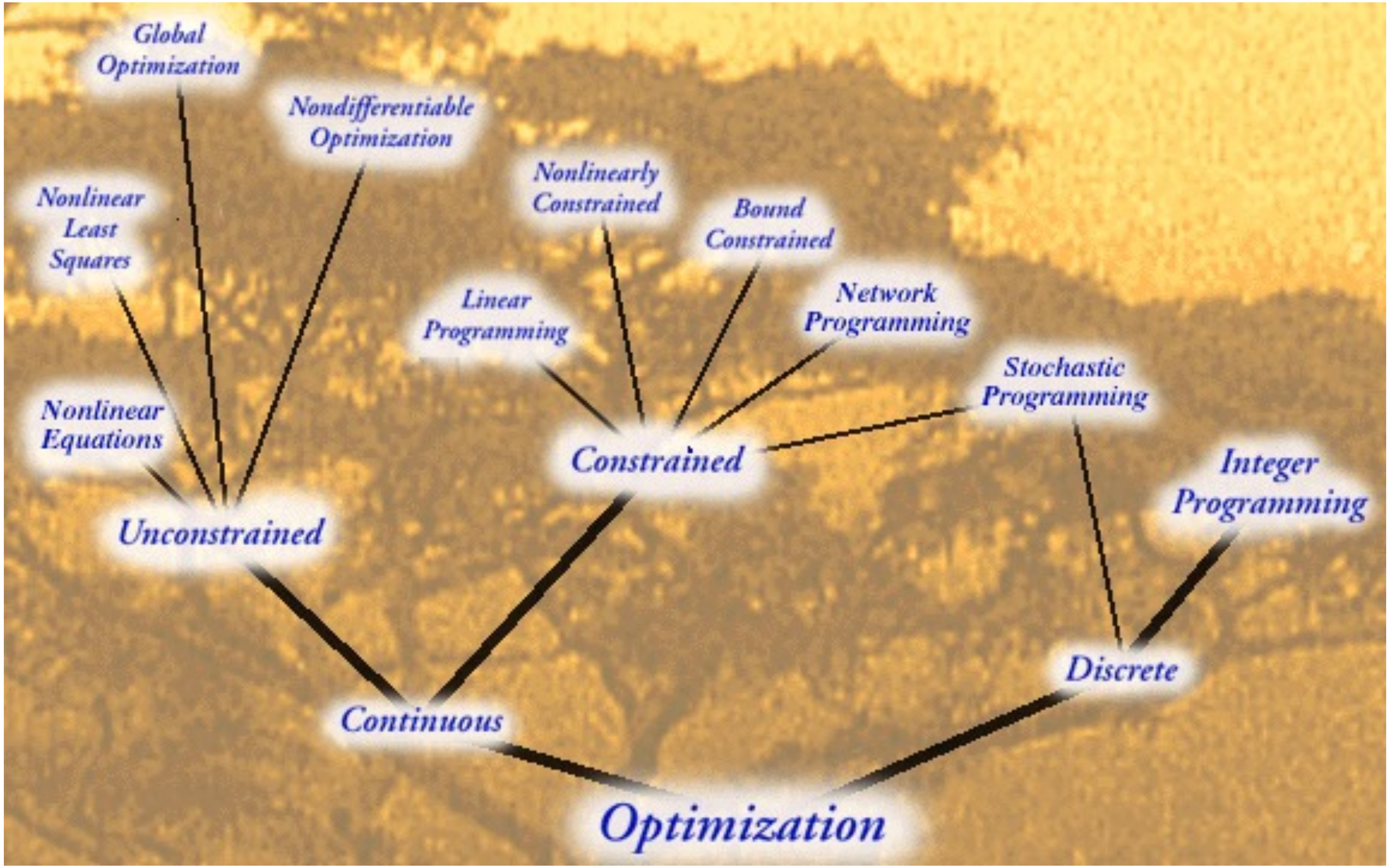
Lecture 3

B1 Optimization - Michaelmas 2023 - V. A. Prisacariu

Linear Programming

- Extreme solutions
- Simplex method
- Interior point method
- Integer programming and relaxation

The Optimization Tree



Linear Programming

The name is historical, it should really be called Linear Optimization.

The problem consists of three parts:

- A linear function to maximised:

$$\text{maximise } f(\mathbf{x}) = c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$$

- Problem constraints:

$$\text{subject to:}$$

$$\begin{gathered} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n \leq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n \leq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n \leq b_m \end{gathered}$$

- Non-negative variables:

$$\text{e.g. } x_1, x_2 \geq 0$$

Linear Programming

The problem is usually expressed in matrix form and then it becomes:

$$\begin{array}{ll} \text{Maximise} & \mathbf{c}^T\mathbf{x} \\ \text{subject to} & \mathrm{A}\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq 0 \end{array}$$

where A is an $m \times n$ matrix.

The objective function $\mathbf{c}^T\mathbf{x}$ and the constraints $\mathrm{A}\mathbf{x} \leq \mathbf{b}$ are all linear functions of $\mathbf{x}$.

Linear programming problems are convex and a local optimum is the global optimum.

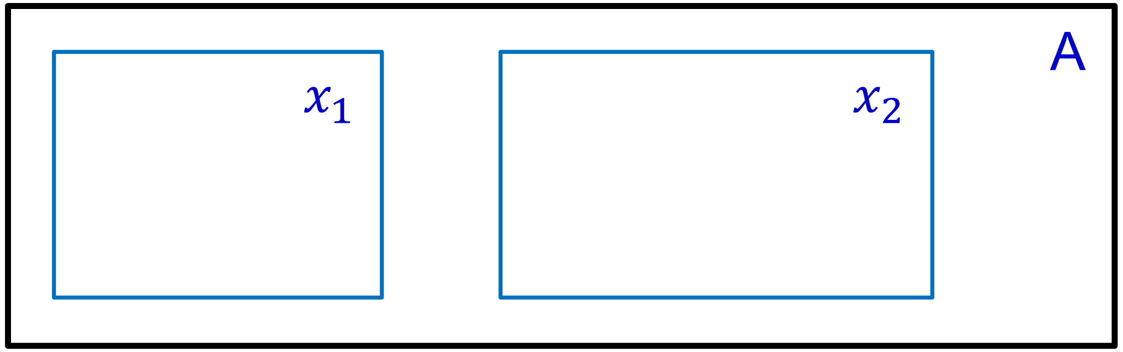
Example 1

A farmer has an area of A square kilometers to be planted with a combination of wheat and barley.

A limited amount F of fertilizer and P of insecticide can be used, each of which is required in different amounts per unit area for wheat (F1, P1) and barley (F2, P2).

Let S1 be the selling price of wheat, and S2 the price of barley, and denote the area planted with wheat and barley as x_1 and x_2 respectively.

The optimal number of square kilometers to plant with wheat vs. barley can be expressed as a linear programming problem.



Example 1

Maximise the revenue (this is the “objective” function):

$$S_1 x_1 + S_2 x_2$$

Subject to

$$x_1 + x_2 \leq \mathrm{A} \text{ (limit on the total area)}$$
$$F_1 x_1 + F_2 x_2 \leq F \text{ (limit on the fertilizer)}$$
$$P_1 x_1 + P_2 x_2 \leq P \text{ (limit on the insecticide)}$$
$$x_1 \geq 0, x_2 \geq 0 \text{ (cannot plant a negative area)}$$

In matrix form this becomes:

Maximize:

$$\begin{bmatrix} S_1 & S_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

Subject to:

$$\begin{bmatrix} 1 & 1 \\ F_1 & F_2 \\ P_1 & P_2 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \leq \begin{bmatrix} A \\ F \\ P \end{bmatrix}, \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \geq 0$$

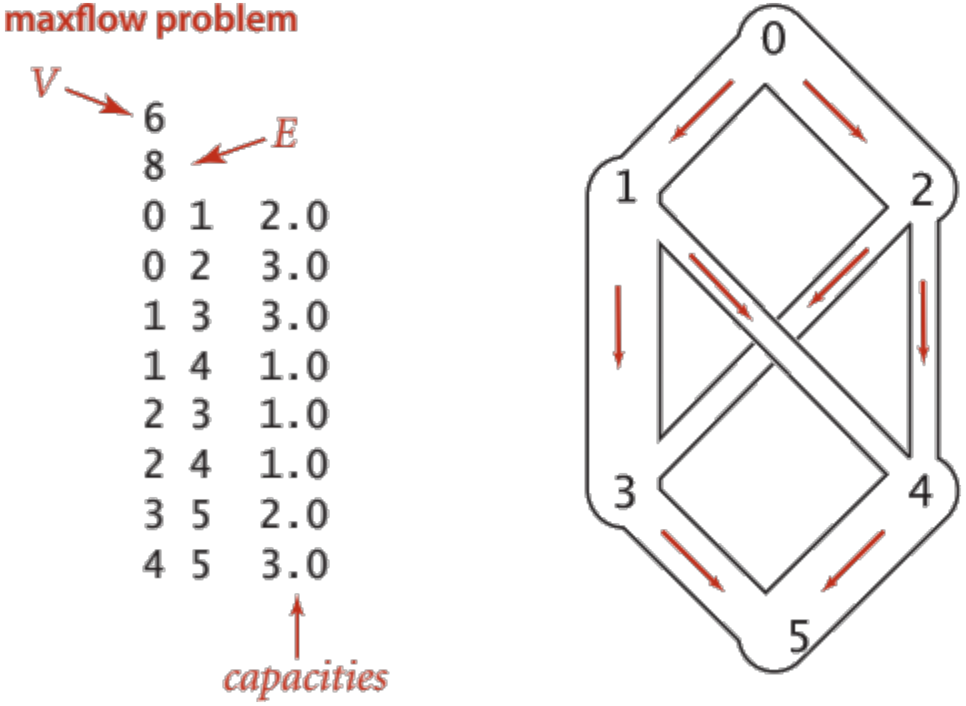
Example 2: Max Flow

Given: a weighted directed graph, source s, destination t.

Interpret edge weights as capacities.

Goal: Find maximum flow from s (node 0) to t (node 5)

- Flow does not exceed capacity in any edge;
- Flow at every vertex satisfies equilibrium [flow in equals flow out].



e.g. oil flowing through pipes, internet routing

Example 2: Max Flow

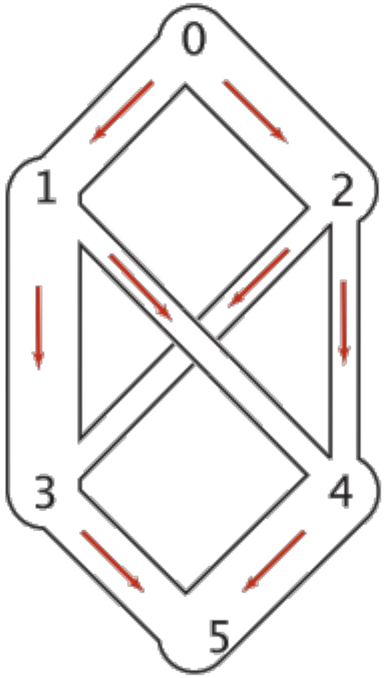
Variables: x_{vw} = flow on edge $v \to w$.
Constraints: Capacity and flow conservation.
Objective function: Net flow into t.

maxflow problem

V → 6
E → 8

```
0 1  2.0
0 2  3.0
1 3  3.0
1 4  1.0
2 3  1.0
2 4  1.0
3 5  2.0
4 5  3.0
```

↑ *capacities*



LP formulation

Maximize $x_{35} + x_{45}$
subject to the constraints

$$\left.\begin{aligned}
0 \le x_{01} \le 2 \\
0 \le x_{02} \le 3 \\
0 \le x_{13} \le 3 \\
0 \le x_{14} \le 1 \\
0 \le x_{23} \le 1 \\
0 \le x_{24} \le 1 \\
0 \le x_{35} \le 2 \\
0 \le x_{45} \le 3
\end{aligned}\right\} \text{capacity constraints}$$

$$\left.\begin{aligned}
x_{01} = x_{13} + x_{14} \\
x_{02} = x_{23} + x_{24} \\
x_{13} + x_{23} = x_{35} \\
x_{14} + x_{24} = x_{45}
\end{aligned}\right\} \text{flow conservation constraints}$$

