example

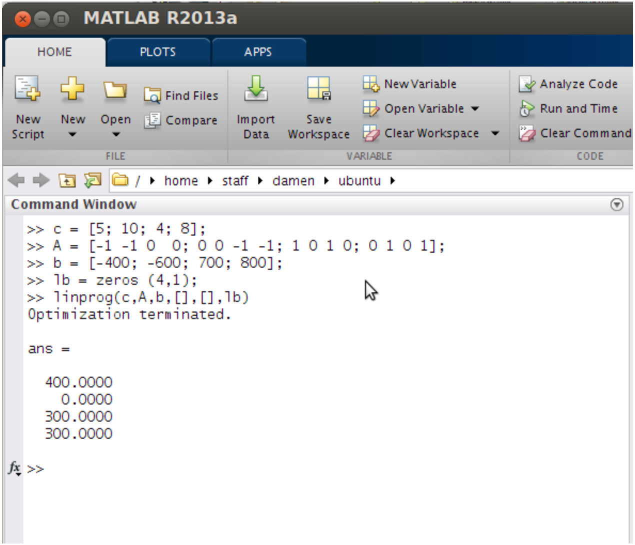
Linear Program - Matrix Representation

$$\mathbf{c} = \begin{bmatrix} 5 \\ 10 \\ 4 \\ 8 \end{bmatrix}, \mathbf{A} = \begin{bmatrix} -1 & -1 & 0 & 0 \\ 0 & 0 & -1 & -1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}, \mathbf{b} = \begin{bmatrix} -400 \\ -600 \\ 700 \\ 800 \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x \\ y \\ w \\ z \end{bmatrix}$$

$$\boxed{\begin{array}{ll} \textbf{minimise} & \mathbf{c}^T\mathbf{x} \\ \textit{subject to} & \mathbf{A}\mathbf{x} \leq \mathbf{b} \\ & \mathbf{x} \geq 0 \end{array}}$$

Linear Programming - Example

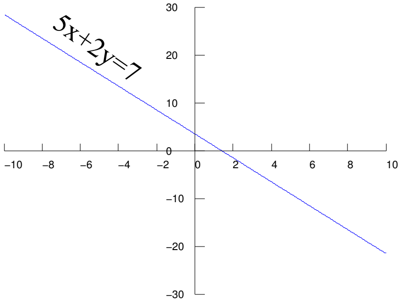
Can be solved using a linear solver



Linear Programming

In two dimensions...

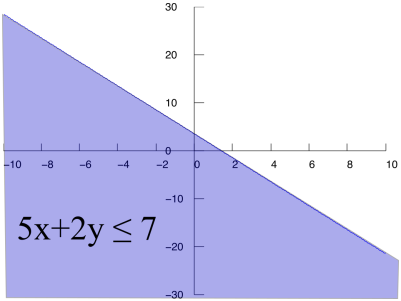
- A linear equation defines a line in space $5x + 2y = 7$



Linear Programming

In two dimensions...

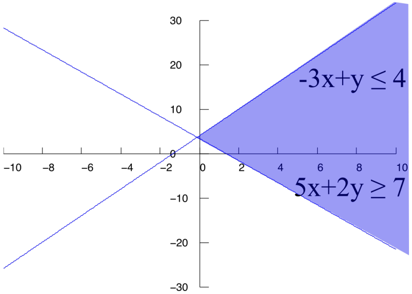
- A linear **inequality** defines a half-space $5x + 2y \leq 7$



Linear Programming

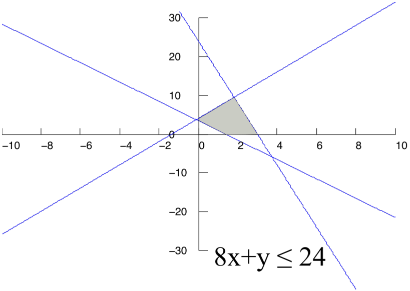
In two dimensions...

- Multiple linear inequalities constrain the space of solutions
- Setting the variables x and y to values that satisfy all constraints results in a **feasible** solution
- Setting the variables x and y to values that fail to satisfy any constraint results in an **infeasible** solution
- The shaded area represents the space of **feasible** solutions



Linear Programming

- When the set of inequalities represent a convex hull in space, the feasible region is said to be **bounded**



Linear Programming

In two dimensions...

