Advanced Algorithmic Techniques (COMP523)

Linear Programming

Recap and plan

- **Previous 2 lectures:**
  - Polynomial time reductions
  - Computational classes: P and NP
  - NP-hardness and NP-completeness
- **This lecture:**
  - Linear Programming
  - Integer Linear Programs
  - LP-Duality
  - Total Unimodularity

A problem

- A company makes two products, X and Y, using two machines, A and B.
    - Each unit of product X requires *50min* processing time on machine A and *30min* processing time on machine B.
    - Each unit of product Y requires *24min* of processing time on machine A and *33min* of processing time on machine B.
    - At the start of the week, there are *30 units* of X and *90 units* of Y in stock.
    - The available processing time on machine A is *40 hours* and on machine B it is *35 hours*.
    - The demand for X in the week is *75 units* and for Y it is *95 units*.
- Goal: Maximise the combined sum of units of X and Y in stock at the end of the week.

A linear program

- Goal: Maximise the combined sum of units of X and Y in stock at the end of the week.

Maximise $(x + 30 - 75) + (y + 90 - 95)$

A linear program

- Each unit of product X requires *50min* processing time on machine A and *30min* processing time on machine B.
- Each unit of product Y requires *24min* of processing time on machine A and *33min* of processing time on machine B.
- At the start of the week, there are *30 units* of X and *90 units* of Y in stock.
- The available processing time on machine A is *40 hours* and on machine B it is *35 hours*.
- The demand for X in the week is *75 units* and for Y it is *95 units*.