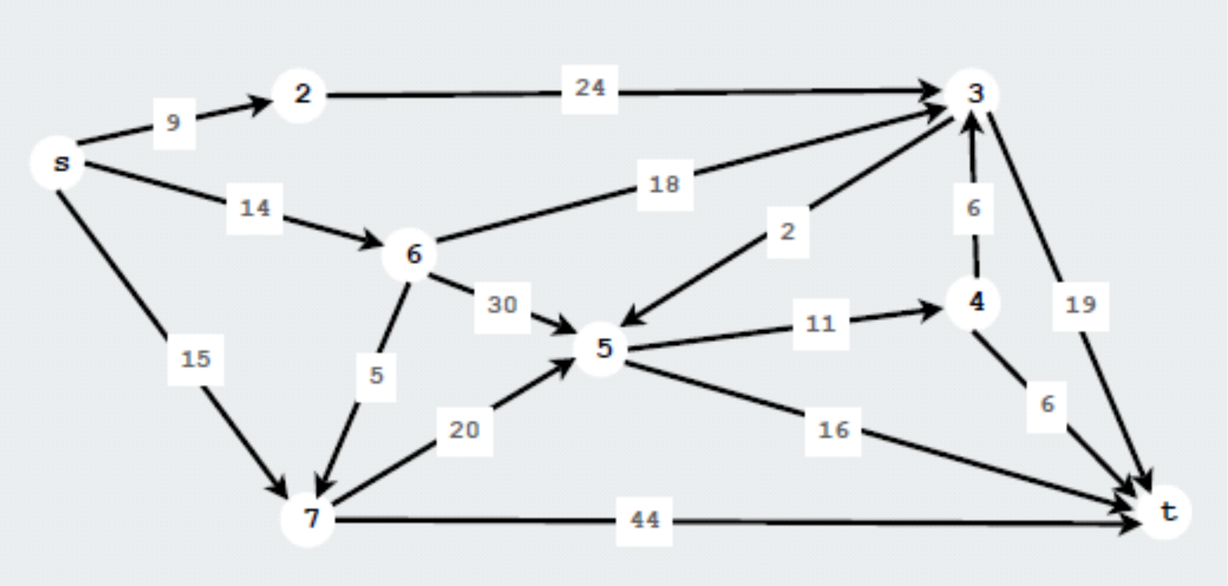
Slide: Robert Sedgewick and Kevin Wayne

Example 3: Shortest Path

Given: a weighted directed graph, with a single source s

Distance from s to t: length of the shortest part from s to t

Goal: Find distance (and shortest path) to every vertex



e.g. plotting routes on Google maps

LP – why is it important?

- We have seen examples of:
  - Allocating limited resources.
  - Network flow;
  - Shortest path;
- Others include:
  - Matching
  - Assignment …
- It is a widely applicable problem-solving model because:
  - Non-negativity is the usual constraint on any variable that represents an amount of something.
  - One is often interested in bounds imposed on limited resources.

Linear Programming 2D Example

$$\max_{x_1, x_2} f(x_1, x_2)$$

Cost function:

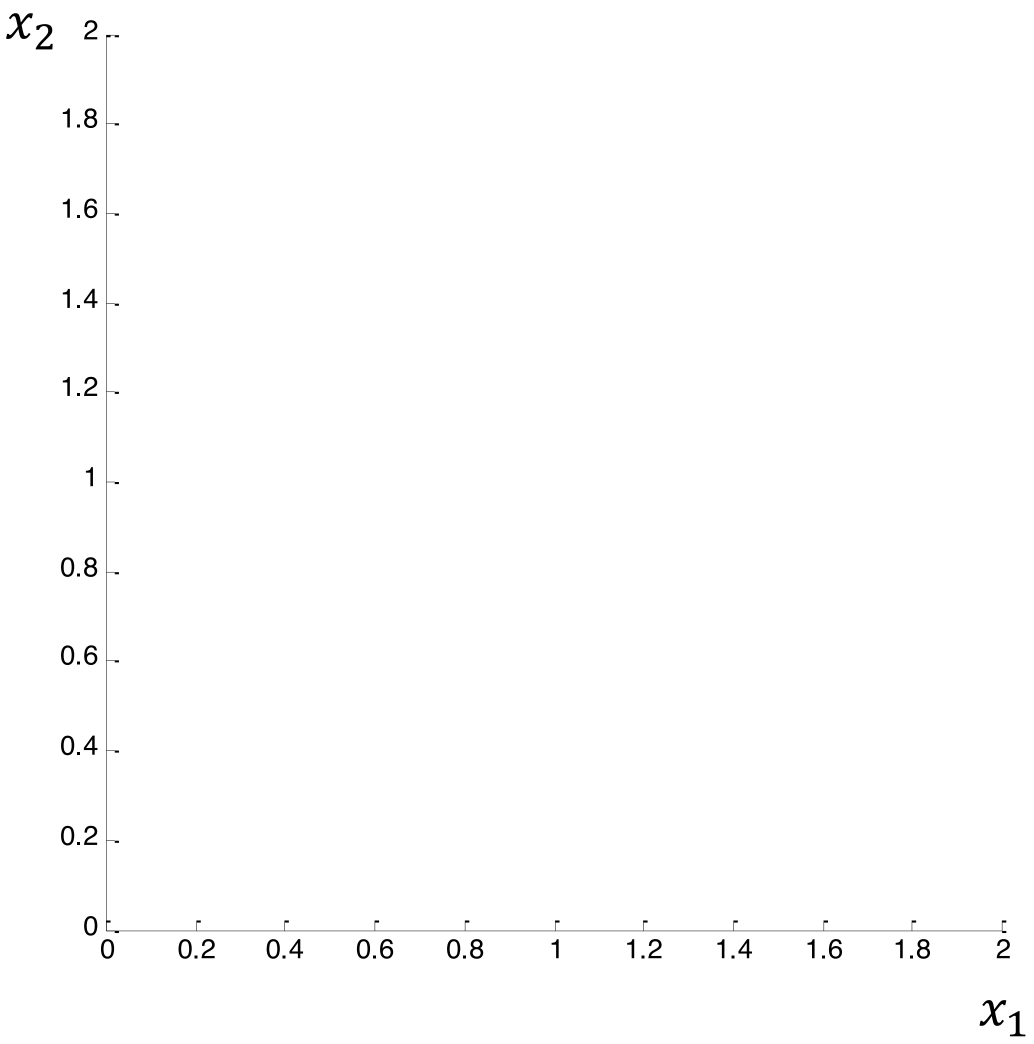
$$f(x_1, x_2) = 3x_1 + 4x_2$$

Inequality constraints:

$$\begin{aligned} x_1 + x_2 &\leq 2 \\ x_1 &\leq 1.5 \\ x_1 + \frac{8}{3}x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Feasible Region

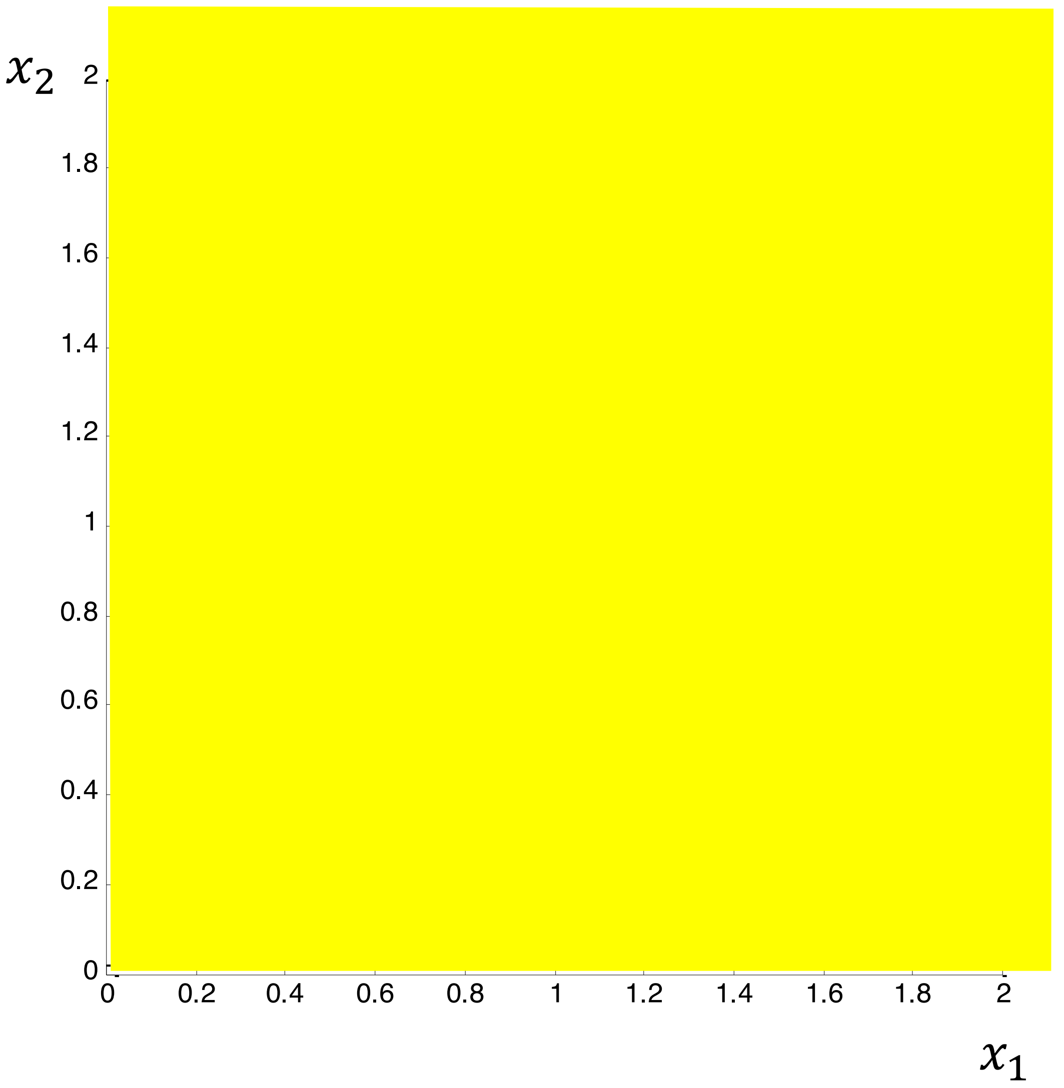
$$
\begin{aligned}
x_1 + x_2 &\leq 2 \\
x_1 &\leq 1.5 \\
x_1 + \frac{8}{3} x_2 &\leq 4 \\
x_1, x_2 &\geq 0
\end{aligned}
$$



Feasible Region

Inequality constraints

$$
\begin{aligned}
x_1 + x_2 &\leq 2 \\
x_1 &\leq 1.5 \\
x_1 + \frac{8}{3} x_2 &\leq 4 \\
x_1, x_2 &\geq 0
\end{aligned}
$$

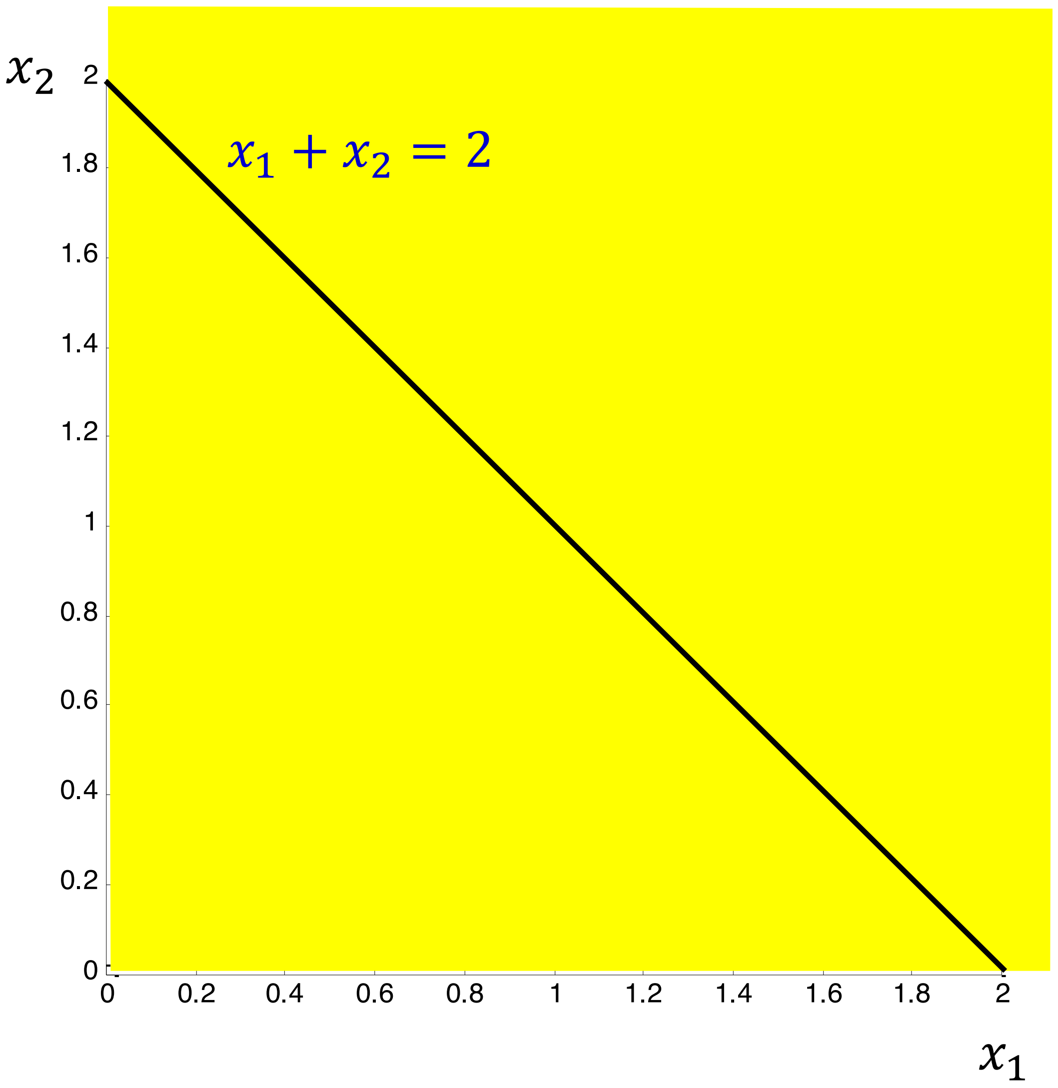


Feasible Region

Inequality constraints

$$\begin{aligned} x_1 + x_2 &\leq 2 \\ x_1 &\leq 1.5 \\ x_1 + \frac{8}{3}x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

