Linear Programming
Week 10

COMP 1201 (Algorithmics)

ECS, University of Southampton

13 May 2020

Previously...

Dynamic Programming

- Invented by **Richard E Bellman** in 1953.
- Programming in the sense of scheduling.
- Why is it called "dynamic programming"?

> *"Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities."*
>
> *– R E Bellman, 1984.*

Richard E Bellman

Linear Programming (optimisation with linear functions)

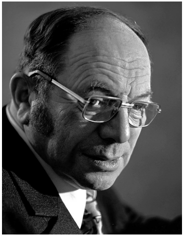
Linear Programming

- Studied by **Leonid Kantorovich** and **Tjalling Koopmans** around 1939.
- A class of **optimisation problems**.
- Optimising **linear functions**, e.g.

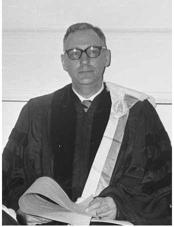
$$3x_1 + x_2 - 2x_3$$

subject to constraints described by linear functions, e.g.

$$x_1 + x_2 + x_3 \leq 0, \; x_1 \geq 0, \; x_2 \geq 0, \; x_3 \geq 0.$$



L Kantorovich



T C Koopmans

Systems of Linear Inequalities

In COMP 1215 (Foundations) you learned how to solve systems of linear equations using *Gaussian elimination*.

$$\begin{aligned} 2x_1 + x_2 - 3x_3 &= 12, \\ 6x_1 - 12x_2 &= 6, \\ x_1 + 3x_2 + x_3 &= 1. \end{aligned}$$

The time complexity of Gaussian elimination is $O(n^3)$ for a linear system with n equations and n variables.

Systems of Linear Inequalities

In COMP 1215 (Foundations) you learned how to solve systems of linear equations using *Gaussian elimination*.

$$
\begin{aligned}
2x_1 + x_2 - 3x_3 &= 12, \\
6x_1 - 12x_2 &= 6, \\
x_1 + 3x_2 + x_3 &= 1.
\end{aligned}
$$

The time complexity of Gaussian elimination is $O(n^3)$ for a linear system with n equations and n variables.

What about solving systems of linear *inequalities*? E.g.

$$
\begin{aligned}
x_1 + x_2 - 2x_3 &\leq 4, \\
36x_1 - 4x_2 &\leq 8, \\
x_1 + x_2 + x_3 &\leq 1.
\end{aligned}
$$

Systems of Linear Inequalities: FME method

Fourier-Motzkin Elimination

- Invented by **Joseph Fourier** in 1827.
- Rediscovered by **Theodore Motzkin** in 1936.
- FME can be used to determine whether a system of linear inequalities is *feasible* (i.e. whether it admits any solutions) and to find feasible points if it is.
- Works by successively *eliminating* variables to produce a (larger) system that has one fewer variable after each iteration.

J-B Joseph Fourier

Fourier-Motzkin Elimination

FME main idea:

1. Select a variable to eliminate from the system, say x_i
2. Rewrite every linear constraint involving x_i as either $x_i \leq U$, or $L \leq x_i$ (these are the only two options, depending on the sign of the coefficient of x_i). L acts as the *lower bound* and U as the *upper bound* on x_i

   This will result in constraints $x_i \leq U_{i1}, \ldots, x_i \leq U_{ik}$, and $L_{i1} \leq x_i, \ldots, L_{ir} \leq x_i$, with the bounds expressed entirely in terms of the remaining variables.
3. Match all the lower bounds on x_i with all the upper bounds, obtaining a system $L_{ij} \leq U_{il}$, where $j = 1, \ldots, r$ and $l = 1, \ldots, k$.
4. Select another variable to eliminate and repeat.

Fourier-Motzkin Elimination (Example)

Fourier-Motzkin Elimination example

Consider the following system of linear inequalities:

$$\begin{aligned} -x_1 + 3x_2 &\geq 2, \\ 2x_1 + x_2 &\leq 1, \\ 5x_1 - 2x_2 &\geq 1, \\ x_1 &\geq 0, \\ x_2 &\geq 0. \end{aligned}$$

We wish to determine feasibility of this system.

We can start by eliminating x_1.

Fourier-Motzkin Elimination (Example)

$$
\begin{aligned}
-x_1 + 3x_2 &\geq 2, \\
2x_1 + x_2 &\leq 1, \\
5x_1 - 2x_2 &\geq 1, \\
x_1 &\geq 0, \\
x_2 &\geq 0.
\end{aligned}
$$

We first re-write all inequalities featuring x_1 into the form $x_1 \leq U_{1i}$ and $L_{1j} \leq x_1$, as appropriate. Start by bringing all x_1 terms to the left-hand side.

Fourier-Motzkin Elimination (Example)

$$
\begin{aligned}
-x_1 &\geq 2 - 3x_2, \\
2x_1 &\leq 1 - x_2, \\
5x_1 &\geq 1 + 2x_2, \\
x_1 &\geq 0, \\
x_2 &\geq 0.
\end{aligned}
$$

This becomes

$$
\begin{aligned}
x_1 &\leq -2 + 3x_2, \\
x_1 &\leq (1/2) - (1/2)x_2, \\
x_1 &\geq (1/5) + (2/5)x_2, \\
x_1 &\geq 0, \\
x_2 &\geq 0.
\end{aligned}
$$

Fourier-Motzkin Elimination (Example)

$$
\begin{aligned}
x_1 &\leq -2 + 3x_2, \\
x_1 &\leq (1/2) - (1/2)x_2, \\
x_1 &\geq (1/5) + (2/5)x_2, \\
x_1 &\geq 0, \\
x_2 &\geq 0.
\end{aligned}
$$

We have two lower and two upper bounds on $x1$; we pair these up:

$$
\begin{aligned}
0 &\leq -2 + 3x_2, \\
(1/5) + (2/5)x_2 &\leq -2 + 3x_2, \\
0 &\leq (1/2) - (1/2)x_2, \\
(1/5) + (2/5)x_2 &\leq (1/2) - (1/2)x_2, \\
x_2 &\geq 0.
\end{aligned}
$$