- If the linear program has no feasible solution, the linear program is said to be **infeasible**
  e.g.
  $5x + 2y \geq 7$
  $x \leq 0$
  $y \leq 0$

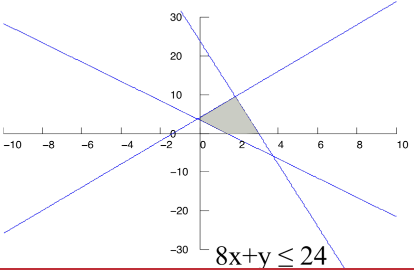
- If a linear program has some feasible solutions but does not have a finite optimal objective value, it is said to be **unbounded**

Linear Programming

- For the linear program

minimise	$x - y$
subject to	$5x + 2y \geq 7$
	$-3x + y \leq 4$
	$8x + y \leq 24$
	$x \geq 0, y \geq 0$

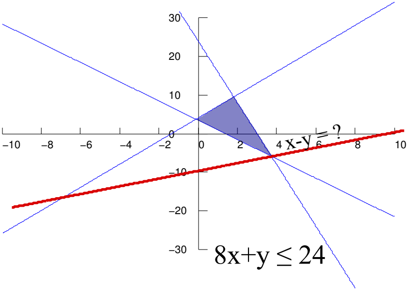
- The search is for a feasible solution that minimises the **objective function** $x - y$



Linear Programming

In two dimensions...

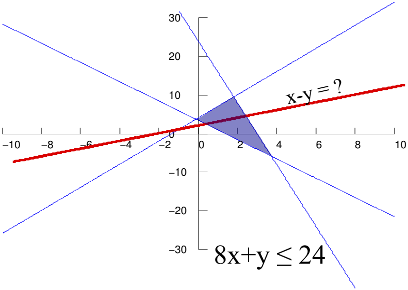
- The objective function takes different values within the feasible region



Linear Programming

In two dimensions...

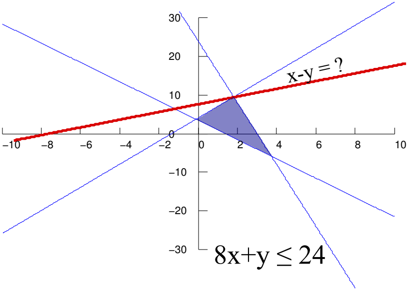
- The objective function takes different values within the feasible region



Linear Programming

In two dimensions...

- The objective function takes different values within the feasible region



Linear Programming

In two dimensions...

- It is no accident that the optimal solution to the linear program occurs at a vertex of the feasible solution.

Linear Programming

In two dimensions...

- It is no accident that the optimal solution to the linear program occurs at a vertex of the feasible solution.
- The optimal value must be at the boundary of the feasible region

Linear Programming

In two dimensions...

- It is no accident that the optimal solution to the linear program occurs at a vertex of the feasible solution.
- The optimal value must be at the boundary of the feasible region
- The intersection is thus either a single vertex or a line segment (that contains two vertices)

Linear Programming

In two dimensions...

- It is no accident that the optimal solution to the linear program occurs at a vertex of the feasible solution.
- The optimal value must be at the boundary of the feasible region
- The intersection is thus either a single vertex or a line segment (that contains two vertices)
- Eureka... calculate the objective function at all vertices!

Linear Programming

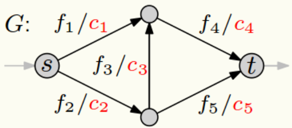
In two dimensions...

- It is no accident that the optimal solution to the linear program occurs at a vertex of the feasible solution.
- The optimal value must be at the boundary of the feasible region
- The intersection is thus either a single vertex or a line segment (that contains two vertices)
- Eureka... calculate the objective function at all vertices!
- But... we cannot easily graph linear programs in 3+ dimensions

Max-Flow as a Linear Program

Max-flow problem can be formulated as a linear program

EXAMPLE



Objective function maximise $f_1 + f_2$

Constraints

Ensure the flow is feasible.

$$f_1 \leqslant c_1$$
$$f_2 \leqslant c_2$$
$$f_3 \leqslant c_3$$
$$f_4 \leqslant c_4$$
$$f_5 \leqslant c_5$$