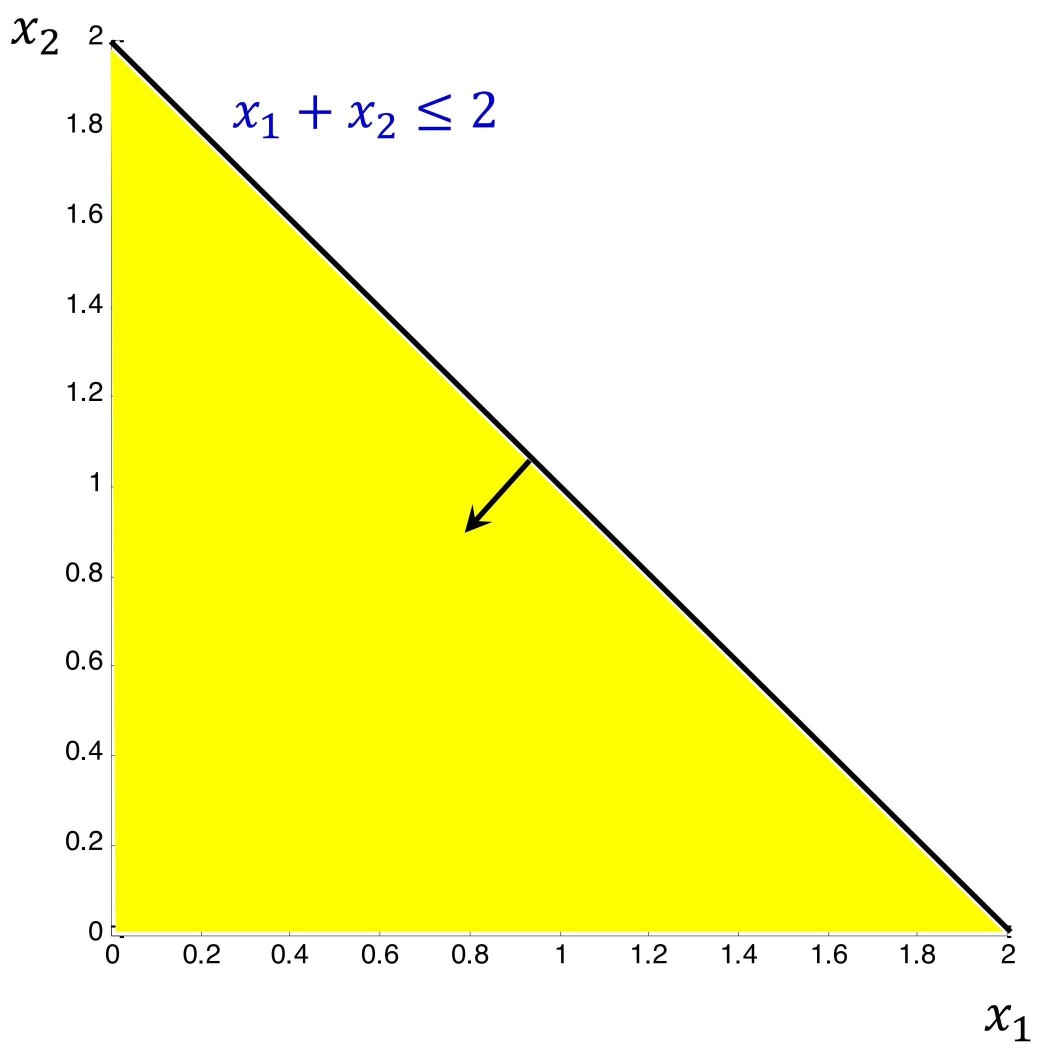
Each constraint gives a boundary.
Each boundary is a hyper-plane.



Feasible Region

Inequality constraints

$$\begin{aligned} x_1 + x_2 &\leq 2 \\ x_1 &\leq 1.5 \\ x_1 + \frac{8}{3}x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

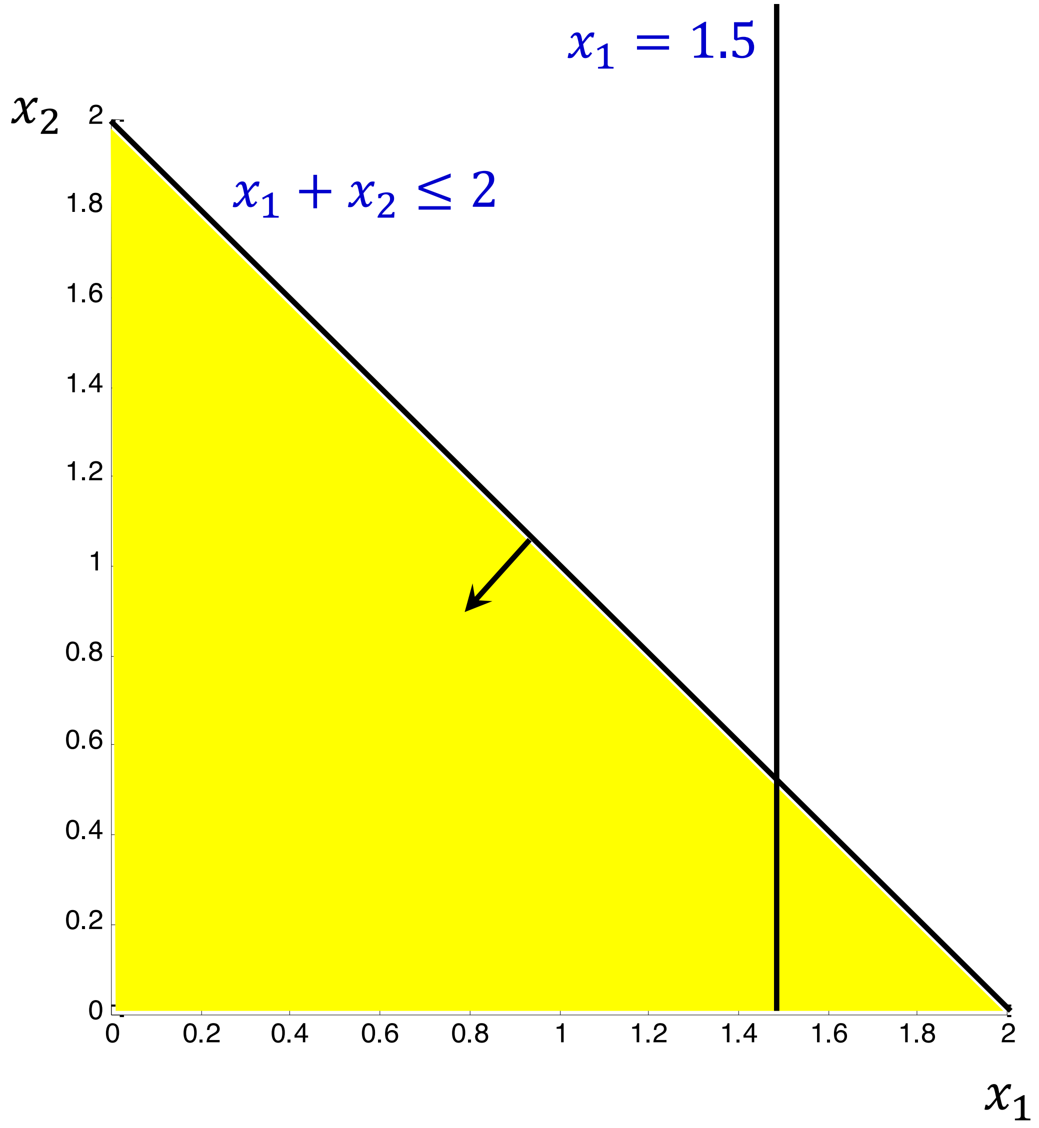


Each constraint gives a boundary.
Each boundary is a hyper-plane.

Feasible Region

Inequality constraints

$$\begin{aligned} x_1 + x_2 &\leq 2 \\ x_1 &\leq 1.5 \\ x_1 + \frac{8}{3}x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

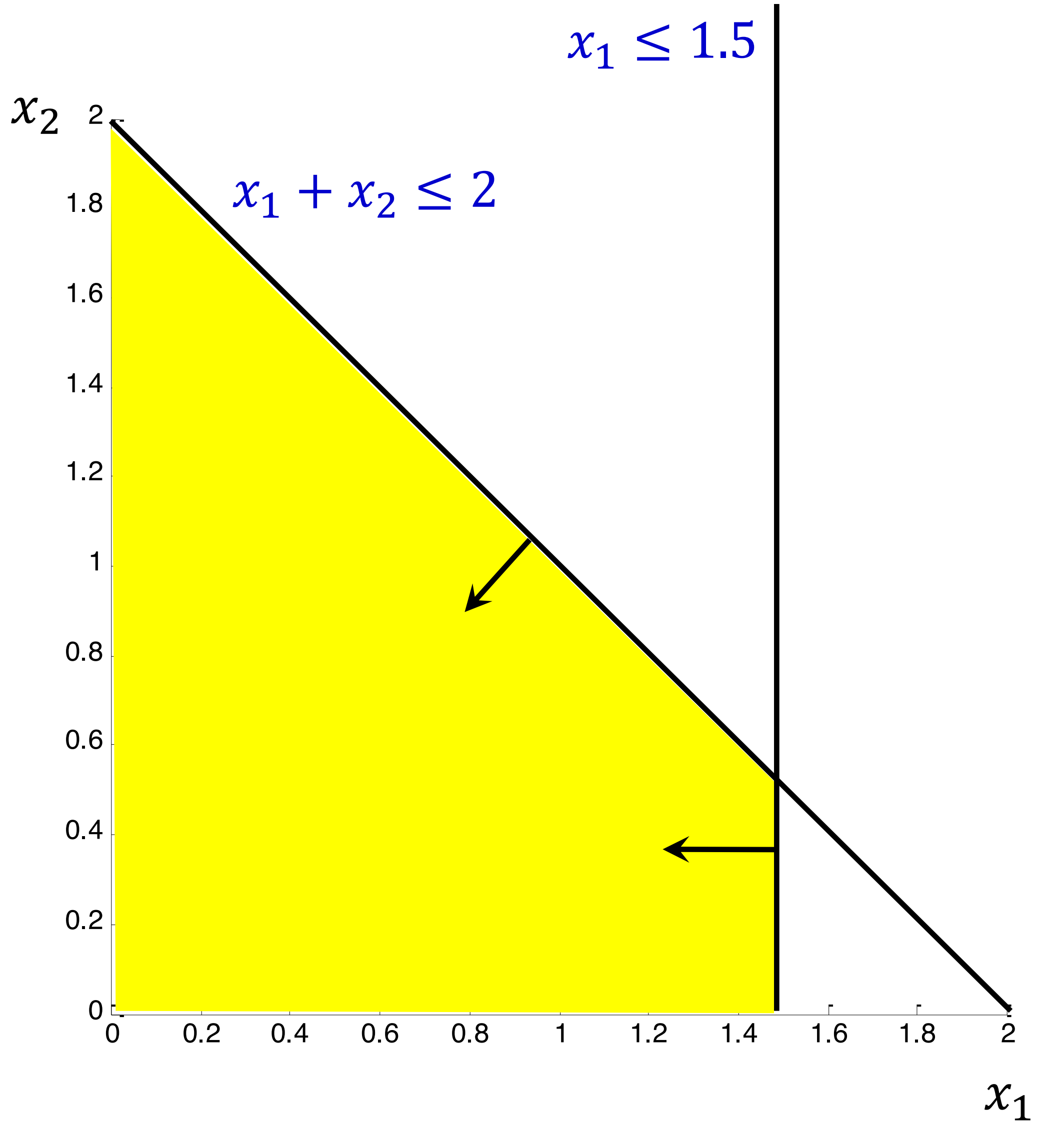


Each constraint gives a boundary.
Each boundary is a hyper-plane.