Fourier-Motzkin Elimination (Example)

$$\begin{aligned}
0 &\leq -2 + 3x_2, \\
(1/5) + (2/5)x_2 &\leq -2 + 3x_2, \\
0 &\leq (1/2) - (1/2)x_2, \\
(1/5) + (2/5)x_2 &\leq (1/2) - (1/2)x_2, \\
x_2 &\geq 0.
\end{aligned}$$

Simplifying, we get:

$$\begin{aligned}
(2/3) &\leq x_2, \\
(11/13) &\leq x_2, \\
1 &\geq x_2, \\
x_2 &\leq (1/3), \\
x_2 &\geq 0.
\end{aligned}$$

Fourier-Motzkin Elimination (Example)

$$
\begin{aligned}
0 &\leq -2 + 3x_2, \\
(1/5) + (2/5)x_2 &\leq -2 + 3x_2, \\
0 &\leq (1/2) - (1/2)x_2, \\
(1/5) + (2/5)x_2 &\leq (1/2) - (1/2)x_2, \\
x_2 &\geq 0.
\end{aligned}
$$

Simplifying, we get: **a conflict**

$$
\begin{aligned}
(2/3) &\leq x_2, \\
(11/13) &\leq x_2, \\
1 &\geq x_2, \\
x_2 &\leq (1/3), \\
x_2 &\geq 0.
\end{aligned}
$$

Fourier-Motzkin Elimination (Example)

From this conflict, we can conclude that our system of linear inequalities is **infeasible** and therefore does not admit any solutions.

$$
\begin{aligned}
(2/3) &\leq x_2, \\
(11/13) &\leq x_2, \\
1 &\geq x_2, \\
x_2 &\leq (1/3), \\
x_2 &\geq 0.
\end{aligned}
$$

N.B. If there hadn't been any conflict, we could have found a value for x_2 within the constraints, and used it to find a value for x_1 by substitution, obtaining a feasible point.

Complexity of Fourier-Motzkin Elimination

While a very easy method, Fourier-Motzkin Elimination suffers from terrible worst case time complexity when viewed as an algorithm.

For a linear system with n variables, reducing it down to system with only one variable would take (in the worst case) $2^{2^{n-1}+2}$ steps.

The time complexity of Fourier-Motzkin Elimination is **doubly-exponential** in the number of variables.

FME is thus not a practical method for checking feasibility of linear inequality constraints, but can be applied successfully on small problems.

Linear Programs

A general **Linear Program** (i.e. a *linear optimisation problem*) has three key features:

1. A linear *objective function* to be maximised/minimised, e.g.

$$c_1x_1 + c_2x_2 + \cdots + c_nx_n\,.$$

   We often use vector notation and dot product to write $\vec{c} \cdot \vec{x}$.

2. A system of linear constraints, e.g. constraints may given by a system of linear inequalities, which may concisely written using matrix notation as $A\vec{x} \leq \vec{b}$ (the constraints may also be of the form $A\vec{x} \geq \vec{b}$, or $A\vec{x} = \vec{b}$, or a combination thereof).

3. The decision variables are non-negative, i.e. $\vec{x} \geq \vec{0}$.

Linear Programs

In solving a Linear Program (LP), we are concerned with solving a linear optimisation problem:

$$\begin{aligned} \textbf{minimise/maximise:} \quad & \vec{c} \cdot \vec{x}, \\ \textbf{subject to:} \quad & A\vec{x} \leq \vec{b}, \\ & \vec{x} \geq \vec{0}. \end{aligned}$$

N.B. We can always reformulate a maximisation problem as a minimisation problem, and vice versa. This is because

$$-\max_{\vec{x}} \; -(\vec{c} \cdot \vec{x}) \; = \; \min_{\vec{x}} \; \vec{c} \cdot \vec{x}\,.$$

Linear Programs

In solving a Linear Program (LP), we are concerned with solving a linear optimisation problem:

$$\begin{aligned} \textbf{minimise/maximise:} \quad & \vec{c} \cdot \vec{x}, \\ \textbf{subject to:} \quad & A\vec{x} \leq \vec{b}, \\ & \vec{x} \geq \vec{0}. \end{aligned}$$

N.B. We can always reformulate a maximisation problem as a minimisation problem, and vice versa. This is because

$$-\max_{\vec{x}} \; -(\vec{c} \cdot \vec{x}) \;=\; \min_{\vec{x}} \; \vec{c} \cdot \vec{x} \,.$$

A **huge number** of problems can be mapped to Linear Programming problems.

Modelling Problems as a Linear Programs

In Linear Programming being able to correctly **model** *problems is as important as being able to solve Linear Programs.*

Modelling Problems as a Linear Programs

In Linear Programming being able to correctly **model** *problems is as important as being able to solve Linear Programs.*

Suppose we wish to maintain a fruit diet on a very tight budget.

Modelling Problems as a Linear Programs

In Linear Programming being able to correctly **model** *problems is as important as being able to solve Linear Programs.*

Suppose we wish to maintain a fruit diet on a very tight budget.

As with any diet, we will have certain *requirements* in terms of nutrients. We would like to meet these at minimum cost.

Modelling Problems as a Linear Programs

In Linear Programming being able to correctly **model** *problems is as important as being able to solve Linear Programs.*

Suppose we wish to maintain a fruit diet on a very tight budget.

As with any diet, we will have certain *requirements* in terms of nutrients. We would like to meet these at minimum cost.

Assume (for simplicity) that only four kinds of fruit are available in our local greengrocer: **apples**, **pears**, **oranges** and **tomatoes**.

Modelling Problems as a Linear Programs

In Linear Programming being able to correctly **model** *problems is as important as being able to solve Linear Programs.*

Suppose we wish to maintain a fruit diet on a very tight budget.

As with any diet, we will have certain *requirements* in terms of nutrients. We would like to meet these at minimum cost.

Assume (for simplicity) that only four kinds of fruit are available in our local greengrocer: **apples**, **pears**, **oranges** and **tomatoes**.