Conservation constraints.

$$f_1 + f_3 - f_4 = 0$$
$$f_2 - f_3 - f_5 = 0$$

$$f_1, f_2, f_3, f_4, f_5 \geqslant 0$$

The Simplex Algorithm

- Takes as input a linear program and returns an optimal solution
- It starts at some vertex and performs a sequence of iterations
- In each iteration, it moves along an edge to a neighbouring vertex whose objective value is smaller than the current vertex
- Terminates when it reaches a local optimum
- As the vertex is a convex hull, the local optimum is actually a global optimum

Simplex Algorithm - Standard Form

The standard form to solve a simplex algorithm is:

$$\text{maximise} \quad \sum_{j=1}^{n} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij} x_j \leq b_i \quad \text{for } i = 1, 2, ..m$$

$$x_j \geq 0 \quad \text{for } j = 1, 2, ..n$$

Dima Damen
Dima.Damen@bristol.ac.uk

The Simplex Algorithm - Standard Form

A linear function is in non-standard form for the Simplex algorithm if

- The objective function is a minimisation rather than a maximisation
- There might be variables without nonnegativity constraints
- There might be equality rather than inequality constraints
- There might be inequality constraints with an opposite sign

The Simplex Algorithm - Standard Form

To convert to a standard form:

- If the objective function is a minimisation $\rightarrow$ negate the coefficients e.g.

  minimise $-2x + 3y$

  becomes

  maximise $2x - 3y$

The Simplex Algorithm - Standard Form

To convert to a standard form:

- If a variable y does not have a non-negativity constraint $\rightarrow$ change y to $y_1 - y_2$ and add $y_1 \geq 0$, $y_2 \geq 0$
  e.g.

$$\boxed{\begin{array}{ll} \text{maximise} & 2x - 3y \\ \text{subject to} & x + y \leq 7 \\ & x - y \leq 4 \\ & x \geq 0 \end{array}} \rightarrow \boxed{\begin{array}{ll} \text{maximise} & 2x - 3y_1 + 3y_2 \\ \text{subject to} & x + y_1 - y_2 \leq 7 \\ & x - y_1 + y_2 \leq 4 \\ & x, y_1, y_2 \geq 0 \end{array}}$$

The Simplex Algorithm - Standard Form

To convert to a standard form:

- If equality constraint exists $\rightarrow$ replace by two inequalities $\leq b$ and $\geq b$ e.g.

$$\boxed{\begin{array}{ll} \text{maximise} & 2x - 3y \\ \text{subject to} & x + y = 7 \\ & .. \end{array}} \rightarrow \boxed{\begin{array}{ll} \text{maximise} & 2x - 3y \\ \text{subject to} & x + y \leq 7 \\ & x + y \geq 7 \\ & .. \end{array}}$$

The Simplex Algorithm - Standard Form

To convert to a standard form:

- If inequality constraint needs to change sign $\rightarrow$ negate
  e.g.

$$\boxed{\begin{array}{ll} \text{maximise} & 2x - 3y \\ \text{subject to} & x + y \geq 7 \\ & .. \end{array}} \rightarrow \boxed{\begin{array}{ll} \text{maximise} & 2x - 3y \\ \text{subject to} & -x - y \leq -7 \\ & .. \end{array}}$$

The Simplex Algorithm - Standard Form

e.g. convert the following linear program into standard form:

$$\begin{array}{ll} \text{minimise} & 2x_1 + 7x_2 + x_3 \\ \text{subject to} & x_1 - x_3 = 7 \\ & 3x_1 + x_2 \geq 24 \\ & x_2 \geq 0 \\ & x_3 \leq 0 \end{array}$$

The Simplex Algorithm - Slack Form

After being in the standard form, the Simplex algorithm puts the linear program in the slack form.

$$\boxed{\begin{array}{lcll} z & = & \sum_{j=1}^{n} c_j x_j & \\ & & & \\ x_{n+i} & = & b_i - \sum_{j=1}^{n} a_{ij} x_j & \text{for } i = 1, 2, .., m \\ & & x_i \geq 0 & \text{for } i = 1, 2, .., n+m \end{array}}$$