Feasible Region

Inequality constraints

$$\begin{aligned} x_1 + x_2 &\leq 2 \\ x_1 &\leq 1.5 \\ x_1 + \frac{8}{3}x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$



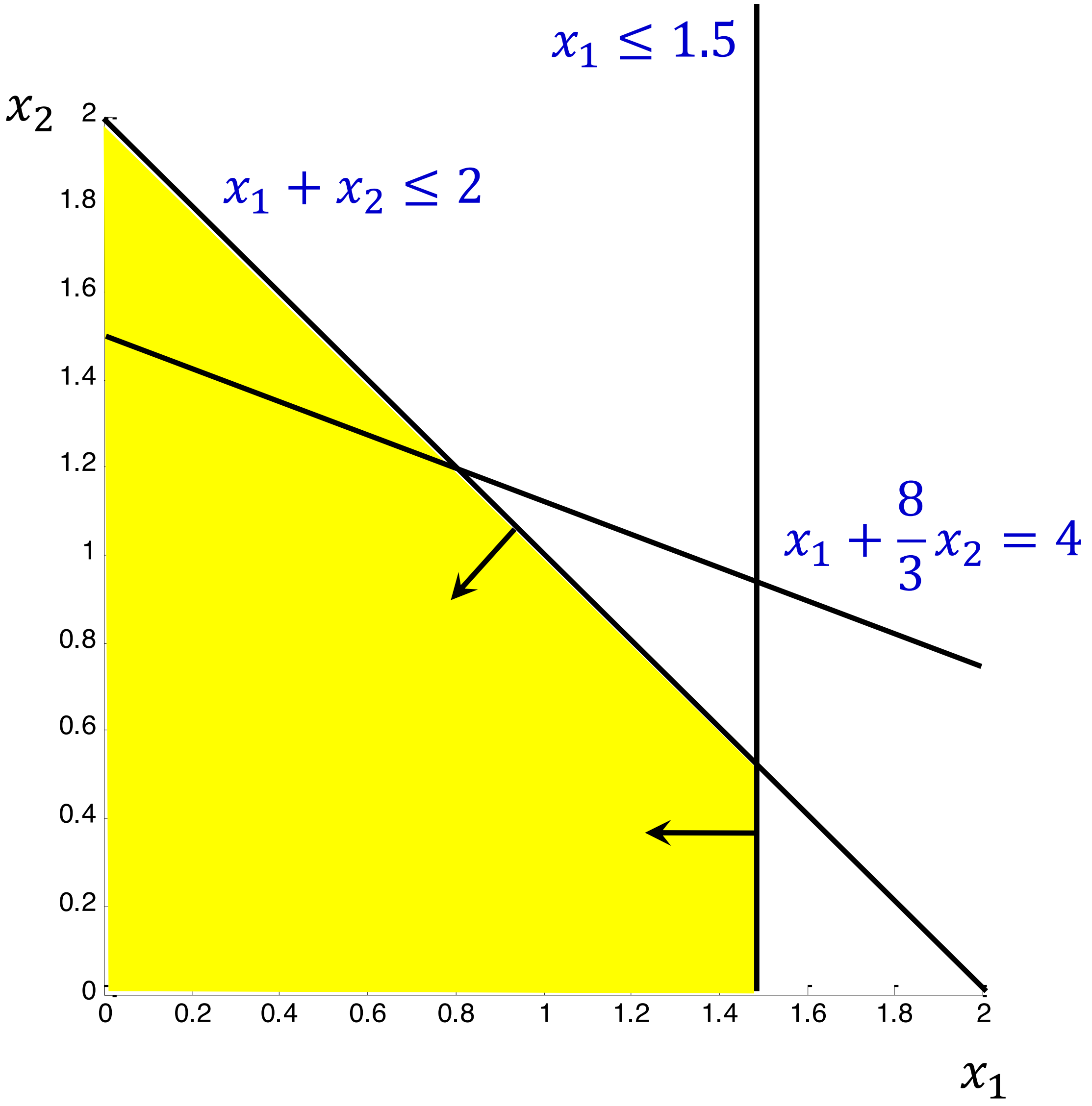
Each constraint gives a boundary.
Each boundary is a hyper-plane.

Feasible Region

Inequality constraints

$$\begin{aligned} x_1 + x_2 &\leq 2 \\ x_1 &\leq 1.5 \\ x_1 + \frac{8}{3}x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$

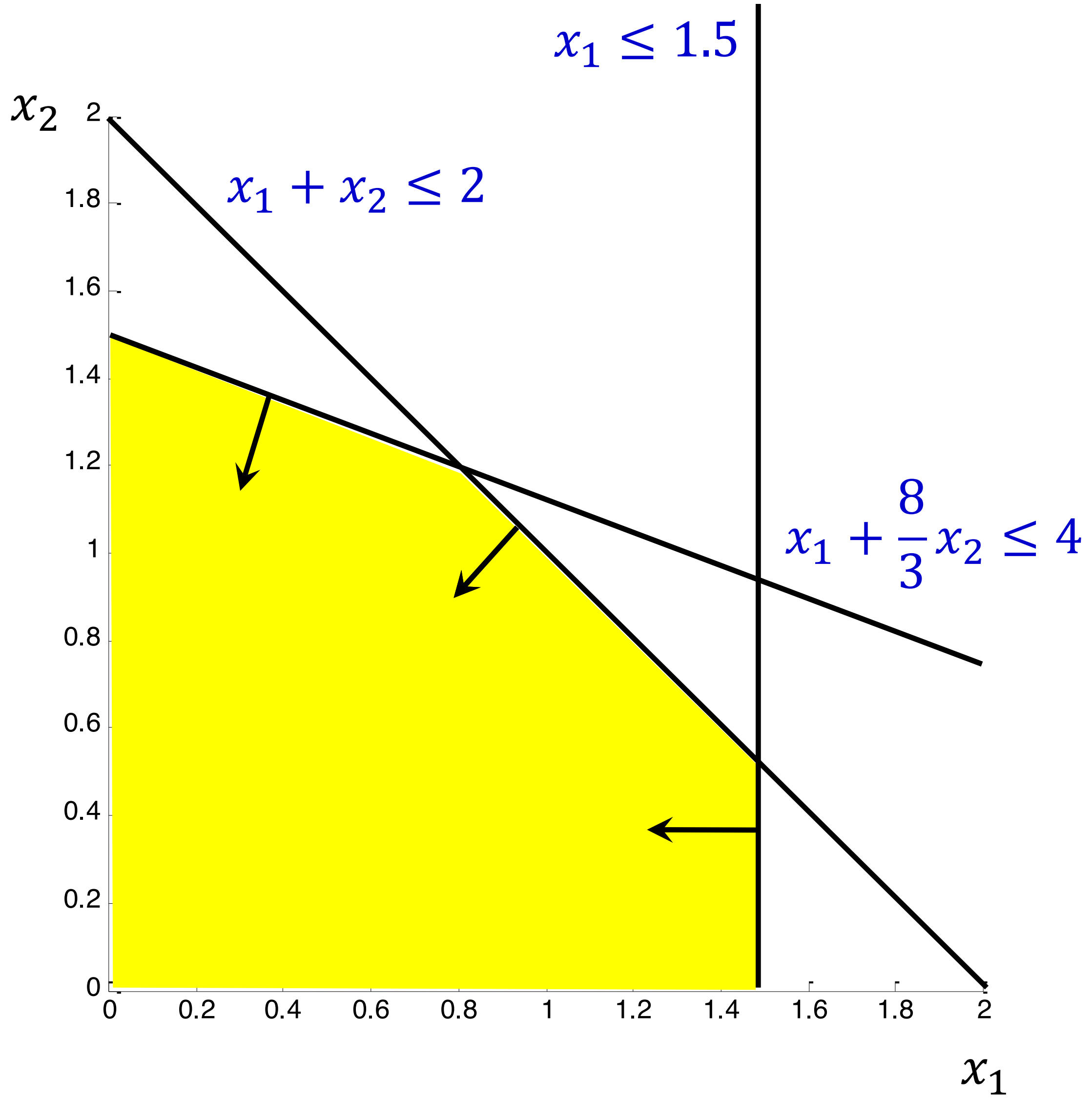
Each constraint gives a boundary.
Each boundary is a hyper-plane.



Feasible Region

Inequality constraints

$$\begin{aligned} x_1 + x_2 &\le 2 \\ x_1 &\le 1.5 \\ x_1 + \frac{8}{3}x_2 &\le 4 \\ x_1, x_2 &\ge 0 \end{aligned}$$



Each constraint gives a boundary.
Each boundary is a hyper-plane.

LP Example

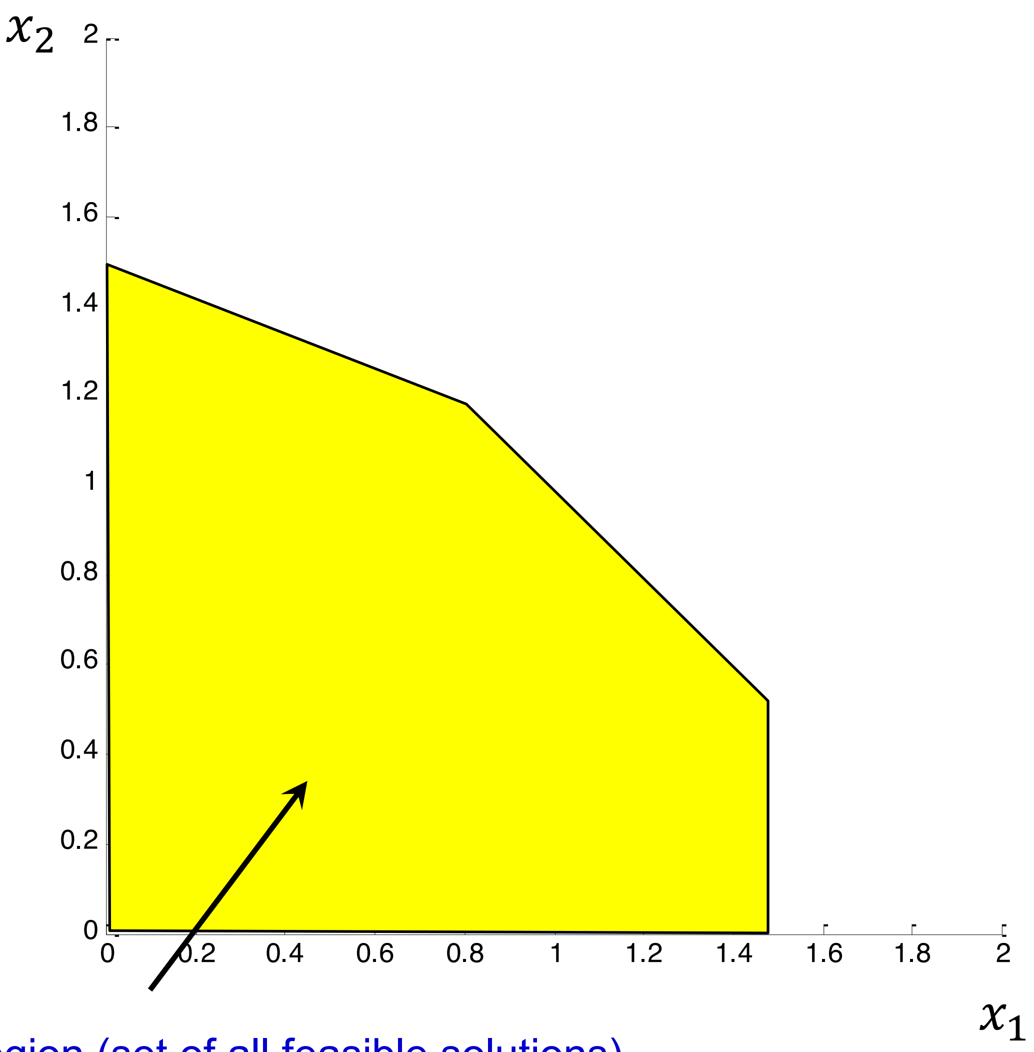
$$\max_{x_1,x_2} f(x_1, x_2)$$

Cost function

$$f(x_1, x_2) = 3x_1 + 4x_2$$

Inequality constraints

$$\begin{aligned} x_1 + x_2 &\leq 2 \\ x_1 &\leq 1.5 \\ x_1 + \frac{8}{3}x_2 &\leq 4 \\ x_1, x_2 &\geq 0 \end{aligned}$$



feasible region (set of all feasible solutions)