Suppose also that our daily dietary requirements are given to us in terms of: **vitamin C**, **calcium**, **iron** and **energy** (i.e. calories):

- We need to consume no more than 20mg of iron, over 60mg of vitamin C, and between 700mg and 1,500mg of calcium.
- We need to consume no more than 2,200 calories.

A Fruit Diet

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple					
Pear					
Orange					
Tomato					

A Fruit Diet

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6				
Pear	4.3				
Orange	53				
Tomato	14				

A Fruit Diet

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6			
Pear	4.3	9			
Orange	53	40			
Tomato	14	9			

A Fruit Diet

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12		
Pear	4.3	9	0.12		
Orange	53	40	0.1		
Tomato	14	9	0.1		

A Fruit Diet

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	
Pear	4.3	9	0.12	57	
Orange	53	40	0.1	47	
Tomato	14	9	0.1	18	

A Fruit Diet

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

A Fruit Diet

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

What about our requirements (constraints)?

A Fruit Diet

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

What about our requirements (constraints)?

Energy ≤ 2200, iron ≤ 20, vitamin C ≥ 60, $700 \leq$ calcium ≤ 1500.

A Fruit Diet

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

What about our requirements (constraints)?

Energy ≤ 2200, iron ≤ 20, vitamin C ≥ 60, $700 \leq$ calcium ≤ 1500.

The amount of fruit also needs to be non-negative (obviously):
Apple ≥ 0, Pear ≥ 0, Orange ≥ 0, Tomato ≥ 0.

A Fruit Diet

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

What about our requirements (constraints)?

Energy ≤ 2200, iron ≤ 20, vitamin C ≥ 60, $700 \leq$ calcium ≤ 1500.

The amount of fruit also needs to be non-negative (obviously): Apple ≥ 0, Pear ≥ 0, Orange ≥ 0, Tomato ≥ 0.

We would like to satisfy these constraints and minimise the cost of our diet. How can we model our problem as a **Linear Program**?

LP Problem Modelling. Step 1: Identify Objective Function.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

Before we worry about constraints, we must ask ourselves: what are we optimising? More formally: *what is the objective function*?

LP Problem Modelling. Step 1: Identify Objective Function.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

Before we worry about constraints, we must ask ourselves: what are we optimising? More formally: *what is the objective function*?

In this case, we're interested in minimising price, but how?

LP Problem Modelling. Step 1: Identify Objective Function.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

Before we worry about constraints, we must ask ourselves: what are we optimising? More formally: *what is the objective function*?

In this case, we're interested in minimising price, but how?

We wish to know how many *units* of each fruit to buy, and each type of fruit is associated with a price.

LP Problem Modelling. Step 1: Identify Objective Function.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

Before we worry about constraints, we must ask ourselves: what are we optimising? More formally: *what is the objective function*?

In this case, we're interested in minimising price, but how?

We wish to know how many *units* of each fruit to buy, and each type of fruit is associated with a price. Our objective function is :

$$57\,\text{Apple} + 62.5\,\text{Pear} + 72.5\,\text{Orange} + 16\,\text{Tomato}\,.$$

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

We now have an objective function. The next step is to write down the constraints on our decision variables.

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

We now have an objective function. The next step is to write down the constraints on our decision variables.

We follow a similar process: the total amount of each nutrient in our tentative selection of fruit can be written as a linear function. E.g., the total amount of vitamin C is given by:

$$4.6\,\mathsf{Apple} + 4.3\,\mathsf{Pear} + 53\,\mathsf{Orange} + 14\,\mathsf{Tomato}\,.$$

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
Apple	4.6	6	0.12	52	57
Pear	4.3	9	0.12	57	62.5
Orange	53	40	0.1	47	72.5
Tomato	14	9	0.1	18	16

Things are now getting bulky. Let's agree that:

$$\text{Apple} = x_1,$$
$$\text{Pear} = x_2,$$
$$\text{Orange} = x_3,$$
$$\text{Tomato} = x_4.$$

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
x_1	4.6	6	0.12	52	57
x_2	4.3	9	0.12	57	62.5
x_3	53	40	0.1	47	72.5
x_4	14	9	0.1	18	16

Things are now getting bulky. Let's agree that:

$$\text{Apple} = x_1,$$
$$\text{Pear} = x_2,$$
$$\text{Orange} = x_3,$$
$$\text{Tomato} = x_4.$$

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
x_1	4.6	6	0.12	52	57
x_2	4.3	9	0.12	57	62.5
x_3	53	40	0.1	47	72.5
x_4	14	9	0.1	18	16

The total amount of vitamin C in our fruit basket is now:

$$4.6x_1 + 4.3x_2 + 53x_3 + 14x_4\,.$$

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
x_1	4.6	6	0.12	52	57
x_2	4.3	9	0.12	57	62.5
x_3	53	40	0.1	47	72.5
x_4	14	9	0.1	18	16

The total amount of vitamin C in our fruit basket is now:

$$4.6x_1 + 4.3x_2 + 53x_3 + 14x_4\,.$$

Remember that we need more than 60mg of vitamin C in our diet, so we obtain the constraint:

$$4.6x_1 + 4.3x_2 + 53x_3 + 14x_4 \geq 60\,.$$

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
x_1	4.6	6	0.12	52	57
x_2	4.3	9	0.12	57	62.5
x_3	53	40	0.1	47	72.5
x_4	14	9	0.1	18	16

The total amount of calcium in our fruit basket is given by:

$$6x_1 + 9x_2 + 40x_3 + 9x_4 \,.$$

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
x_1	4.6	6	0.12	52	57
x_2	4.3	9	0.12	57	62.5
x_3	53	40	0.1	47	72.5
x_4	14	9	0.1	18	16

The total amount of calcium in our fruit basket is given by:

$$6x_1 + 9x_2 + 40x_3 + 9x_4 \,.$$

We require between 700 and 1,500mg of calcium in our diet, so we obtain the following two constraints:

$$6x_1 + 9x_2 + 40x_3 + 9x_4 \geq 700 \,,$$

$$6x_1 + 9x_2 + 40x_3 + 9x_4 \leq 1500 \,.$$

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
x_1	4.6	6	0.12	52	57
x_2	4.3	9	0.12	57	62.5
x_3	53	40	0.1	47	72.5
x_4	14	9	0.1	18	16