The Simplex Algorithm - Slack Form

- For each inequality $\sum_{j=1}^{n} a_{ij} x_j \leq b_i$
- Introduce a new variable s (called the **slack variable** because it measures the difference between the left and the right hand sides of the equation.)
- Rewrite the inequality $s = b_i - \sum_{j=1}^{n} a_{ij} x_j$
- Add a non-negativity constraint $s \geq 0$

The Simplex Algorithm - Slack Form

e.g. convert the following linear program from standard form to slack form:

$$\begin{array}{ll} \text{maximise} & 2x_1 - 3x_2 + 3x_3 \\ \text{subject to} & x_1 + x_2 - x_3 \leq 7 \\ & -x_1 - x_2 + x_3 \leq -7 \\ & x_1 - 2x_2 + 2x_3 \leq 4 \\ & x_1, x_2, x_3 \geq 0 \end{array}$$

The Simplex Algorithm - Slack Form

- Step 1: add the slack variables
- Step 2: Replace the objective function value by z

$$
\boxed{
\begin{array}{rl}
z = & 2x_1 - 3x_2 + 3x_3 \\
x_4 = & 7 - x_1 - x_2 + x_3 \\
x_5 = & -7 - +x_1 + x_2 - x_3 \\
x_6 = & 4 - x_1 + 2x_2 - 2x_3 \\
 & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{array}
}
$$

The Simplex Algorithm - Slack Form

- Step 1: add the slack variables
- Step 2: Replace the objective function value by z

$$
\boxed{
\begin{array}{rl}
z = & 2x_1 - 3x_2 + 3x_3 \\
x_4 = & 7 - x_1 - x_2 + x_3 \\
x_5 = & -7 - +x_1 + x_2 - x_3 \\
x_6 = & 4 - x_1 + 2x_2 - 2x_3 \\
 & x_1, x_2, x_3, x_4, x_5, x_6 \geq 0
\end{array}
}
$$

- The variables on the left hand size of equalities are called **basic variables**
- The variables on the right hand size of equalities are called **nonbasic variables**

Dima Damen
Dima.Damen@bristol.ac.uk

The Simplex Algorithm - Basic Solution

- A feasible solution can be found by setting all nonbasic variables (right-hand side variables) to 0
- This is a feasible solution and is a *vertex* in the convex hull because of the non-negativity constraint.
- For the example below, $\mathbf{x} = (0, 0, 0, 30, 24, 36)$ and $z = 0$
- This is referred to as a basic feasible solution

$$
\begin{array}{llllllllll}
z & = & & & 3x_1 & + & x_2 & + & 2x_3 & (1) \\
x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 & (2) \\
x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 & (3) \\
x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- At each iteration in the simplex algorithm, we select a non-basic variable x_i
- This chosen variable should have a positive coefficient in the objective function
- So that increasing its value would increase the objective function
- Let's choose x_1

$$
\begin{array}{lllllllll}
z & = & & & 3x_1 & + & x_2 & + & 2x_3 & (1) \\
x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 & (2) \\
x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 & (3) \\
x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- We try to increase x_1 as much as possible without increasing any non-negativity constraint

$$
\begin{array}{lllllllll}
z & = & & & 3x_1 & + & x_2 & + & 2x_3 & (1) \\
x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 & (2) \\
x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 & (3) \\
x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- We try to increase x_1 as much as possible without increasing any non-negativity constraint
  - In (2), x_1 could be set to 30 max
  - In (3), x_1 could be set to 12 max
  - In (4), x_1 could be set to 9 max

$$
\boxed{
\begin{array}{lllllllllll}
z & = & & & 3x_1 & + & x_2 & + & 2x_3 & (1) \\
x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 & (2) \\
x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 & (3) \\
x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 & (4)
\end{array}
}
$$

The Simplex Algorithm - Iteration

- We try to increase x_1 as much as possible without increasing any non-negativity constraint
  - In (2), x_1 could be set to 30 max
  - In (3), x_1 could be set to 12 max
  - In (4), x_1 could be set to 9 max
- Equation (4) is the tightest constraint
- We therefore switch the roles of x_1 and x_6

$$
\boxed{
\begin{array}{lllllllll}
z & = & & & 3x_1 & + & x_2 & + & 2x_3 & (1) \\
x_4 & = & 30 & - & x_1 & - & x_2 & - & 3x_3 & (2) \\
x_5 & = & 24 & - & 2x_1 & - & 2x_2 & - & 5x_3 & (3) \\
x_6 & = & 36 & - & 4x_1 & - & x_2 & - & 2x_3 & (4)
\end{array}
}
$$