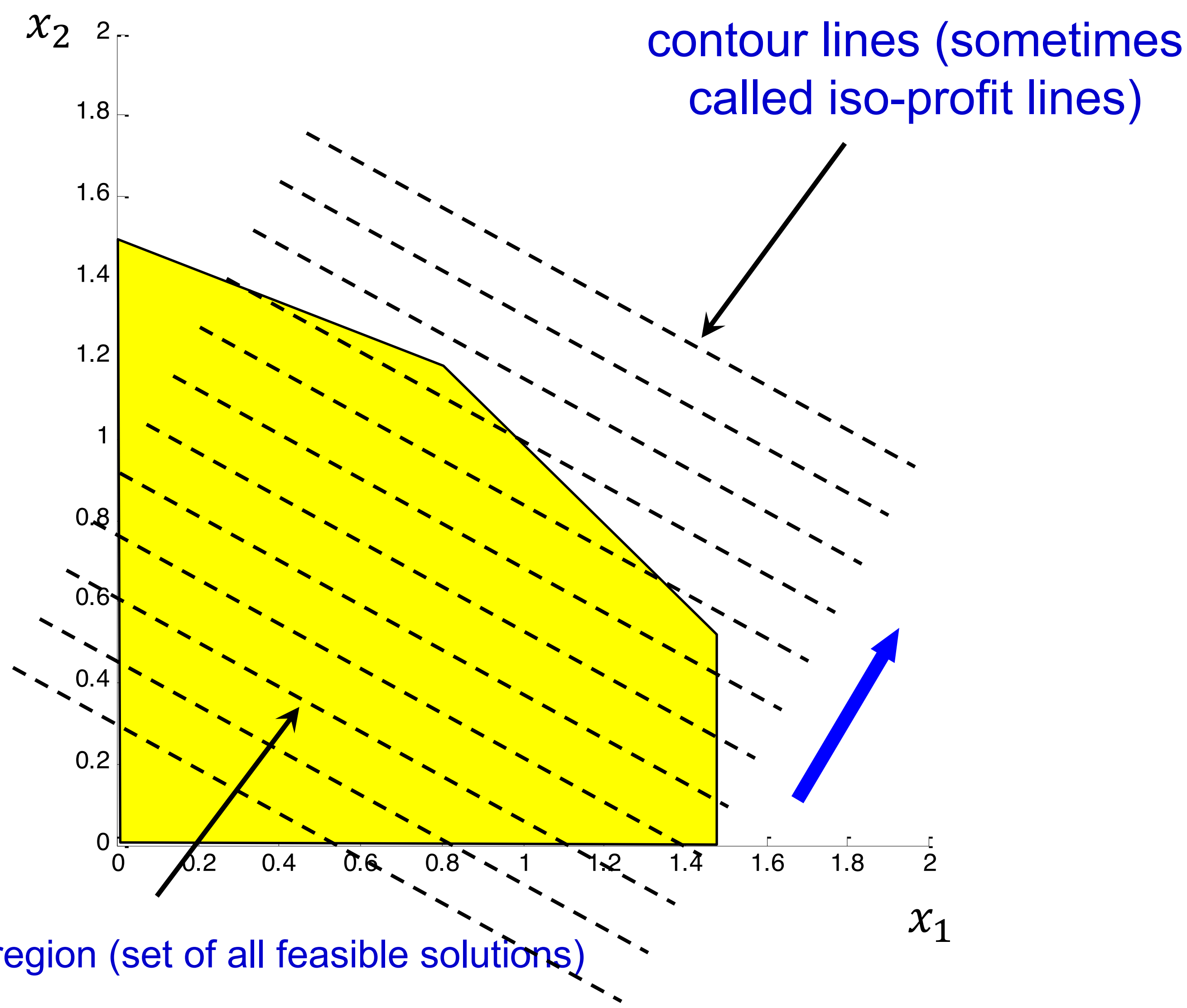
LP Example

Optimization:
Move the contour line

$$3x_1 + 4x_2 = c$$

as far up as possible, as long as it still touches the polygon.

The higher *up* the line, the higher the value of c, which we are trying to maximise.



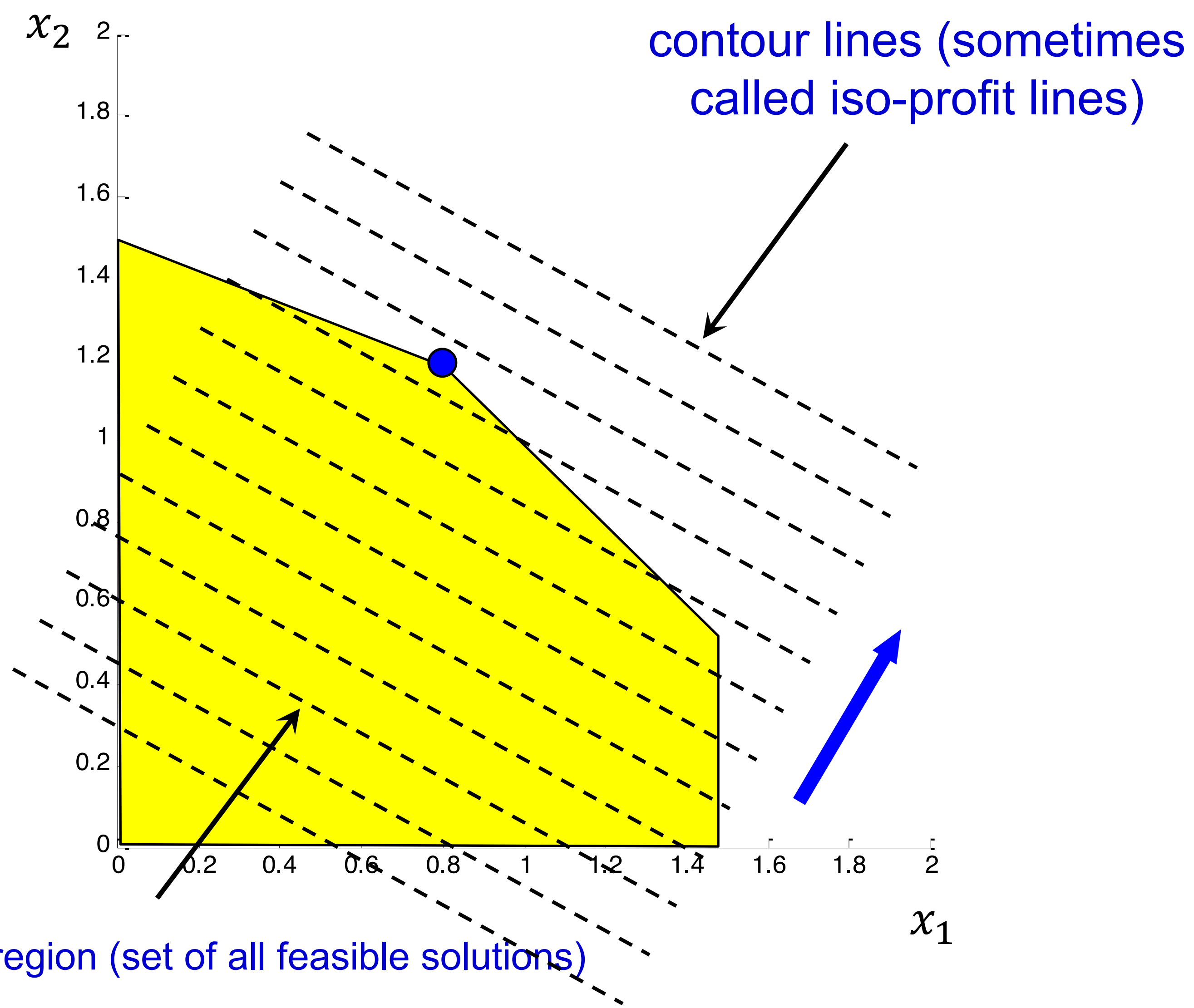
LP Example

Optimization:
Move the contour line

$$3x_1 + 4x_2 = c$$

as far up as possible, as long as it still touches the polygon.

Optimal point is therefore a vertex in the polygon.



LP Example – Change of Cost Function

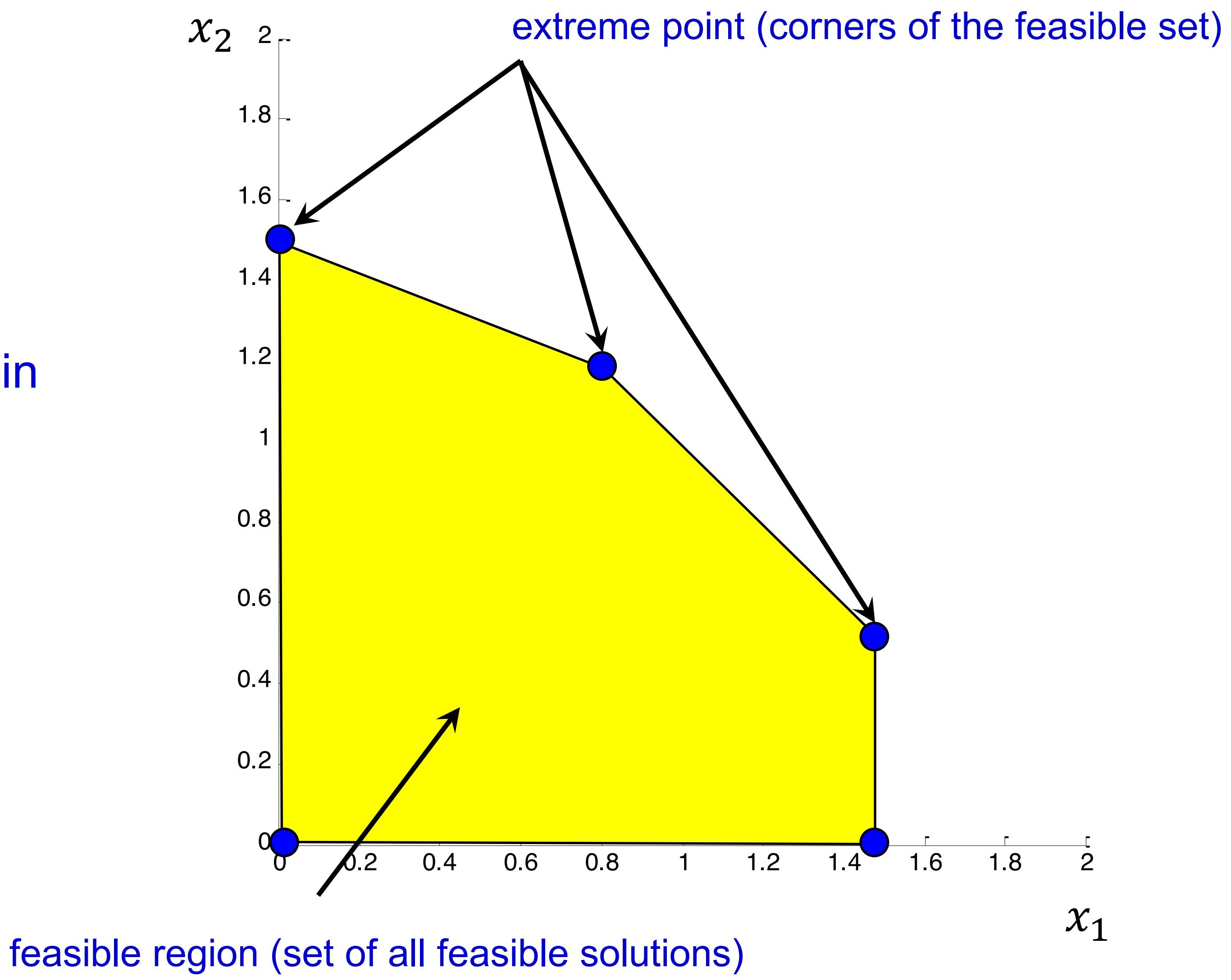
Optimization:
Move the contour line

$$3x_1 + 4x_2 = c$$

as far up as possible, as long as it still touches the polygon.

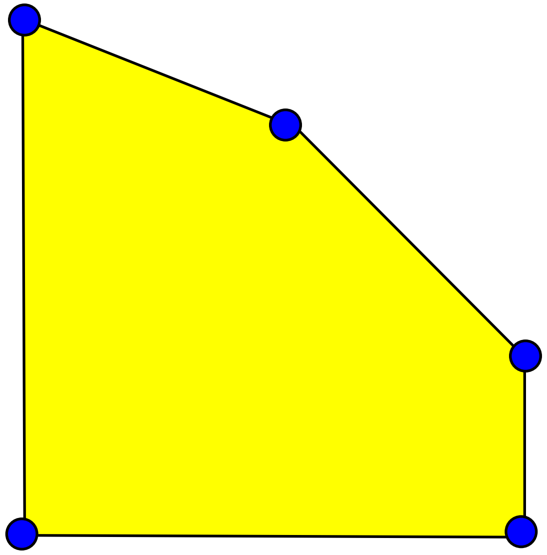
Optimal point is therefore a vertex in the polygon.