$$50x + 24y \leq 2400 \qquad x \geq 75 - 30$$

$$30x + 33y \leq 2100 \qquad y \geq 95 - 90$$

A linear program

Maximise $x + y - 50$

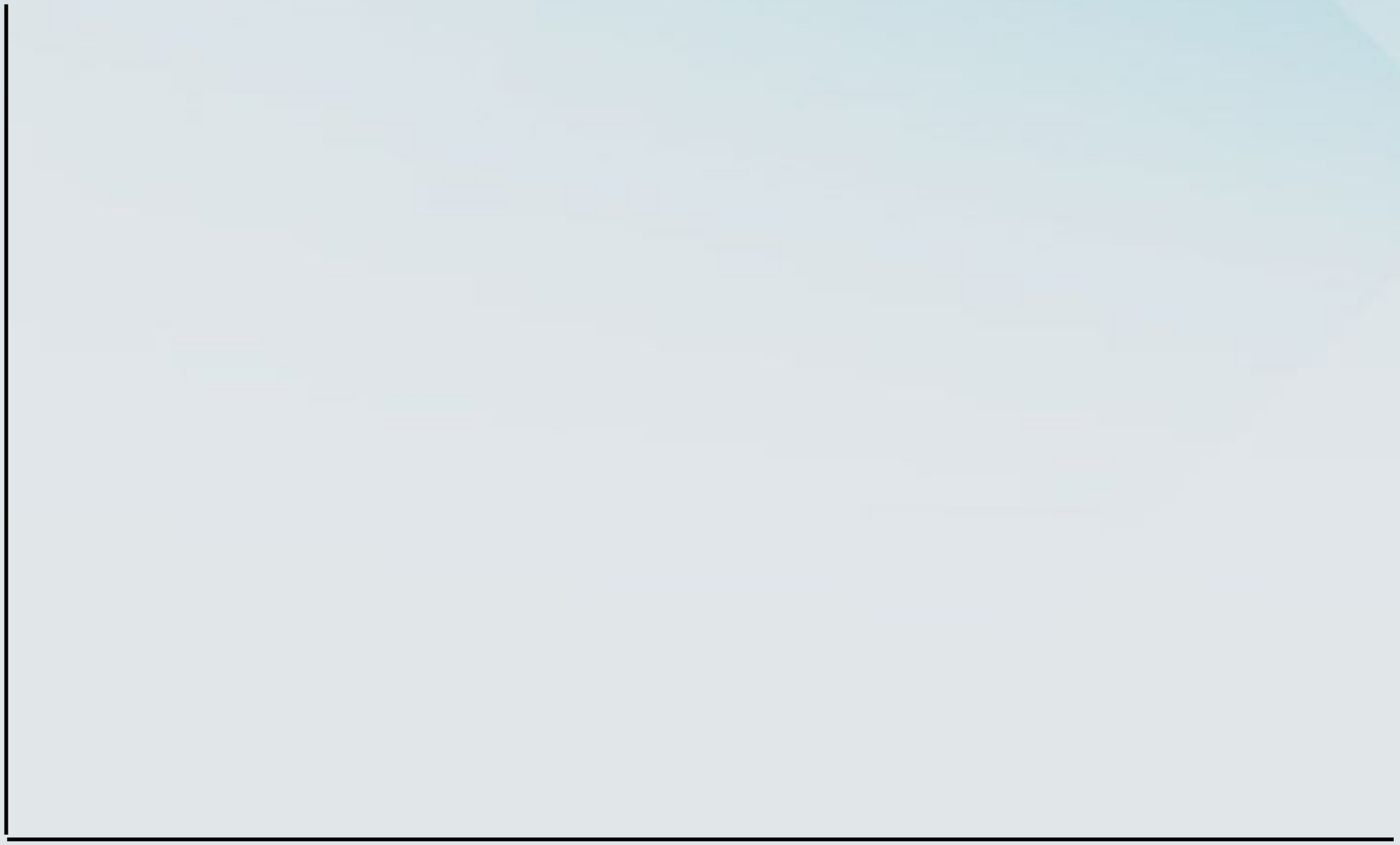
subject to $50x + 24y \leq 2400$

$30x + 33y \leq 2100$