The total amount of iron is:

$$0.12x_1 + 0.12x_2 + 0.1x_3 + 0.1x_4\,.$$

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
x_1	4.6	6	0.12	52	57
x_2	4.3	9	0.12	57	62.5
x_3	53	40	0.1	47	72.5
x_4	14	9	0.1	18	16

The total amount of iron is:

$$0.12x_1 + 0.12x_2 + 0.1x_3 + 0.1x_4\,.$$

We cannot consume more than 20mg of iron daily, so we get:

$$0.12x_1 + 0.12x_2 + 0.1x_3 + 0.1x_4 \leq 20\,.$$

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
x_1	4.6	6	0.12	52	57
x_2	4.3	9	0.12	57	62.5
x_3	53	40	0.1	47	72.5
x_4	14	9	0.1	18	16

The total amount energy in our diet is:

$$52x_1 + 57x_2 + 47x_3 + 18x_4\,.$$

LP Problem Modelling. Step 2: Impose Constraints.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
x_1	4.6	6	0.12	52	57
x_2	4.3	9	0.12	57	62.5
x_3	53	40	0.1	47	72.5
x_4	14	9	0.1	18	16

The total amount energy in our diet is:

$$52x_1 + 57x_2 + 47x_3 + 18x_4\,.$$

To stay healthy, we need to consume fewer than 2,200 calories; thus our constraint is:

$$52x_1 + 57x_2 + 47x_3 + 18x_4 \leq 2200\,.$$

LP Problem Modelling. Step 3: Write Down the Problem.

	Nutrients (mg/unit)				
Fruit	Vitamin C	Calcium	Iron	Energy (kcal)	Price (p)
x_1	4.6	6	0.12	52	57
x_2	4.3	9	0.12	57	62.5
x_3	53	40	0.1	47	72.5
x_4	14	9	0.1	18	16

minimise: $57x_1 + 62.5x_2 + 72.5x_3 + 16x_4\,,$

subject to:

$$4.6x_1 + 4.3x_2 + 53x_3 + 14x_4 \geq 60$$

$$6x_1 + 9x_2 + 40x_3 + 9x_4 \geq 700$$

$$6x_1 + 9x_2 + 40x_3 + 9x_4 \leq 1500$$

$$0.12x_1 + 0.12x_2 + 0.1x_3 + 0.1x_4 \leq 20$$

$$52x_1 + 57x_2 + 47x_3 + 18x_4 \leq 2200$$

$$x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0.$$

LP Problem Modelling. Step 4: Solve.

What happens when we solve this optimisation problem (using an LP sover)?

$$
\begin{aligned}
\textbf{minimise}: \quad & 57x_1 + 62.5x_2 + 72.5x_3 + 16x_4\,, \\
\textbf{subject to}: \quad & 4.6x_1 + 4.3x_2 + 53x_3 + 14x_4 \geq 60 \\
& 6x_1 + 9x_2 + 40x_3 + 9x_4 \geq 700 \\
& 6x_1 + 9x_2 + 40x_3 + 9x_4 \leq 1500 \\
& 0.12x_1 + 0.12x_2 + 0.1x_3 + 0.1x_4 \leq 20 \\
& 52x_1 + 57x_2 + 47x_3 + 18x_4 \leq 2200 \\
& x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0.
\end{aligned}
$$

LP Problem Modelling. Step 4: Solve.

What happens when we solve this optimisation problem (using an LP sover)?

$$
\begin{aligned}
\textbf{minimise:}\quad & 57x_1 + 62.5x_2 + 72.5x_3 + 16x_4\,, \\
\textbf{subject to:}\quad & 4.6x_1 + 4.3x_2 + 53x_3 + 14x_4 \geq 60 \\
& 6x_1 + 9x_2 + 40x_3 + 9x_4 \geq 700 \\
& 6x_1 + 9x_2 + 40x_3 + 9x_4 \leq 1500 \\
& 0.12x_1 + 0.12x_2 + 0.1x_3 + 0.1x_4 \leq 20 \\
& 52x_1 + 57x_2 + 47x_3 + 18x_4 \leq 2200 \\
& x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0.
\end{aligned}
$$

Solution: $x_1 = 0,\ x_2 = 0,\ x_3 = 0,\ x_4 = 77.7778$, which would cost 1244.44p. (Only eat tomatoes.)

LP Problem Modelling. Step 4: Solve.

What if we're not *that* keen of tomatoes?

LP Problem Modelling. Step 4: Solve.

What if we're not *that* keen of tomatoes?

$$
\begin{aligned}
\textbf{minimise}: \quad & 57x_1 + 62.5x_2 + 72.5x_3 + 16x_4\,, \\
\textbf{subject to}: \quad & 4.6x_1 + 4.3x_2 + 53x_3 + 14x_4 \geq 60 \\
& 6x_1 + 9x_2 + 40x_3 + 9x_4 \geq 700 \\
& 6x_1 + 9x_2 + 40x_3 + 9x_4 \leq 1500 \\
& 0.12x_1 + 0.12x_2 + 0.1x_3 + 0.1x_4 \leq 20 \\
& 52x_1 + 57x_2 + 47x_3 + 18x_4 \leq 2200 \\
& x_4 \leq 10 \\
& x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0.
\end{aligned}
$$

LP Problem Modelling. Step 4: Solve.