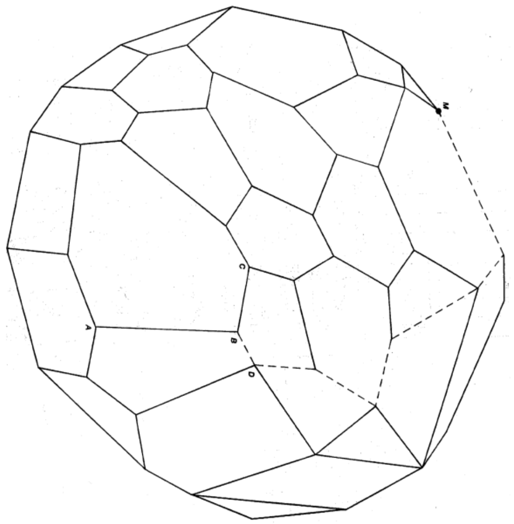
We call these the extreme points.



Linear Programming – optima at vertices

- The key point is that for any (linear) objective function the optima only occur at the corners (vertices) of the feasible polygonal region (never on the interior region).
- Similarly, in 3D the optima only occur at the vertices of a polyhedron (and in nD at the vertices of a polytope).
- However, the optimum is not necessarily unique: it is possible to have a set of optimal solutions covering an edge or face of a polyhedron.





Sketch solutions for LP optimization methods

We will look at 2 non-geometric methods to solve LP problems:

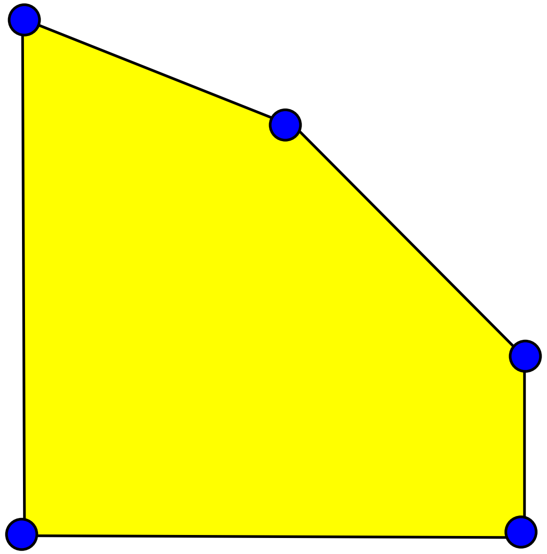
1. Simplex method:
   - Tableau exploration of vertices based on linear algebra.

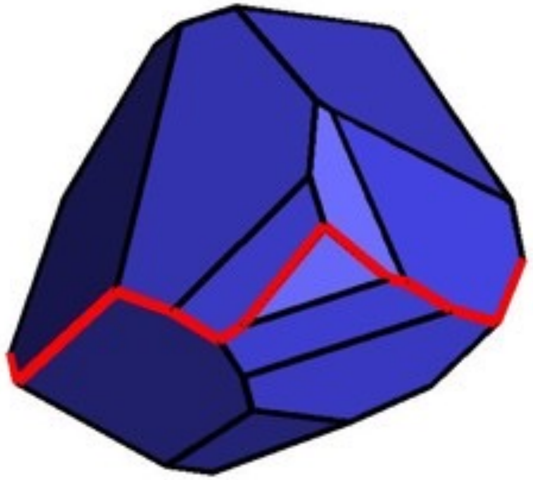
2. Interior point method:
   - Continuous optimization with constraints cast as barriers.

Simplex algorithm – solution idea

How to find the maximum?

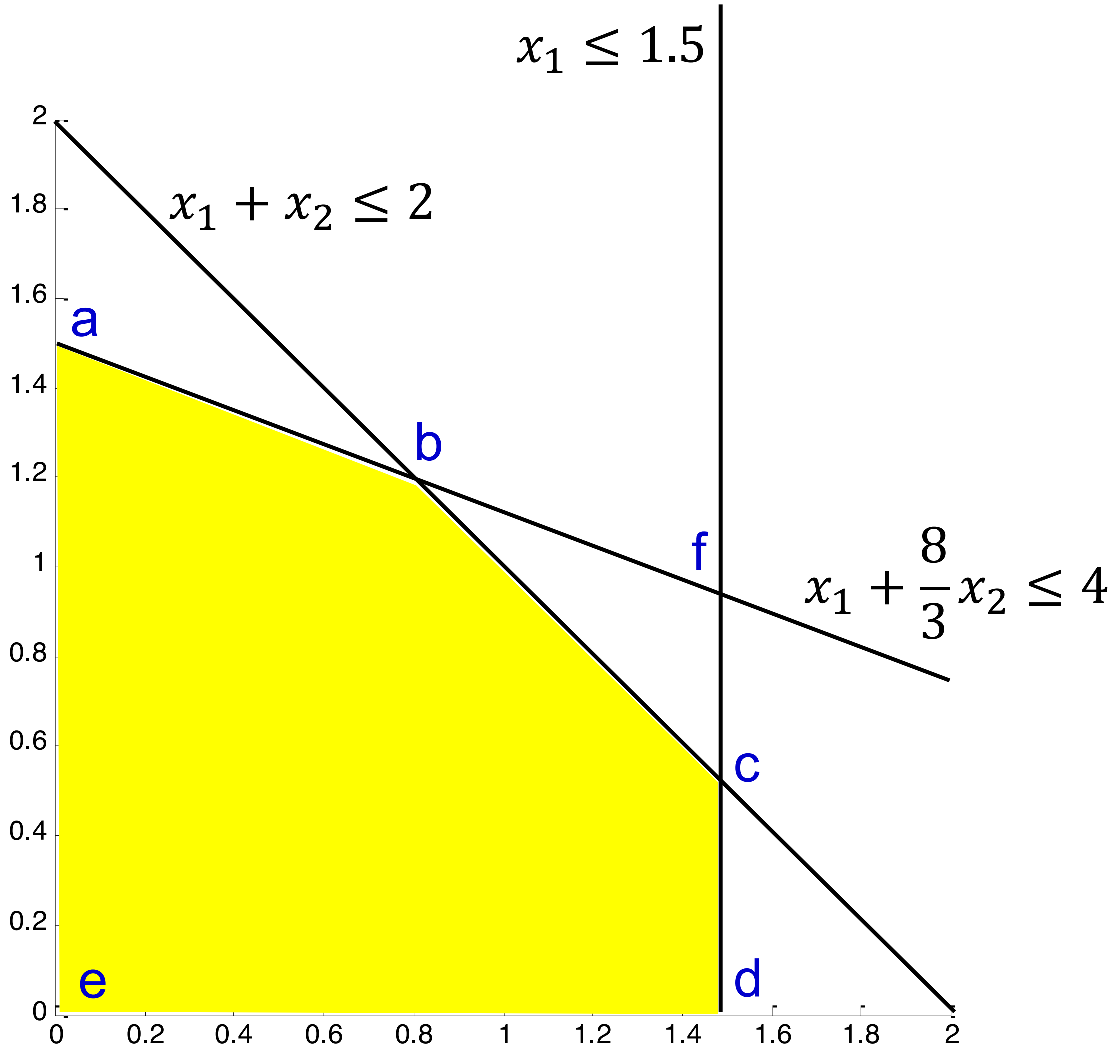
- Try every vertex? But there are too many in large problems.
- Instead, simply go from one vertex to the next increasing the cost function each time, and in an intelligent manner to avoid having to visit (and test) every vertex.
- This is the idea of the simplex algorithm.





Simplex Method

- Optimum must be at the intersection of constraints.
- Intersections are easy to find, change inequalities to equalities.
- Intersections are called basic solutions.
- Some intersections are outside the feasible region (e.g. f) and so need not be considered
- The others (which are vertices of the feasible region) are called basic feasible solutions.



Worst complexity …

There are:

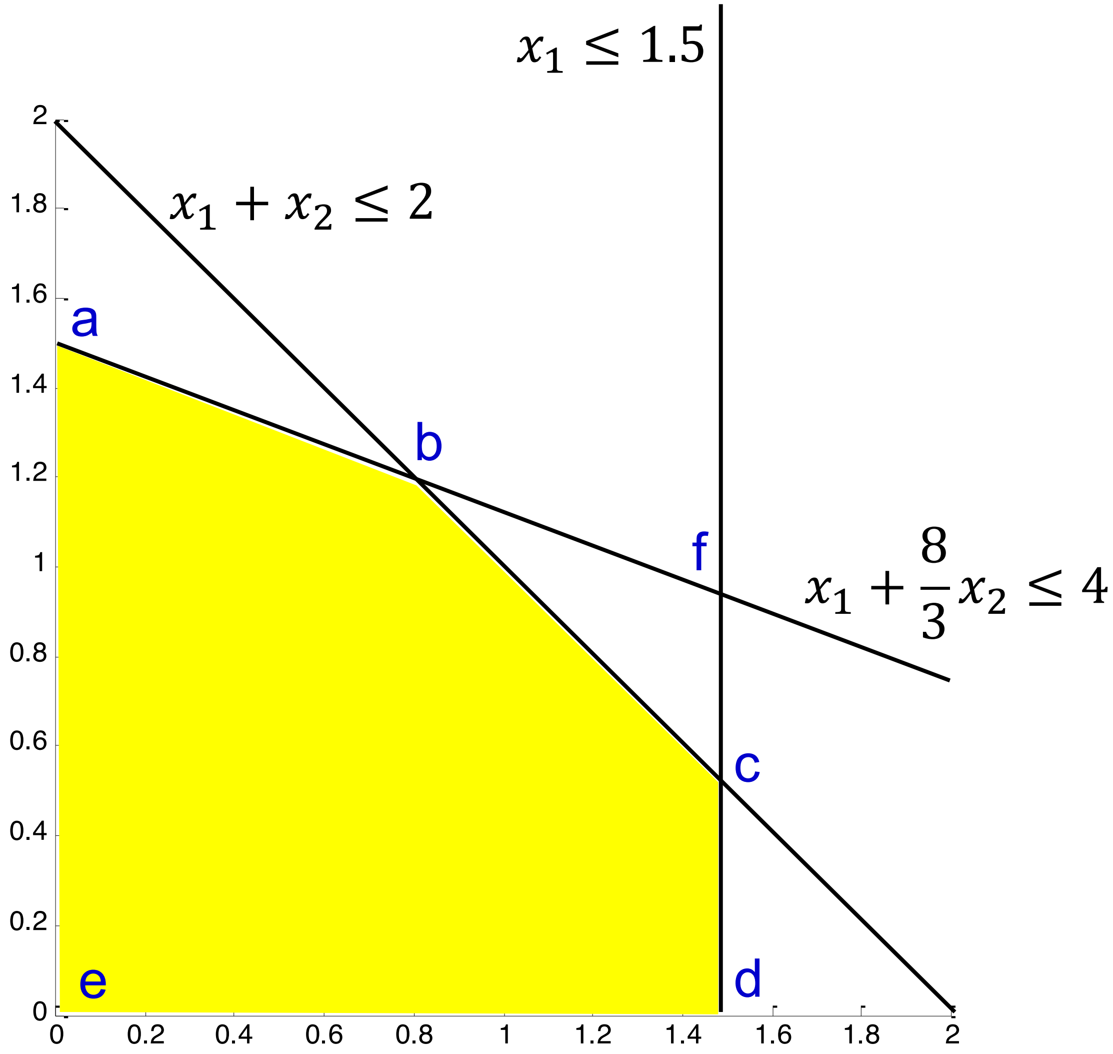
$$C_n^M = \frac{m!}{(m-n)!\,n!}$$

possible solutions, where m is the total number of constraints and n is the dimension of the space.