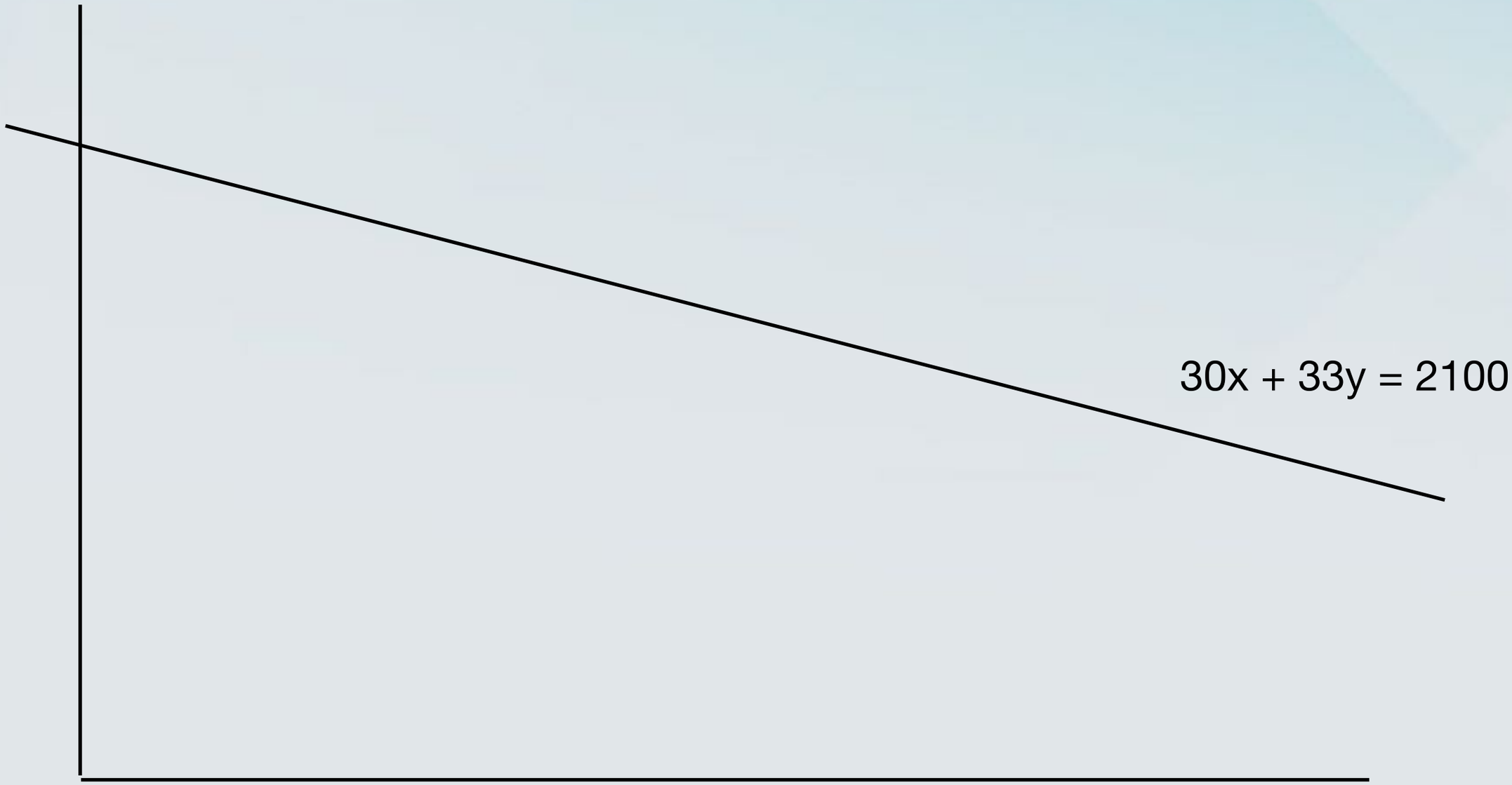
$x \geq 45$

$y \geq 5$

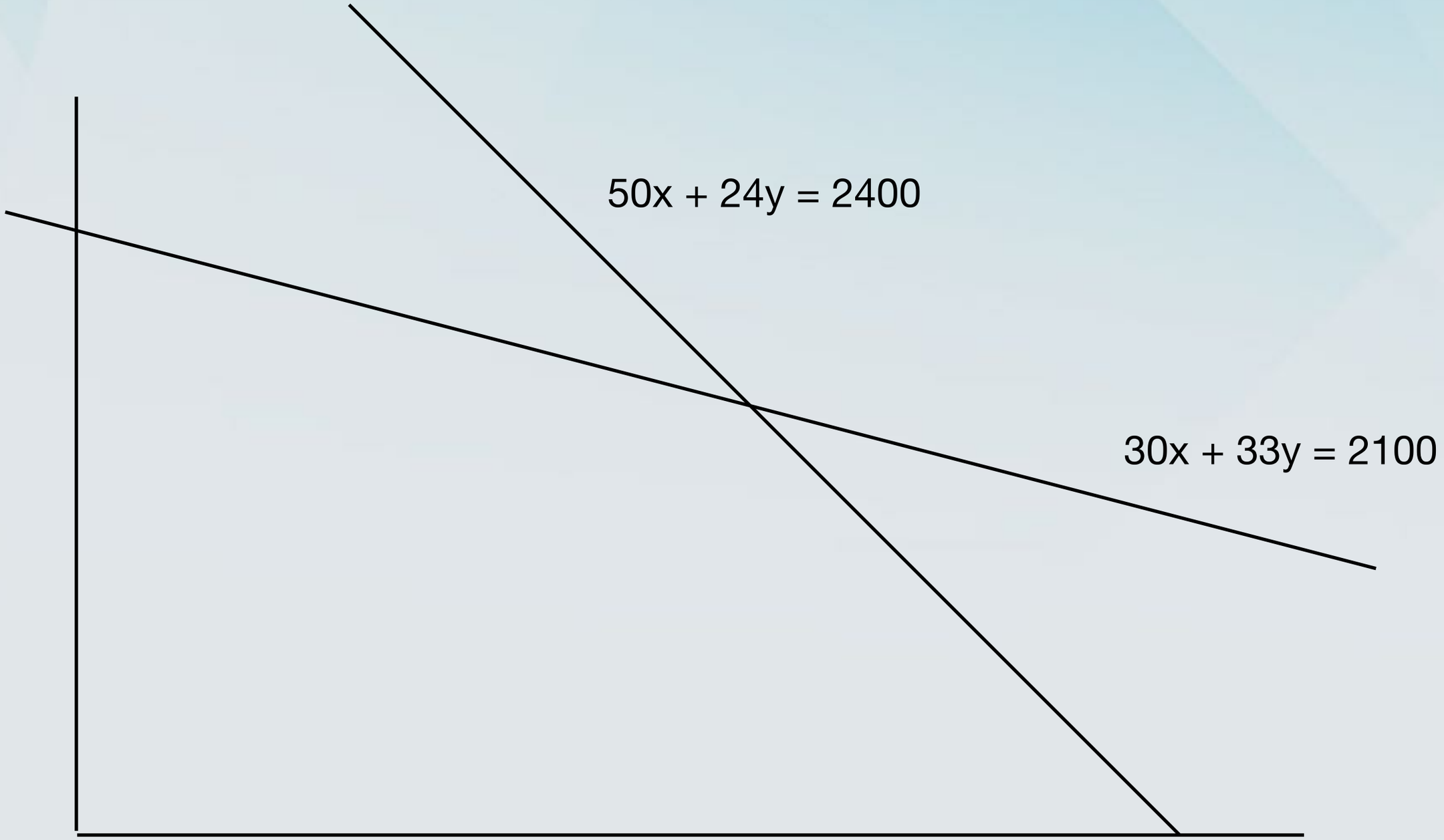
Solving the linear program



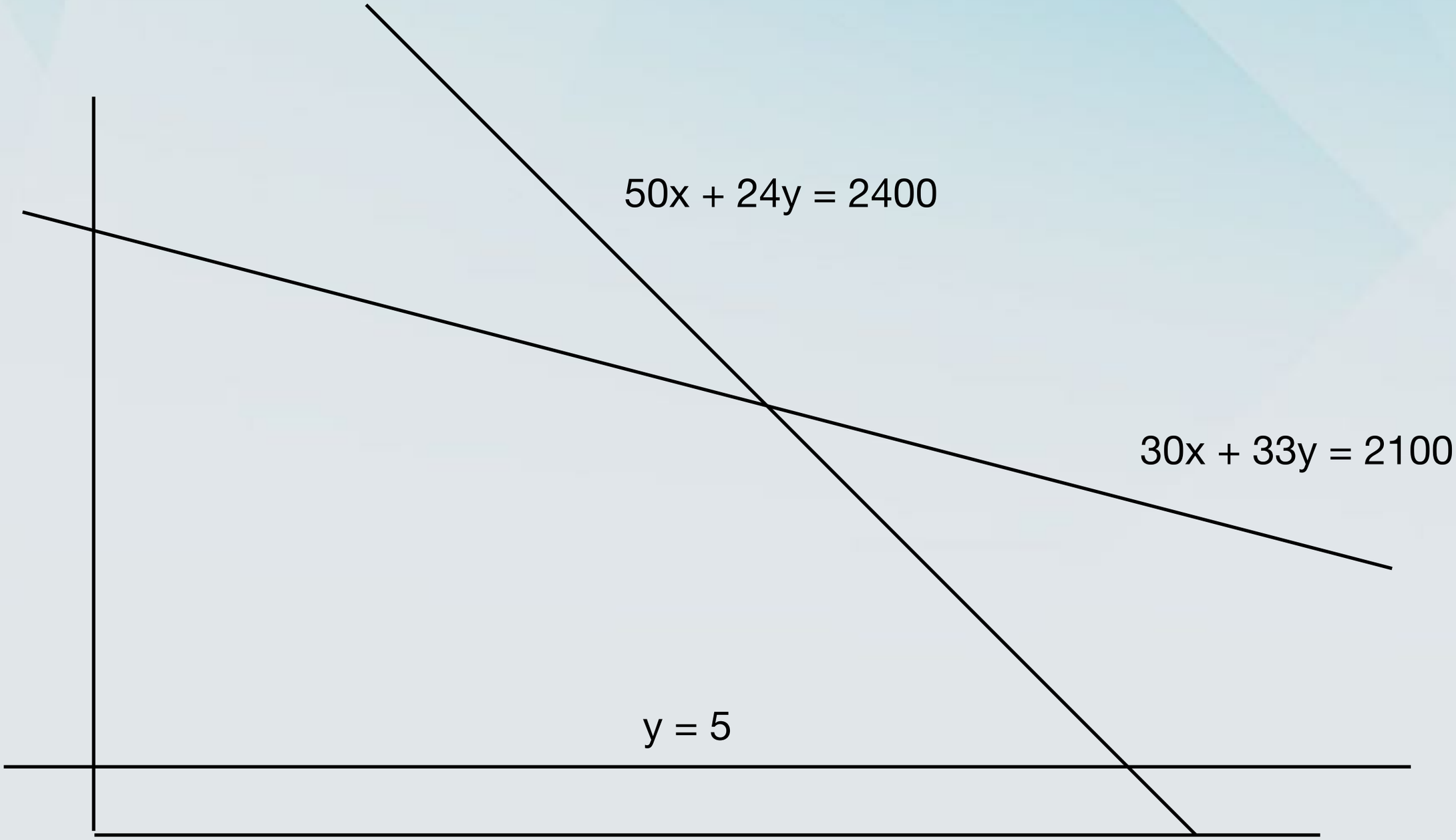
Solving the linear program



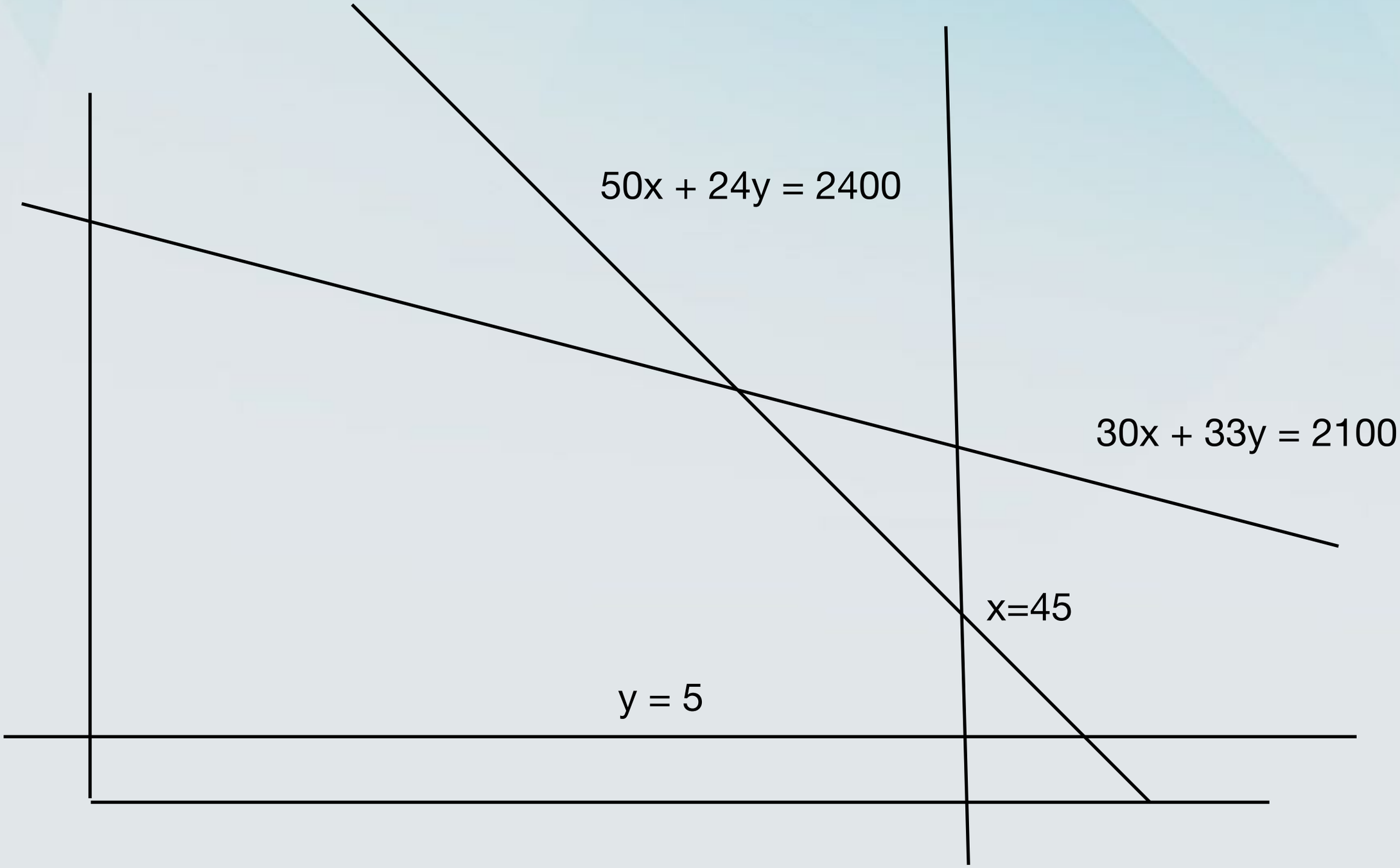