The Simplex Algorithm - Iteration

- We re-arrange (4) so $x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}$

The Simplex Algorithm - Iteration

- We re-arrange (4) so $x_1 = 9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}$
- x_1 will be replaced by x_6 on the right-hand side of Equations 1-3

$$\begin{array}{lllllllll}
z & = & & + & 3(9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}) & + & x_2 & + & 2x_3 & (1) \\
x_4 & = & 30 & - & (9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}) & - & x_2 & - & 3x_3 & (2) \\
x_5 & = & 24 & - & 2(9 - \frac{x_2}{4} - \frac{x_3}{2} - \frac{x_6}{4}) & - & 2x_2 & - & 5x_3 & (3) \\
x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} & (4)
\end{array}$$

The Simplex Algorithm - Iteration

$$
\begin{array}{llllllllll}
z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} & (1) \\
x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} & (2) \\
x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} & (3) \\
x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- After this operation the basic variables become (x_1, x_4 and x_5)

$$
\begin{array}{|ccccccccc|}
\hline
z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} & (1) \\
x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} & (2) \\
x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} & (3) \\
x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} & (4) \\
\hline
\end{array}
$$

The Simplex Algorithm - Iteration

- After this operation the basic variables become (x_1, x_4 and x_5)
- The solution just changes to be (9,0,0,21,6,0) and the objective function increases to 27

$$
\begin{array}{lllllllllr}
z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} & (1) \\
x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} & (2) \\
x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} & (3) \\
x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- After this operation the basic variables become (x_1, x_4 and x_5)
- The solution just changes to be (9,0,0,21,6,0) and the objective function increases to 27
- This too is a vertex in the convex hull

$$
\begin{array}{lllllllll}
z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} & (1) \\
x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} & (2) \\
x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} & (3) \\
x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- This operation is known as a **pivot**, which exchanges the positions of one nonbasic variable (called the **entering variable**) and one basic variable (called the **leaving variable**)
- Next we choose another entering variable (we can choose x_2 or x_3 and let's choose x_3)

$$
\begin{array}{lllllllll}
z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} & (1) \\
x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} & (2) \\
x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} & (3) \\
x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- We try to increase x_3 as much as possible without increasing any non-negativity constraint

$$
\boxed{
\begin{array}{lllllllll}
z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} & (1) \\[2ex]
x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} & (2) \\[2ex]
x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} & (3) \\[2ex]
x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} & (4)
\end{array}
}
$$

The Simplex Algorithm - Iteration

- We try to increase x_3 as much as possible without increasing any non-negativity constraint
  - In (2), x_3 could be set to 18 max
  - In (3), x_3 could be set to 8.4 max
  - In (4), x_3 could be set to 1.5 max

$$
\begin{array}{llllllllll}
z & = & 27 & + & \frac{x_2}{4} & + & \frac{x_3}{2} & - & \frac{3x_6}{4} & (1) \\
x_1 & = & 9 & - & \frac{x_2}{4} & - & \frac{x_3}{2} & - & \frac{x_6}{4} & (2) \\
x_4 & = & 21 & - & \frac{3x_2}{4} & - & \frac{5x_3}{2} & + & \frac{x_6}{4} & (3) \\
x_5 & = & 6 & - & \frac{3x_2}{2} & - & 4x_3 & + & \frac{x_6}{2} & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- Equation (4) is the tightest constraint
- We therefore switch the roles of x_3 and x_5

The Simplex Algorithm - Iteration

- Equation (4) is the tightest constraint
- We therefore switch the roles of x_3 and x_5
- $x_3 = \frac{3}{2} - \frac{3x_2}{8} - \frac{x_5}{4} + \frac{x_6}{8}$

z	=	27	+	$\frac{x_2}{4}$	+	$\frac{x_3}{2}$	-	$\frac{3x_6}{4}$	(1)
x_1	=	9	-	$\frac{x_2}{4}$	-	$\frac{x_3}{2}$	-	$\frac{x_6}{4}$	(2)
x_3	=	$\frac{3}{2}$	-	$\frac{3x_2}{8}$	-	$\frac{x_5}{4}$	+	$\frac{x_6}{8}$	(3)
x_4	=	21	-	$\frac{3x_2}{4}$	-	$\frac{5x_3}{2}$	+	$\frac{x_6}{4}$	(4)