In this case:

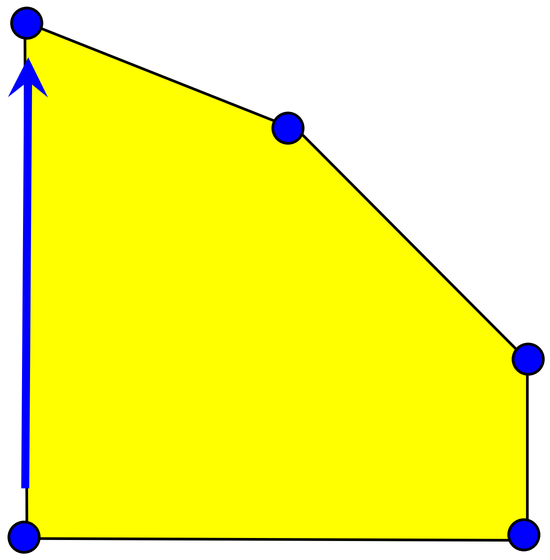
$$C_2^5 = \frac{5!}{(3)!\,2!} = 10$$

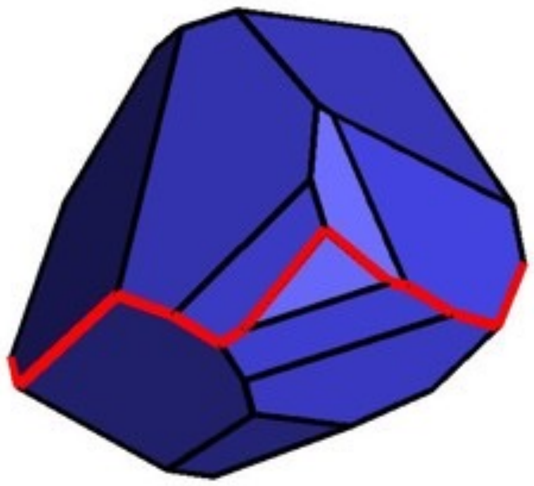
However, for large problems, the number of solutions can be huge, and it's not realistic to explore them all.



The Simplex Algorithm

- Start from a basic feasible solution (i.e. a vertex of feasible region).
- Consider all the vertices connected to the current one by an edge.
- Choose the vertex which increases the cost function the most (and is still feasible of course).
- Repeat until no further increases are possible.





In practice this is very efficient and avoids visiting all vertices

Matlab LP Function linprog

Linprog() for medium scale problems uses the Simplex algorithm

Example

Find $\mathbf{x}$ that optimizes

$$f(\mathbf{x}) = -5x_1 - 4x_2 - 6x_3$$

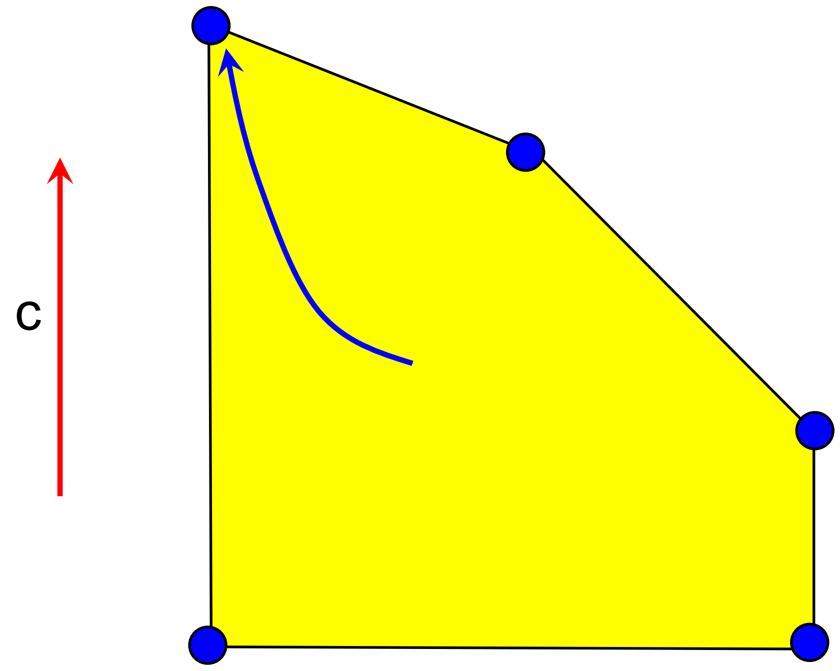
subject to

$$\begin{aligned} x_1 - x_2 + x_3 &\leq 20 \\ 3x_1 + 2x_2 + 4x_3 &\leq 42 \\ 3x_1 + 2x_2 &\leq 30 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

```
>> f = [-5; -4; -6];
>>  A = [1 -1 1
         3 2 4
         3 2 0];
>> b = [20; 42; 30];
>> lb = zeros(3,1);
>> x = linprog(f,A,b,[],[],lb);
>> Optimization terminated.
>> x
x =
   0.0000
  15.0000
   3.0000
```

Interior Point Method

- Solve LP using continuous optimization methods.
- Represent inequalities by barrier functions.
- Follow path through interior of feasible region to vertex.



Barrier function method

We wish to solve the following problem:

$$\begin{aligned} &\text{Maximise} && f(\mathbf{x}) = \mathbf{c}^T\mathbf{x} \\ &\text{subject to} && \mathbf{a}_\mathrm{i}^T\mathbf{x} \leq b_i, i = 1, \dots, m \end{aligned}$$

Problem can be rewritten as:

$$\text{Maximise } f(\mathbf{x}) - \sum_{i=1}^{m} I\left(\mathbf{a}_i^T\mathbf{x} - b_i\right)$$

where I is the indicator function:

$$I(u) = \begin{cases} 0, \text{for } u \leq 0 \\ \infty, \text{for } u > 0 \end{cases}$$

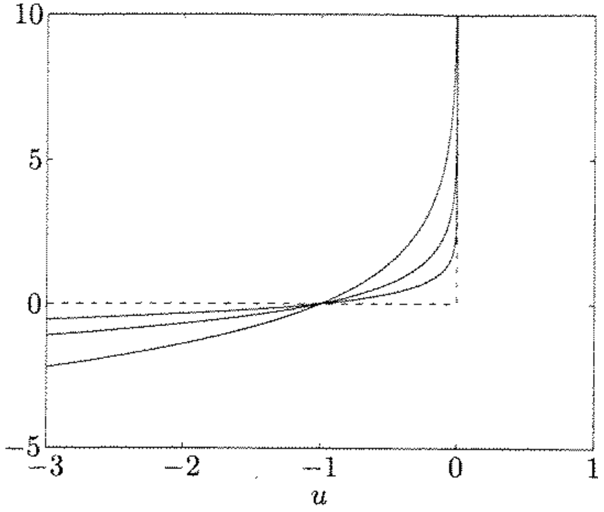
Approximation via logarithmic barrier functon

Approximate indicator function by (differentiable) logarithmic barrier:

$$\text{Maximise } f(\mathbf{x}) - \frac{1}{t}\sum_{i=1}^{m} \log(-\mathbf{a}_i^T\mathbf{x} + b_i)$$

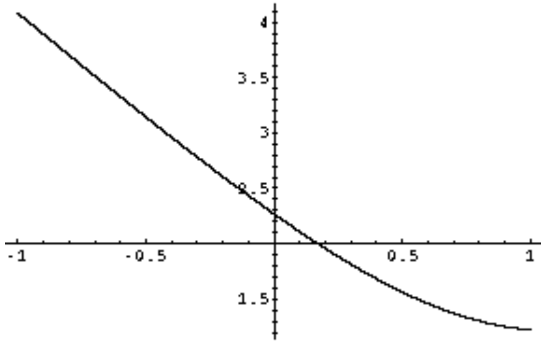
- For $t > 0$, $-\left(\frac{1}{t}\right)\log(-u)$ is a smooth approximation of $I(u)$.
- Approximation improves as $t \to \infty$.

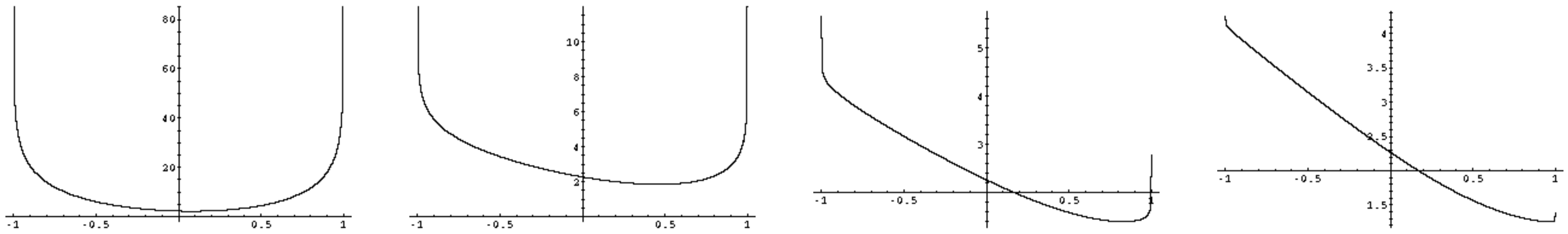
$$\frac{-\log(-u)}{t}$$



Barrier Method – Example

Function $f(x)$ to be minimized subject to $x \geq -1$ and $x \leq 1$.





Function $f(x) - \frac{1}{t}\log(1 - x^2)$ for increasing values of t

Algorithm for Interior Point Method

Problem becomes:

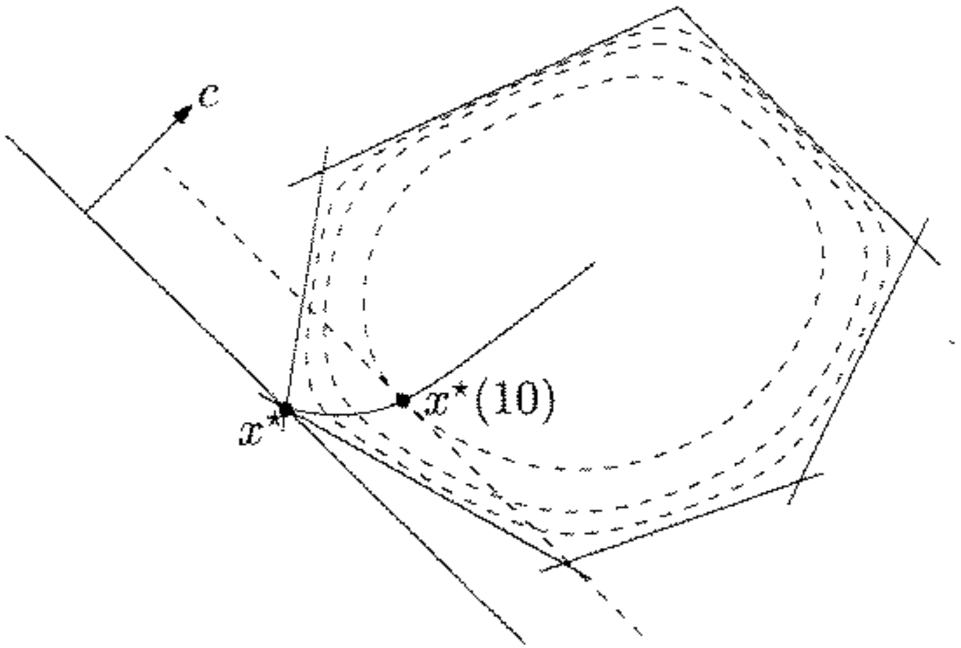
$$\text{Maximise } f(\mathbf{x}) - \frac{1}{t}\sum_{i=1}^{m}\log\left(-\mathbf{a}_i^T\mathbf{x} + b_i\right) = tf(\mathbf{x}) - \sum_{i=1}^{m}\log\left(-\mathbf{a}_i^T\mathbf{x} + b_i\right)$$

Algorithm:

- Solve using (e.g.) Newton's method.
- $t \to \mu t$.
- Repeat until convergence.

As t increases, this converges to the solution of the original problem.

Algorithm for Interior Point Method



Trace of the central path: optimum for varying values of t.

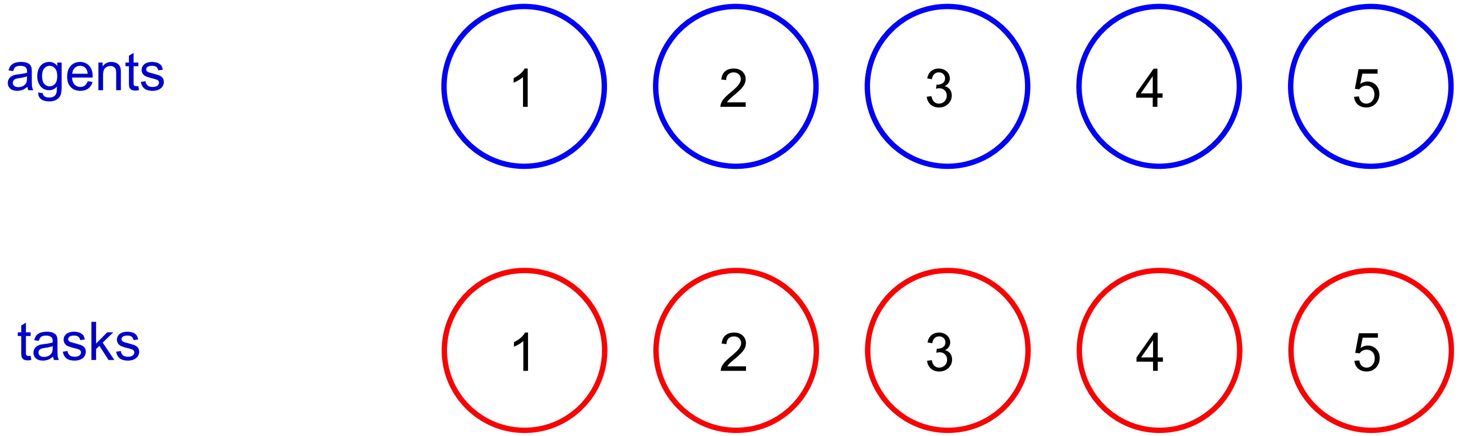
Diagram copied from Boyd and Vandenberghe

Integer Programming

- There are often situations where the solution is required to be an integer or have boolean values (0 or 1 only).
- For example:
  - assignment problems;
  - scheduling problems;
  - matching problems.
- Linear programming can also be used for these cases.