Solving the linear program



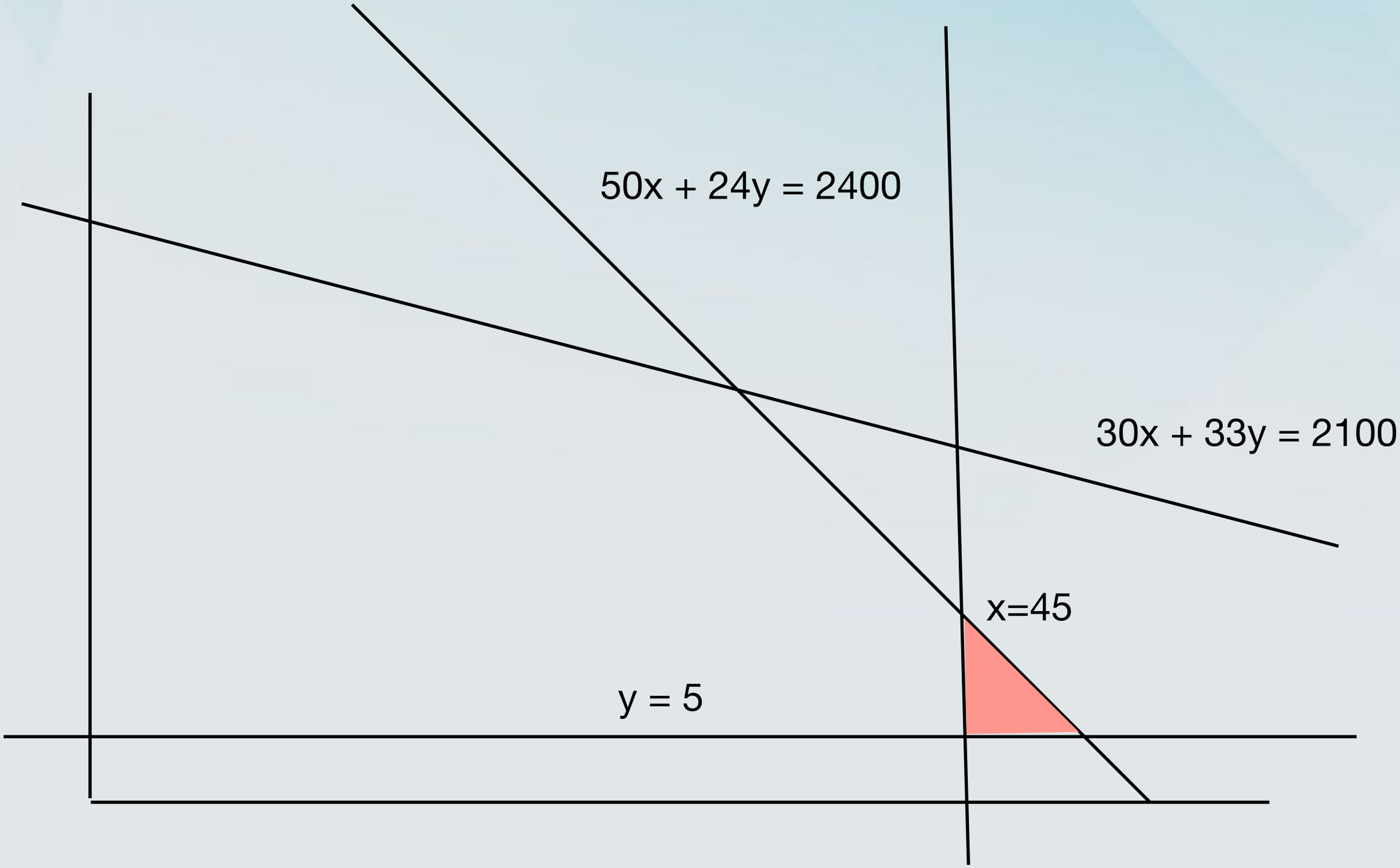
Solving the linear program



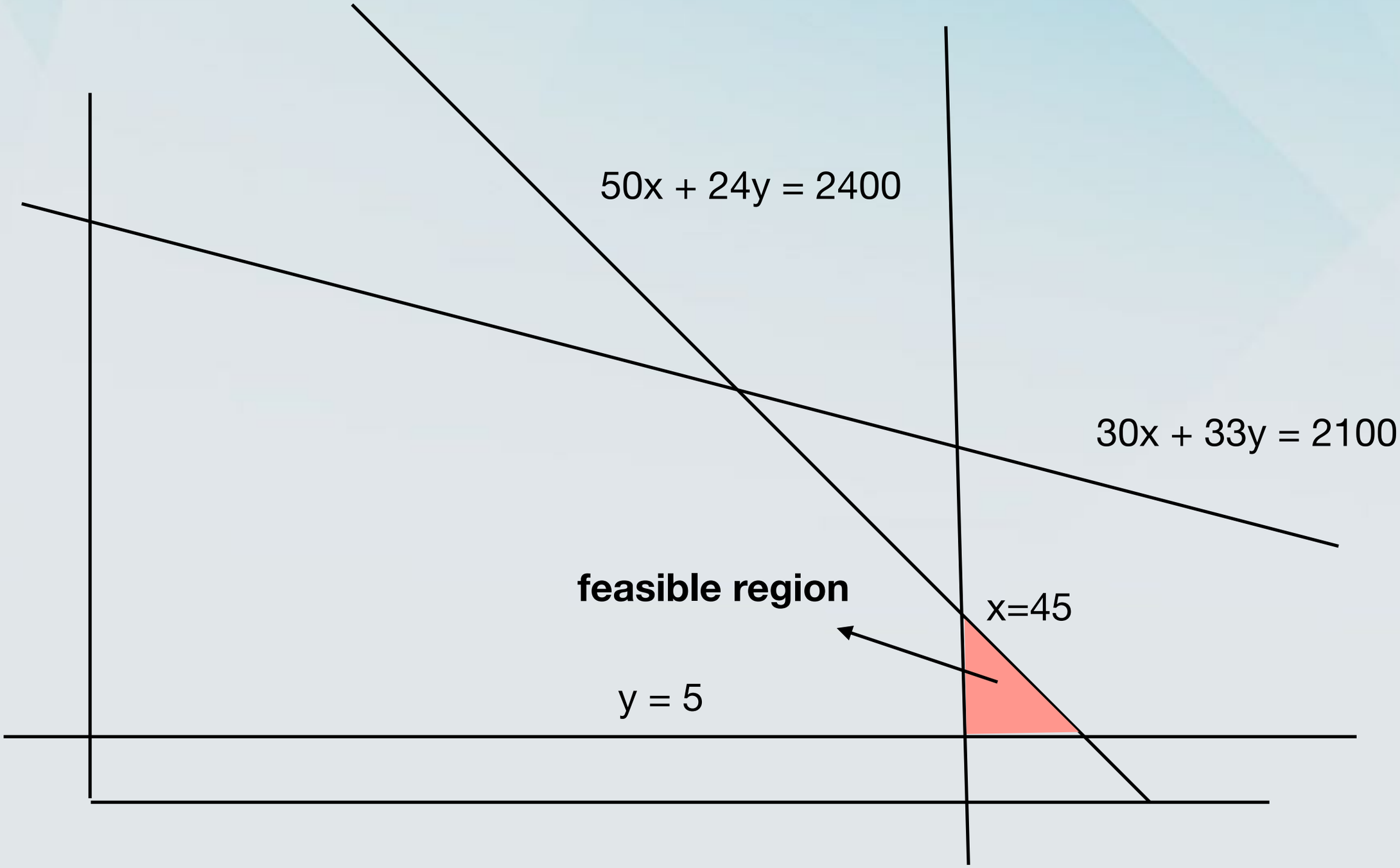
Solving the linear program



Solving the linear program



Solving the linear program



Solving the linear program

- To find the optimal solution, it suffices to examine the *corners* of the feasible region.
- These are the intersection points of the lines defined by the constraints.
  - e.g., 50x+24y - 2400 = x - 45