The Simplex Algorithm - Iteration

- The current solution is (33/4, 0, 3/2, 69/4, 0, 0) with the objective value 111/4

$$
\begin{array}{ccccccccccl}
z & = & \frac{111}{4} & + & \frac{x_2}{16} & - & \frac{x_5}{8} & - & \frac{11x_6}{16} & & (1) \\
x_1 & = & \frac{33}{4} & - & \frac{x_2}{16} & + & \frac{x_5}{8} & - & \frac{5x_6}{16} & & (2) \\
x_3 & = & \frac{3}{2} & - & \frac{3x_2}{8} & - & \frac{x_5}{4} & + & \frac{x_6}{8} & & (3) \\
x_4 & = & \frac{69}{4} & + & \frac{3x_2}{16} & + & \frac{5x_5}{8} & - & \frac{x_6}{16} & & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- The current solution is (33/4, 0, 3/2, 69/4, 0, 0) with the objective value 111/4
- Next we increase x_2 by substituting it with x_3

$$
\begin{array}{lllllllllr}
z & = & \frac{111}{4} & + & \frac{x_2}{16} & - & \frac{x_5}{8} & - & \frac{11x_6}{16} & (1) \\
x_1 & = & \frac{33}{4} & - & \frac{x_2}{16} & + & \frac{x_5}{8} & - & \frac{5x_6}{16} & (2) \\
x_3 & = & \frac{3}{2} & - & \frac{3x_2}{8} & - & \frac{x_5}{4} & + & \frac{x_6}{8} & (3) \\
x_4 & = & \frac{69}{4} & + & \frac{3x_2}{16} & + & \frac{5x_5}{8} & - & \frac{x_6}{16} & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- The current solution is (8,4, 0, 18, 0, 0) with the objective value 28

$$
\begin{array}{lllllllll}
z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} & (1) \\
x_1 & = & 8 & + & \frac{x_6}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} & (2) \\
x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} & (3) \\
x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- The current solution is (8,4, 0, 18, 0, 0) with the objective value 28
- In the original linear program $x_1 = 8$, $x_2 = 4$ and $x_3 = 0$, with the objective value = 28

$$
\boxed{
\begin{array}{llllllllr}
z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} \quad (1) \\
x_1 & = & 8 & + & \frac{x_6}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} \quad (2) \\
x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} \quad (3) \\
x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & \quad (4)
\end{array}
}
$$

The Simplex Algorithm - Iteration

- The current solution is (8,4, 0, 18, 0, 0) with the objective value 28
- In the original linear program $x_1 = 8$, $x_2 = 4$ and $x_3 = 0$, with the objective value = 28
- The slack variables measure how much slack remains within each inequality

$$
\begin{array}{llllllllll}
z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} & (1) \\
x_1 & = & 8 & + & \frac{x_6}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} & (2) \\
x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} & (3) \\
x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- The current solution is (8,4, 0, 18, 0, 0) with the objective value 28
- In the original linear program $x_1 = 8$, $x_2 = 4$ and $x_3 = 0$, with the objective value = 28
- The slack variables measure how much slack remains within each inequality
- All non-basic variables have a negative coefficient in the objective function

$$
\begin{array}{ccccccccc}
z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} & (1) \\
x_1 & = & 8 & + & \frac{x_6}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} & (2) \\
x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} & (3) \\
x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & & (4)
\end{array}
$$

The Simplex Algorithm - Iteration

- The current solution is (8,4, 0, 18, 0, 0) with the objective value 28
- In the original linear program $x_1 = 8$, $x_2 = 4$ and $x_3 = 0$, with the objective value = 28
- The slack variables measure how much slack remains within each inequality
- All non-basic variables have a negative coefficient in the objective function
- The vertex with the maximum value is reached — terminate

$$
\boxed{
\begin{array}{ccccccccc}
z & = & 28 & - & \frac{x_3}{6} & - & \frac{x_5}{6} & - & \frac{2x_6}{3} & (1) \\[2ex]
x_1 & = & 8 & + & \frac{x_6}{6} & + & \frac{x_5}{6} & - & \frac{x_6}{3} & (2) \\[2ex]
x_2 & = & 4 & - & \frac{8x_3}{3} & - & \frac{2x_5}{3} & + & \frac{x_6}{3} & (3) \\[2ex]
x_4 & = & 18 & - & \frac{x_3}{2} & + & \frac{x_5}{2} & & & (4)
\end{array}
}
$$