What if we're not *that* keen of tomatoes?

$$
\begin{aligned}
\textbf{minimise}\text{:} \quad & 57x_1 + 62.5x_2 + 72.5x_3 + 16x_4\,, \\
\textbf{subject to}\text{:} \quad & 4.6x_1 + 4.3x_2 + 53x_3 + 14x_4 \geq 60 \\
& 6x_1 + 9x_2 + 40x_3 + 9x_4 \geq 700 \\
& 6x_1 + 9x_2 + 40x_3 + 9x_4 \leq 1500 \\
& 0.12x_1 + 0.12x_2 + 0.1x_3 + 0.1x_4 \leq 20 \\
& 52x_1 + 57x_2 + 47x_3 + 18x_4 \leq 2200 \\
& x_4 \leq 10 \\
& x_1 \geq 0,\ x_2 \geq 0,\ x_3 \geq 0,\ x_4 \geq 0.
\end{aligned}
$$

Solution: $x_1 = 0,\ x_2 = 0,\ x_3 = 15.25,\ x_4 = 10$, at the cost of 1265.63p. (More expensive, but we can have oranges for variety.)

LP Problem Modelling. (Another Example)

A huge number of practical problems can be modelled using LP.

To give another example: suppose we have a problem where we're interested in shipping commodities (from some finite set C) produced by a number of different factories F.

- The amount of commodity $c \in C$ produced by factory $f \in F$ is denoted by x_{cf}.
- The shipping cost of commodity c from factory f to its retailer is denoted by p_{cf}.
- We want to choose x_{cf} in such a way as to minimise the overall shipping costs, i.e.

$$\sum_{c \in C,\ f \in F} p_{cf} x_{cf}$$

- However, we have other constraints.

LP Problem Modelling. (Another Example)

- Each factory can only produce a certain amount of commodities:

$$\forall\ f \in F. \qquad \sum_{c \in C} x_{cf} \le b_f\,.$$

- There is a finite demand d_c for each commodity c:

$$\forall\ c \in C. \qquad \sum_{f \in F} x_{cf} = d_c\,.$$

- Obviously we can only produce positive amounts of commodities, so $x_{cf} \ge 0$.

LP Problem Modelling. (Another Example)

We arrive at the following LP formulation of the problem:

minimise: $$\sum_{c \in C,\ f \in F} p_{cf} x_{cf}\,,$$

subject to:

$$\forall\ f \in F. \quad \sum_{c \in C} x_{cf} \leq b_f$$

$$\forall\ c \in C. \quad \sum_{f \in F} x_{cf} = d_c\,,$$

$$\forall\ c \in C.\ \forall\ f \in F. \quad x_{cf} \geq 0.$$

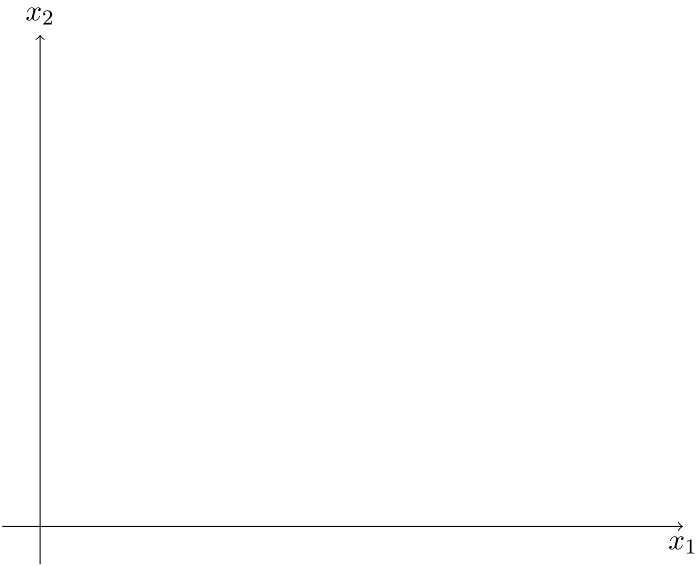
LP Solvers

- Realistic problems have many more constraints and a large number variables.
- **Tremendous progress** has been made in improving the efficiency of Linear Programming solvers (see paper by Bixby on the last slide).
- State-of-the-art solvers can deal with problem instances with *hundreds of thousands*, or even *millions* of variables.
- You have access to efficient LP solvers through tools such as MATLAB (`linprog()`) and Mathematica (`LinearProgramming[]`).
- Other noteworthy packages offering LP solver functionality: GLPK (GNU Linear Programming Kit).

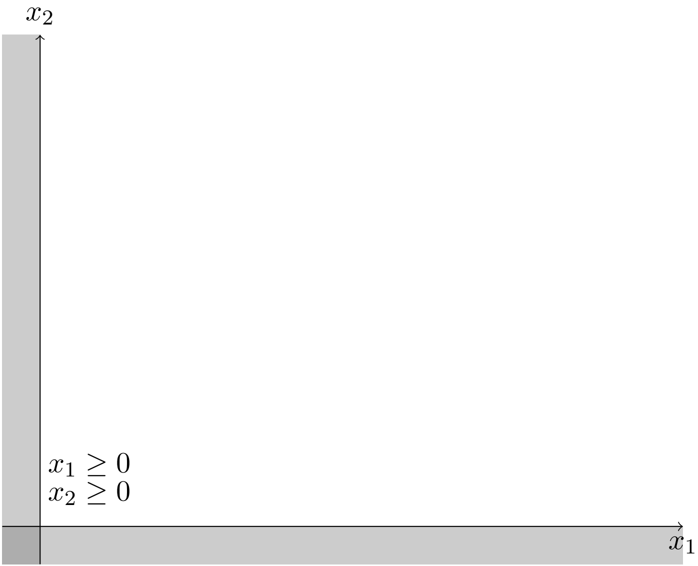
Structure of Linear Programs

- The constraints in a Linear Program describe a **convex polytope** in n-dimensional space.
- The objective function will attain its minimum/maximum at a **vertex** of the polytope (i.e. the optimum is never in the interior).
- The set of constraints may be **infeasible**, in which case a linear program has *no solutions*.
- This can be rather disappointing, but should not happen if we have formulated a sensible problem.