Linear programming

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

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

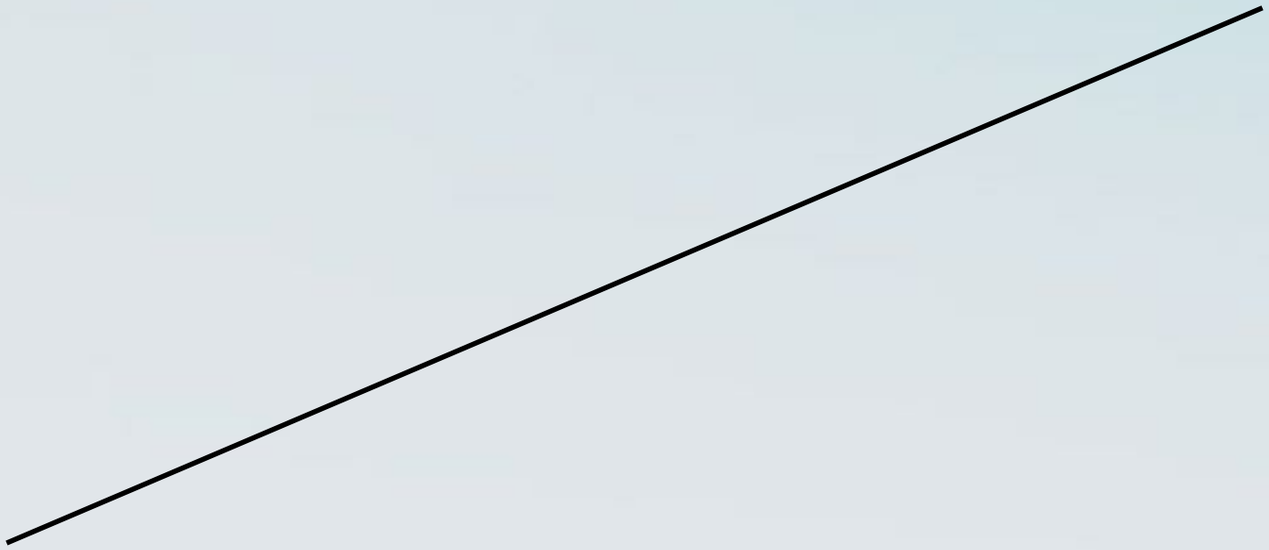
$$x_j \geq 0, \quad j = 1, ..., n$$

Linear programming (in matrix form)

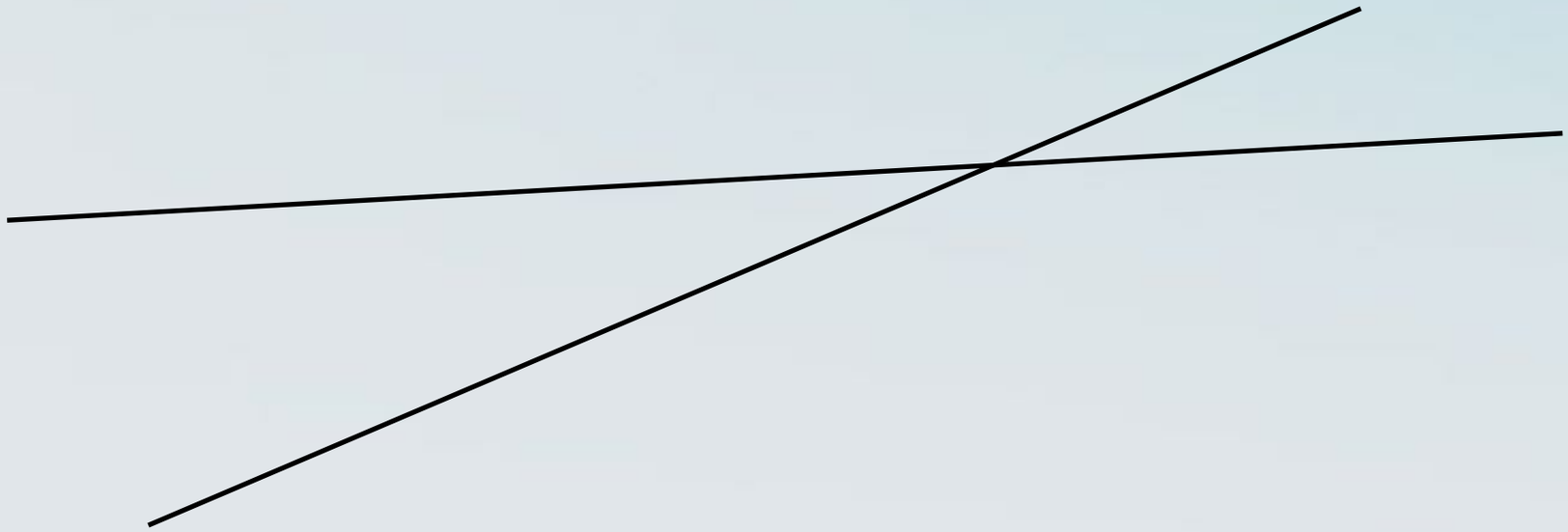
$$\begin{aligned} \text{maximise} \quad & c^{\mathrm{T}} x \\ \text{subject to} \quad & Ax \leq b, \\ & x \geq 0 \end{aligned}$$