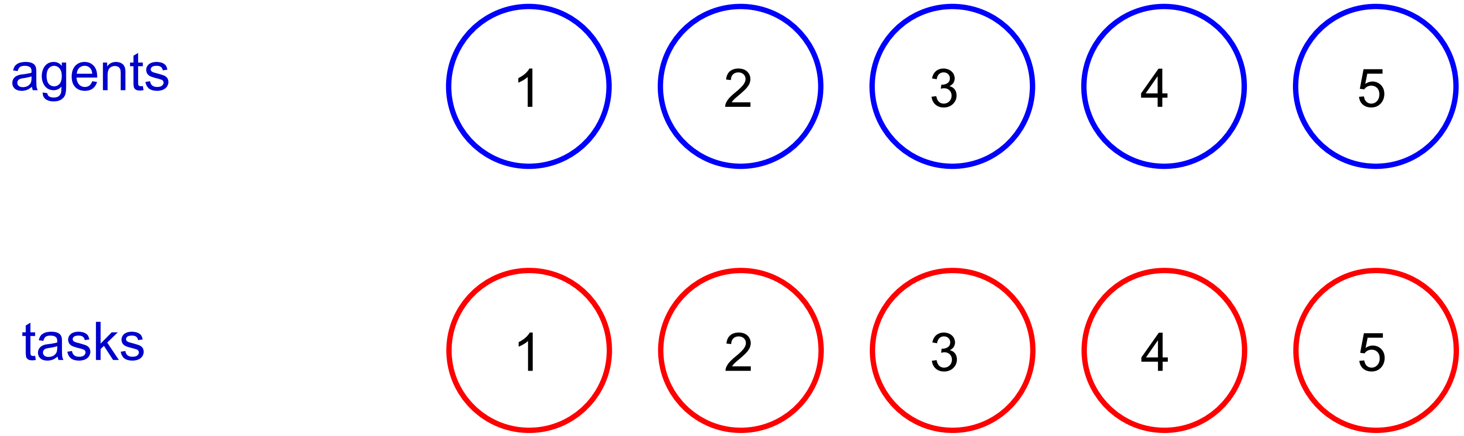
The air-crew scheduling problem is a good example of an integer programming problem, since only whole pilots and cabin staff are valid solutions.

Integer Programming Example: Assignment Problem



e.g. 3 taxis, 3 people, assign taxis so as to minimize time to pick up, e.g. nearest taxis.

Integer Programming Example: Assignment Problem



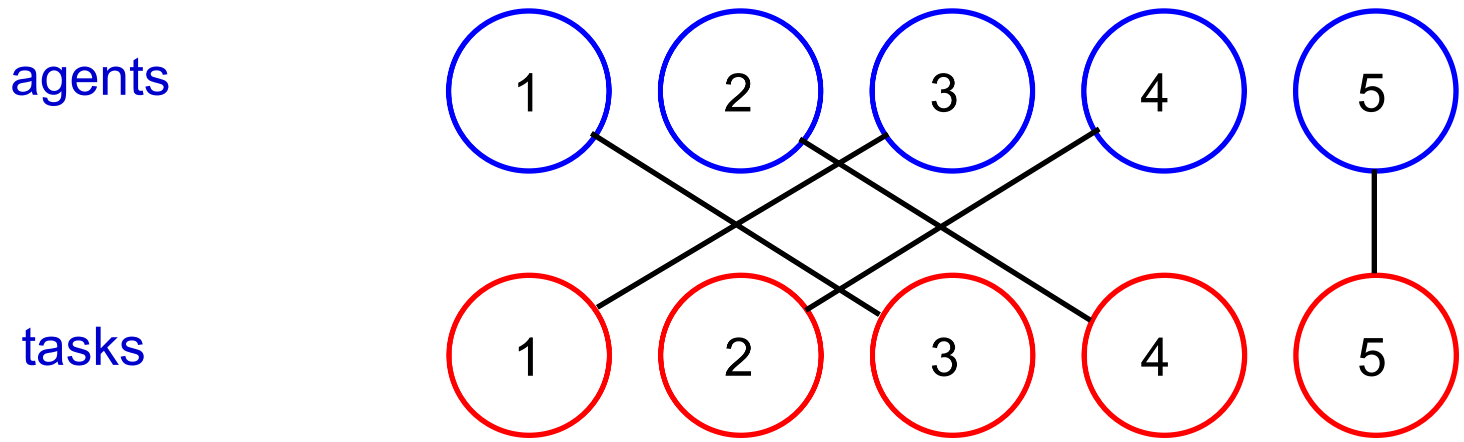
Objective:

- assign n agents to n tasks to minimize total cost - e.g. 3 taxis, 3 people, assign taxis so as to minimize time to pick up, e.g. nearest taxis.

Assignment constraints:

- each agent assigned to one task only.
- each task assigned to one agent only.

Integer Programming Example: Assignment Problem



Objective:

- assign n agents to n tasks to minimize total cost - e.g. 3 taxis, 3 people, assign taxis so as to minimize time to pick up, e.g. nearest taxis.

Assignment constraints:

- each agent assigned to one task only.
- each task assigned to one agent only.

Integer Programming Example: Assignment Problem

Problem specification

x_{ij} is the assignment of an agent i to a task j (can take values 0 or 1).
c_{ij} is the (non-negative) cost of assigning agent i to task j.

Example solution for x_{ij} →

$$
\begin{array}{c|ccccc}
 & & & \text{tasks } j & & \\
\text{agents } i & 0 & 1 & 0 & 0 & 0 \\
 & 0 & 0 & 0 & 1 & 0 \\
 & 0 & 0 & 1 & 0 & 0 \\
 & 0 & 0 & 0 & 0 & 1 \\
 & 1 & 0 & 0 & 0 & 0
\end{array}
$$

Each agent i assigned to one task only

- only one entry in each row

Each task j assigned to one agent only

- only one entry in each column

Integer Programming Example: Assignment Problem: Cost Matrix

tasks

agents

	1'	2'	3'	4'	5'
1	3	8	9	15	10
2	4	10	7	16	14
3	9	13	11	19	10
4	8	13	12	20	13
5	1	7	5	11	9

cost = 3 + 10 + 11 + 20 + 9 =53

tasks

agents

	1'	2'	3'	4'	5'
1	3	8	9	15	10
2	4	10	7	16	14
3	9	13	11	19	10
4	8	13	12	20	13
5	1	7	5	11	9

cost = 8 + 7 + 20 + 8 + 11 = 44

Cost matrix e.g. 5 jobs/tasks of different length, 5 workers diff cost per hour

Integer Programming Example: Assignment Problem

Linear Programme formulation:

$$\min_{x_{ij}} f(\mathbf{x}) = \sum_{ij} x_{ij} c_{ij}$$

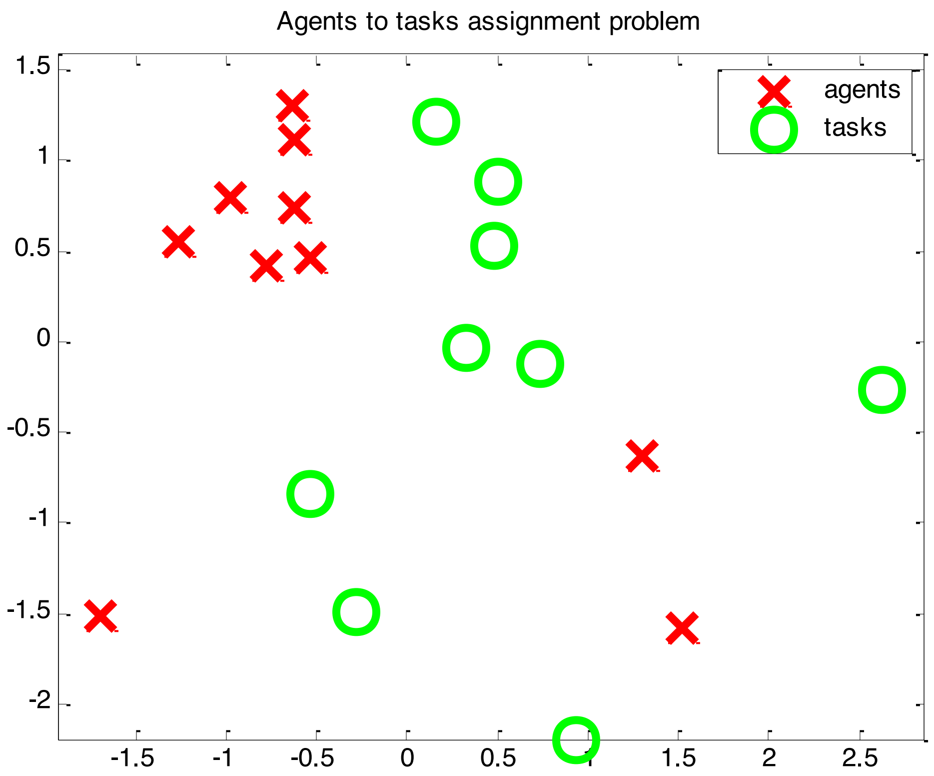
Subject to:

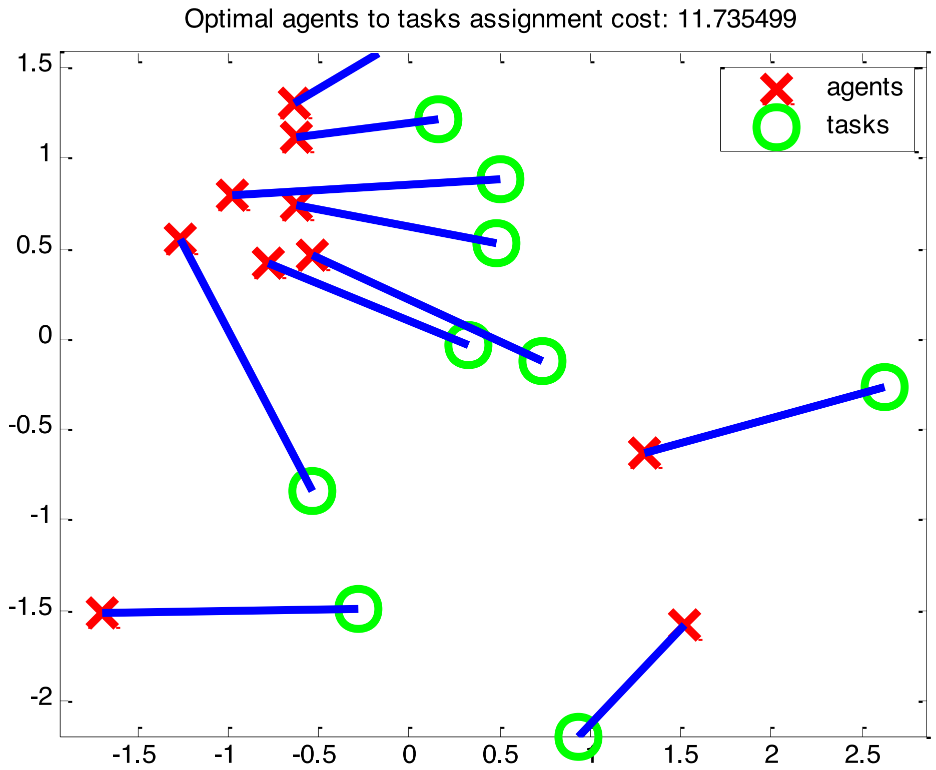
- (inequalities) $\sum_j x_{ij} \leq 1$, $\forall i$ – each agent assigned to at most one task.
- (equalities) $\sum_i x_{ij} = 1$, $\forall j$ – each task assigned to exactly one agent.

$$\begin{array}{cc} & \text{tasks } j \\ \text{agents } i & \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

This is a relaxation of the problem, because the variables x_{ij} are not forced to take boolean values.

Integer Programming Example: Assignment Problem





Cost of assignment = distance between agent and task

e.g. application: tracking, correspondence

Example Application: Tracking Pedestrians

Multiple Object Tracking using Flow Linear Programming, Berclaz, Fleuret & Fua, 2009

What is next?

- Convexity (when does a function have only a global minimum?).
- Robust cost functions.
- Optimizing non-convex functions.