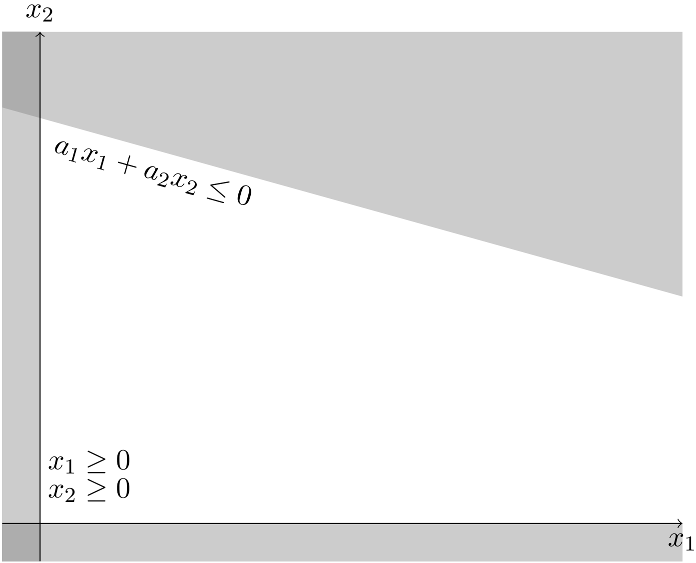
Linear Programming (Unique Solutions)



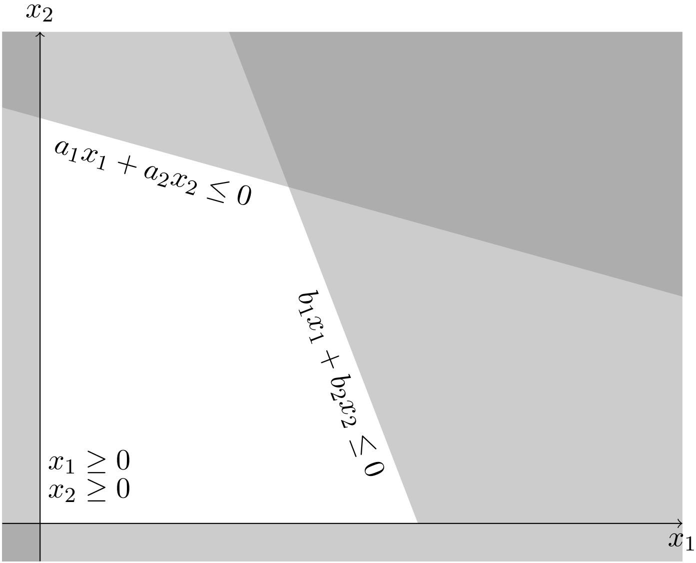
Linear Programming (Unique Solutions)



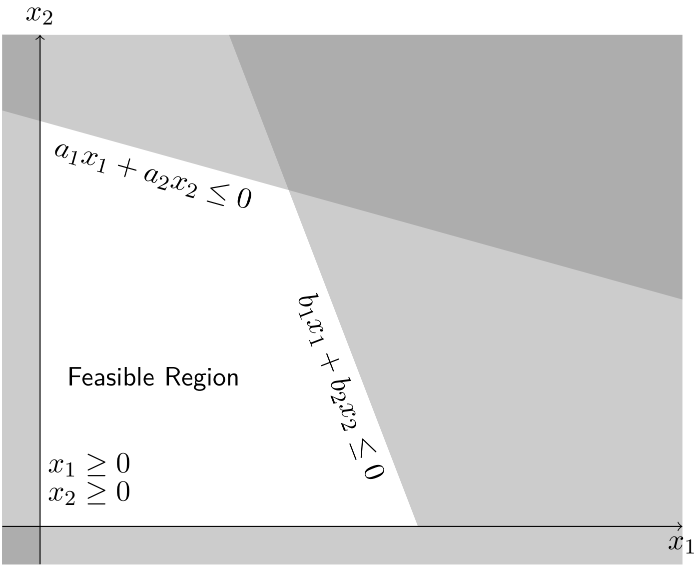
Linear Programming (Unique Solutions)



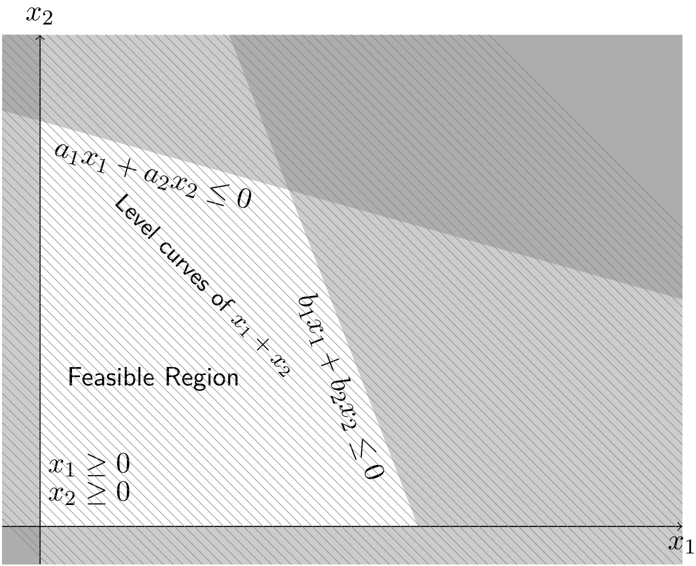
Linear Programming (Unique Solutions)



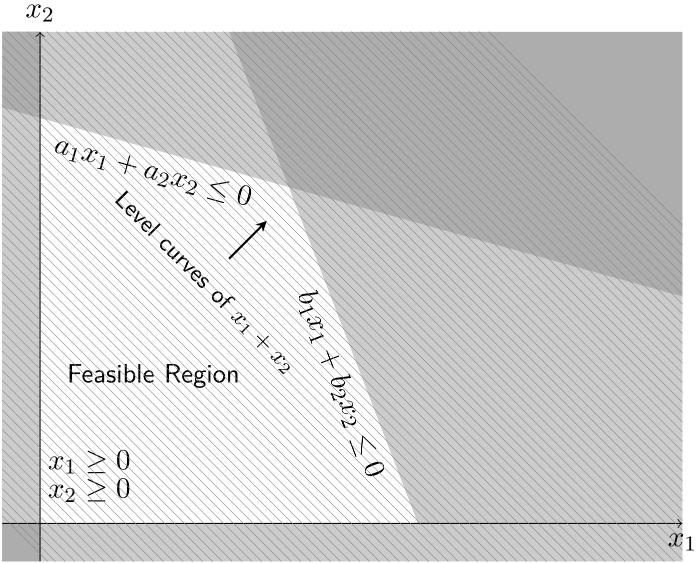
Linear Programming (Unique Solutions)