The Simplex Algorithm

(N,B,A,b,c,v) = SIMPLEX (A,b,c);

- convert to slack form (N: nonbasic variable, B: basic variable)

The Simplex Algorithm

(N,B,A,b,c,v) = SIMPLEX (A,b,c);
while *some index* $j \in N$ *has* $c_j > 0$ **do**

end

- convert to slack form (N: nonbasic variable, B: basic variable)
- find a nonbasic variable that can increase the objective value

The Simplex Algorithm

(N,B,A,b,c,v) = SIMPLEX (A,b,c);
while *some index* $j \in N$ *has* $c_j > 0$ **do**
  for *each* $i \in B$ **do**
    $\Delta_i = b_i / a_{ij}$;
  end

end

- convert to slack form (N: nonbasic variable, B: basic variable)
- find a nonbasic variable that can increase the objective value

The Simplex Algorithm

(N,B,A,b,c,v) = SIMPLEX (A,b,c);
while *some index* $j \in N$ *has* $c_j > 0$ **do**
  for *each* $i \in B$ **do**
    $\Delta_i = b_i / a_{ij}$;
  end
  choose $l \in B$ that minimises Δ_l;

end

- convert to slack form (N: nonbasic variable, B: basic variable)
- find a nonbasic variable that can increase the objective value
- find the tight basic variable

The Simplex Algorithm

(N,B,A,b,c,v) = SIMPLEX (A,b,c);
while *some index* $j \in N$ *has* $c_j > 0$ **do**
    for *each* $i \in B$ **do**
        $\Delta_i = b_i / a_{ij}$;
    end
    choose $l \in B$ that minimises Δ_l;
    if $\Delta_l ==$ inf **then**
        return "unbounded"
    else
        (N,B,A,b,c,v) = PIVOT(N,B,A,b,c,v,l,e)
    end
end

- convert to slack form (N: nonbasic variable, B: basic variable)
- find a nonbasic variable that can increase the objective value
- find the tight basic variable
- replace the nonbasic with the basic variable

The Simplex Algorithm

(N,B,A,b,c,v) = SIMPLEX (A,b,c);
while *some index* $j \in N$ *has* $c_j > 0$ **do**
  for *each* $i \in B$ **do**
    $\Delta_i = b_i / a_{ij}$;
  end
  choose $l \in B$ that minimises Δ_l;
  if $\Delta_l ==$ inf **then**
    return "unbounded"
  else
    (N,B,A,b,c,v) = PIVOT(N,B,A,b,c,v,l,e)
  end
end
for *i=1..n* **do**
  if $i \in B$ **then**
    $\bar{x}_i = b_i$
  else
    $\bar{x}_i = 0$
  end
end
return $(\bar{x}_1, \bar{x}_2, .., \bar{x}_n)$

- convert to slack form (N: nonbasic variable, B: basic variable)
- find a nonbasic variable that can increase the objective value
- find the tight basic variable
- replace the nonbasic with the basic variable
- find the values of the original variables
- return the values of original variables

Example

Solve the following linear program using the Simplex Algorithm

$$\begin{array}{ll} \text{maximise} & 18x_1 + 12.5x_2 \\ \text{subject to} & x_1 + x_2 \leq 20 \\ & x_1 \leq 12 \\ & x_2 \leq 16 \\ & x_1, x_2 \geq 0 \end{array}$$

Further Reading

- Introduction to Algorithms
  T.H. Cormen, C.E. Leiserson, R.L. Rivest and C. Stein.
  MIT Press/McGraw-Hill, ISBN: 0-262-03293-7.
  - Chapter 27 – Linear Programming
- Youtube Lessons - Class of 2015/2016 - Simplex Algorithm
  - Dylan Cope `https://youtu.be/2hFdmP6fgJQ` [668 views - Nov 2016]
  - Oliver Crow `https://youtu.be/RO5477EKlXE` [249 views - Nov 2016]