Feasible region

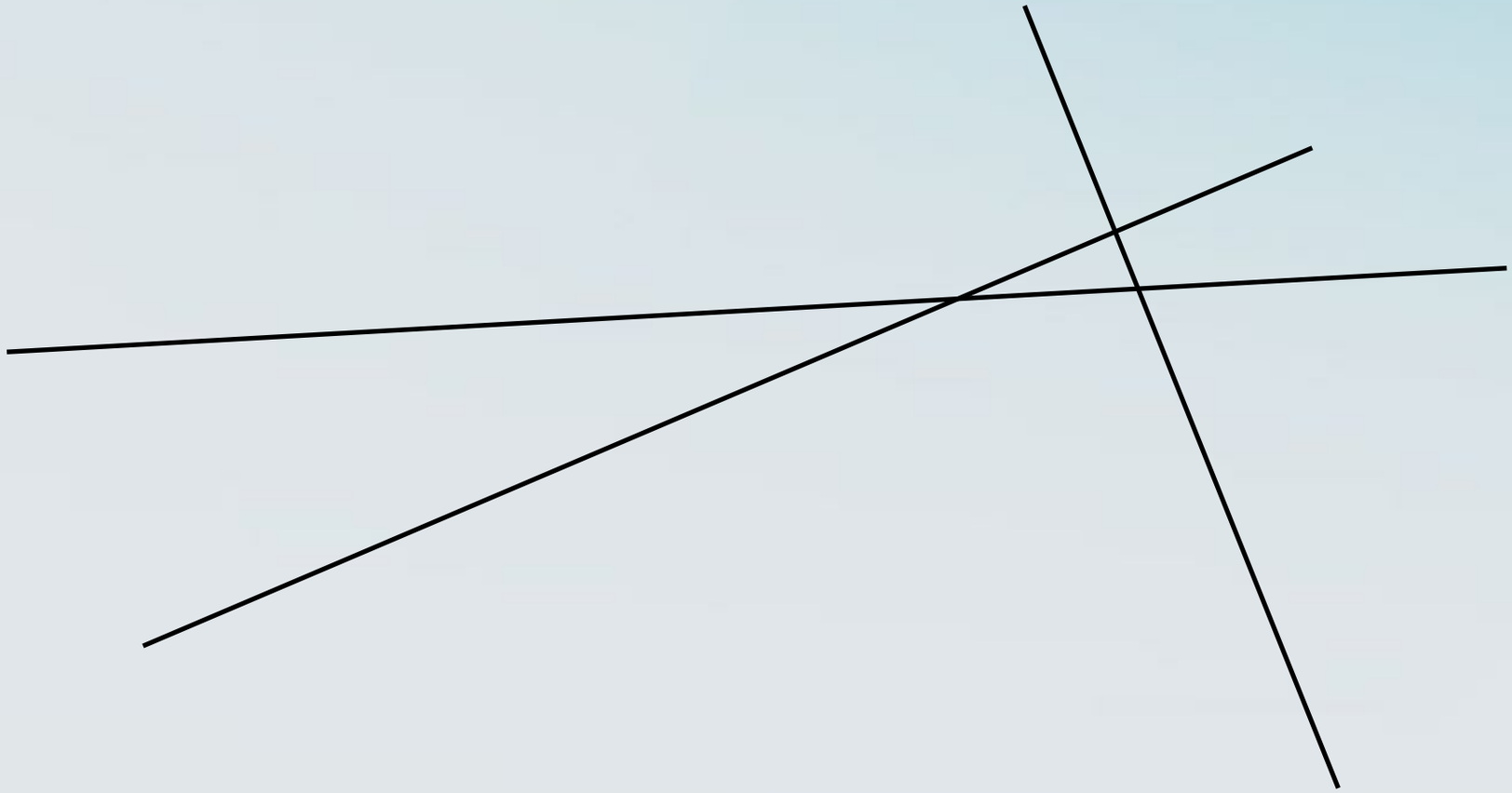
Feasible region



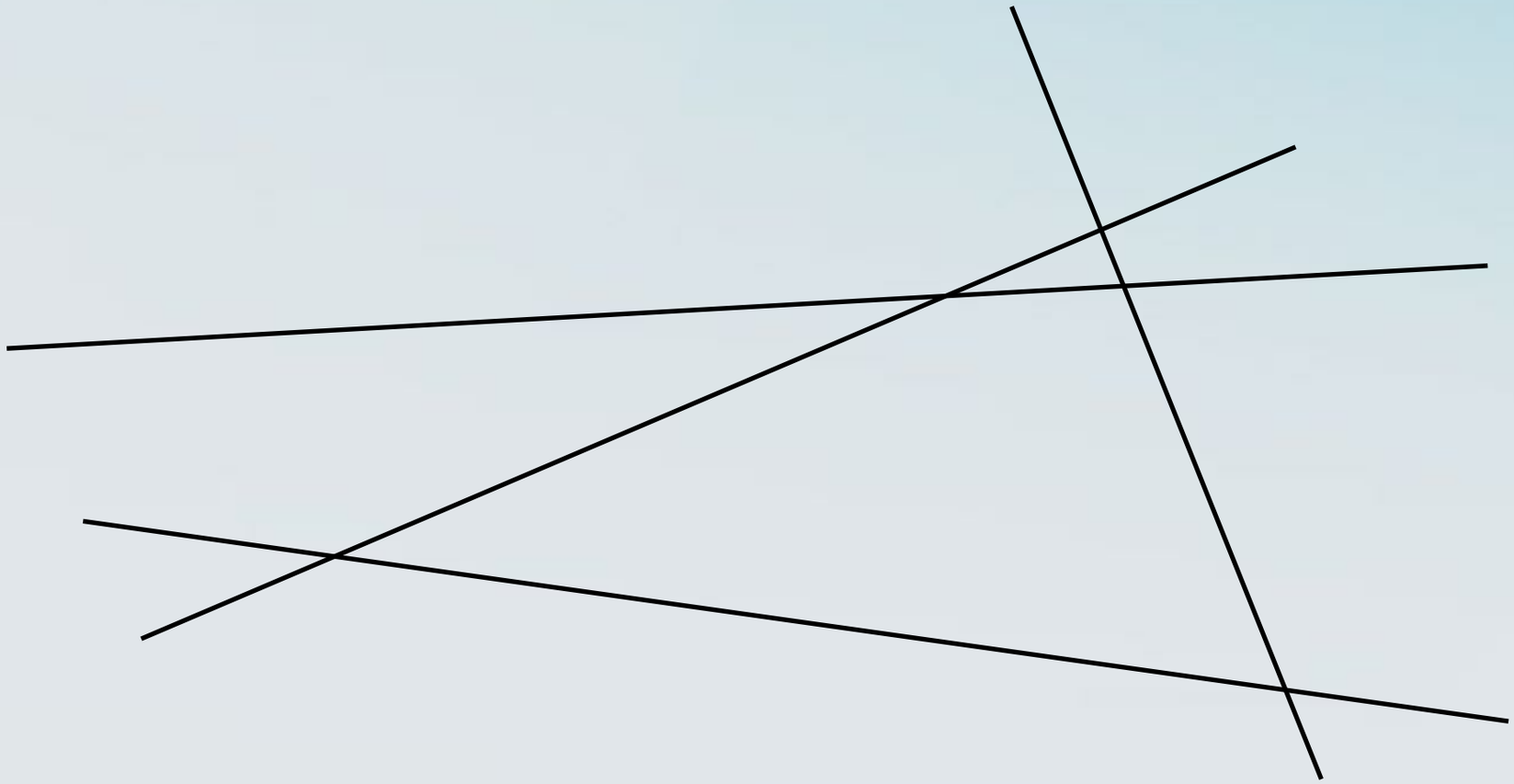