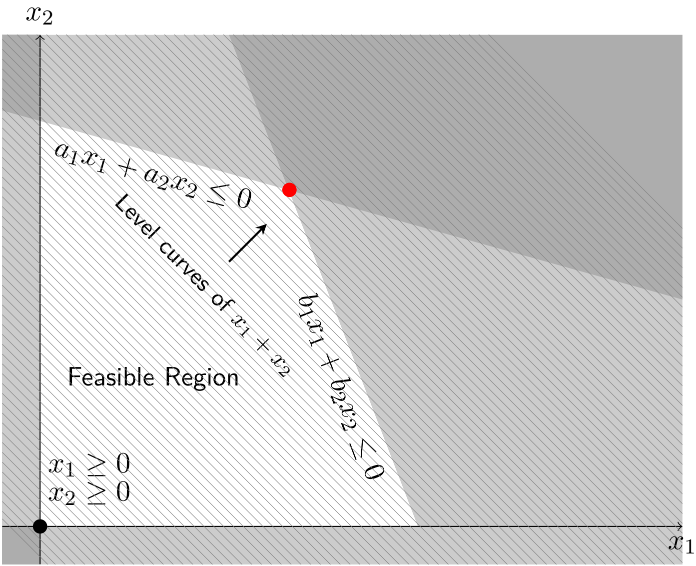
Linear Programming (Unique Solutions)



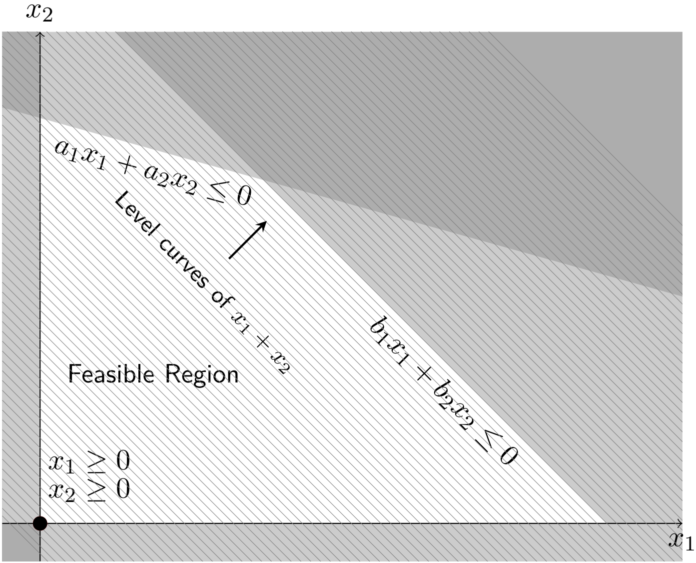
Linear Programming (Unique Solutions)



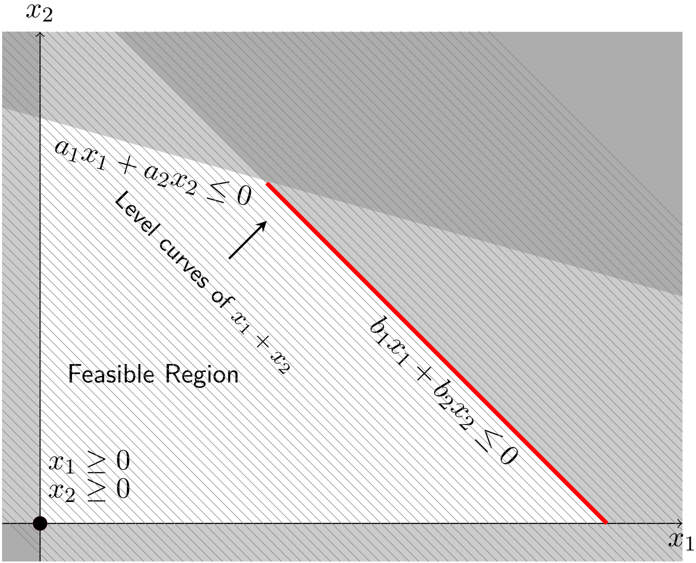
Linear Programming (Unique Solutions)



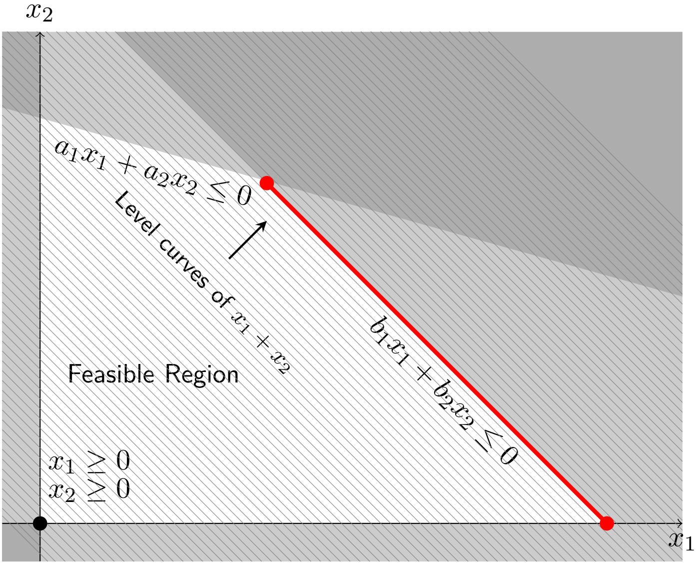
Linear Programming (Multiple Solutions)



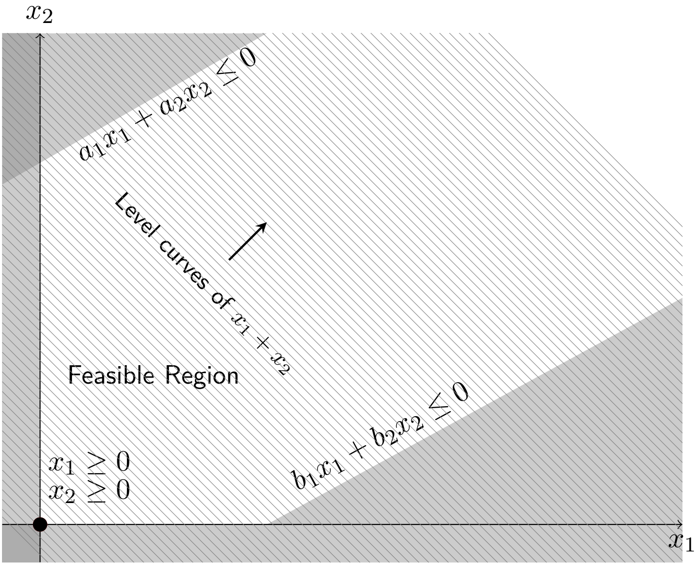
Linear Programming (Multiple Solutions)



Linear Programming (Multiple Solutions)



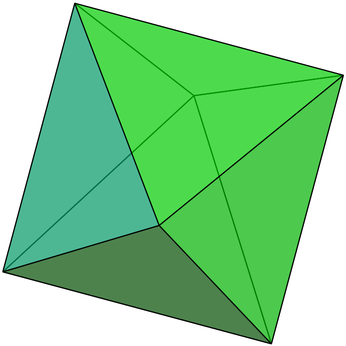
Linear Programming (Unbounded Solutions)



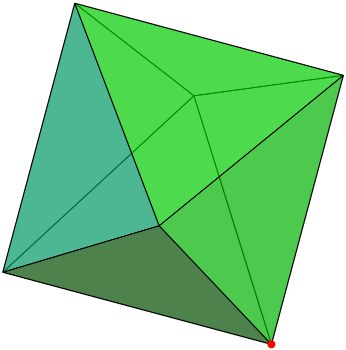
Linear Programming (Solution Strategy)

- The space of *feasible solutions* is a polytope.
- The maximum/minimum of a linear objective function will always lie on a vertex of the polytope.
- Our solution policy will be to start at some vertex and move to a neighbouring vertex that gives the best improvement in cost.
- When no further moves are possible, we are done.
- However, there is still **a lot of work** to realise this solution strategy (Simplex Algorithm).

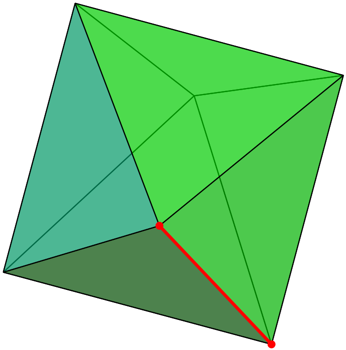
Linear Programming (Solution Strategy)



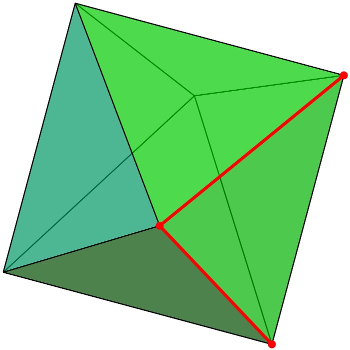
Linear Programming (Solution Strategy)



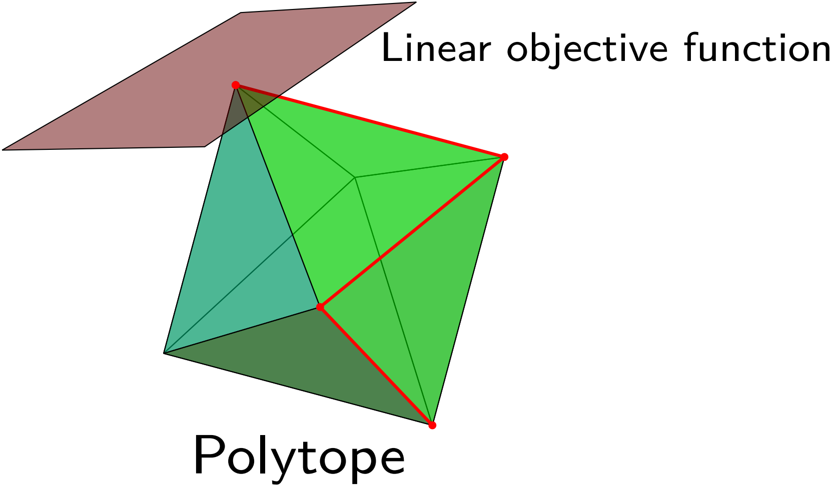
Linear Programming (Solution Strategy)



Linear Programming (Solution Strategy)



Linear Programming (Solution Strategy)



Further Topics in Linear Programming

- John von Neumann developed the idea of **duality** (turning a maximisation problem for a set of variables $\vec{x}$ into a minimisation problem for a *dual* set of variables $\vec{y}$ associated with each constraint).
- von Neumann used this idea as the basis for game theory (in particular for two-player zero-sum games).
- Unfortunately, we won't cover these exciting topics in this course. Next lecture will give a short overview of the ideas behind the Simplex Algorithm.

Further Reading:

1. **Jiří Matoušek, Bernd Gärtner** "Understanding and Using Linear Programming" https://link.springer.com/book/10.1007/978-3-540-30717-4
2. **Robert E. Bixby** "A Brief History of Linear and Mixed-Integer Programming Computation" https://www.math.uni-bielefeld.de/documenta/vol-ismp/25_bixby-robert.pdf

Optional (open problems):

- **Stephen Smale** "Mathematical Problems for the Next Century", *Problem 9: The Linear Programming Problem.* https://link.springer.com/content/pdf/10.1007/BF03025291.pdf (Problems 3, 5, 17 and 18 are also computer science problems; 17 has been solved, the rest are still open.)

Acknowledgements: Partly based on earlier COMP 1201 slides by Dr Adam Prügel-Bennett, University of Southampton.