Feasible region



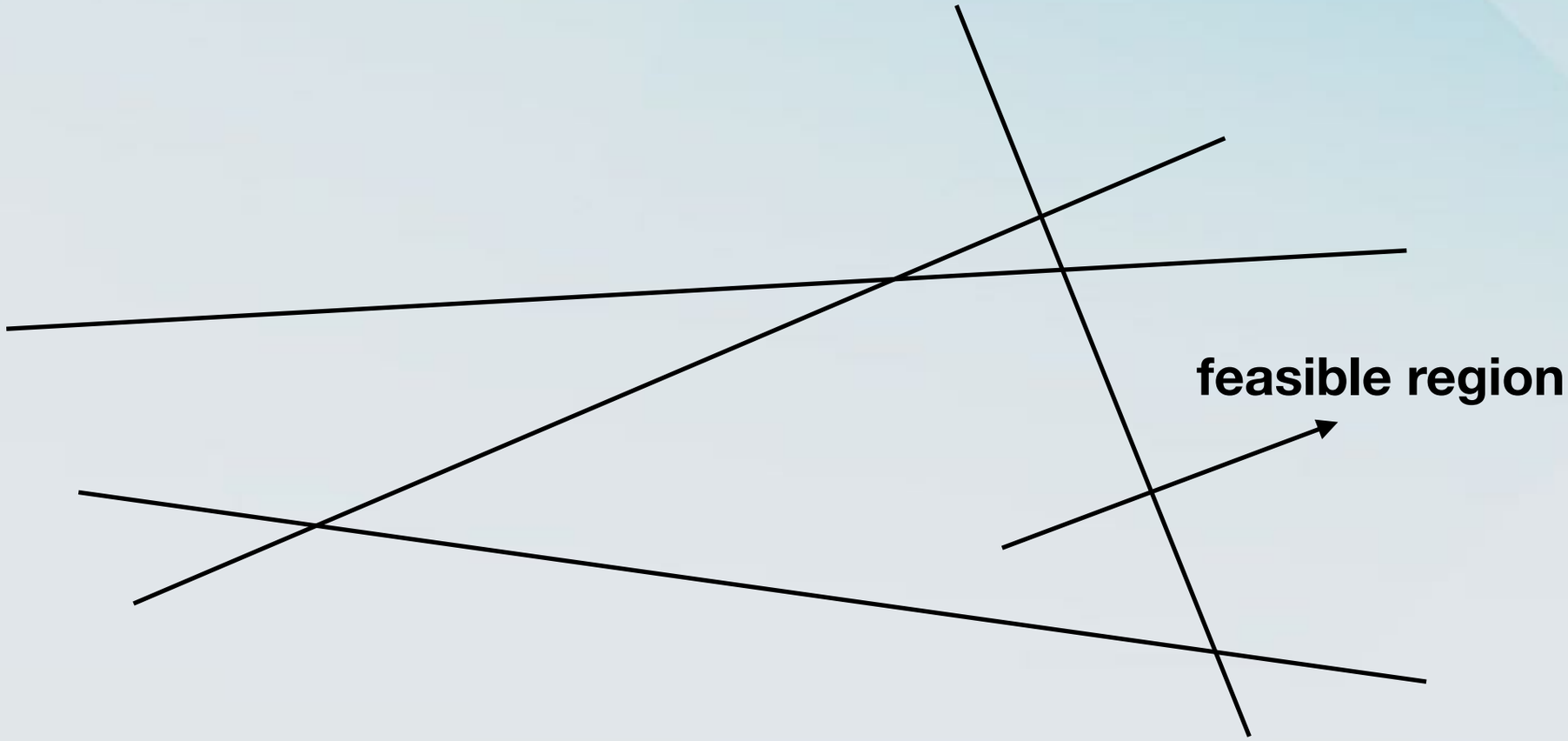
Feasible region



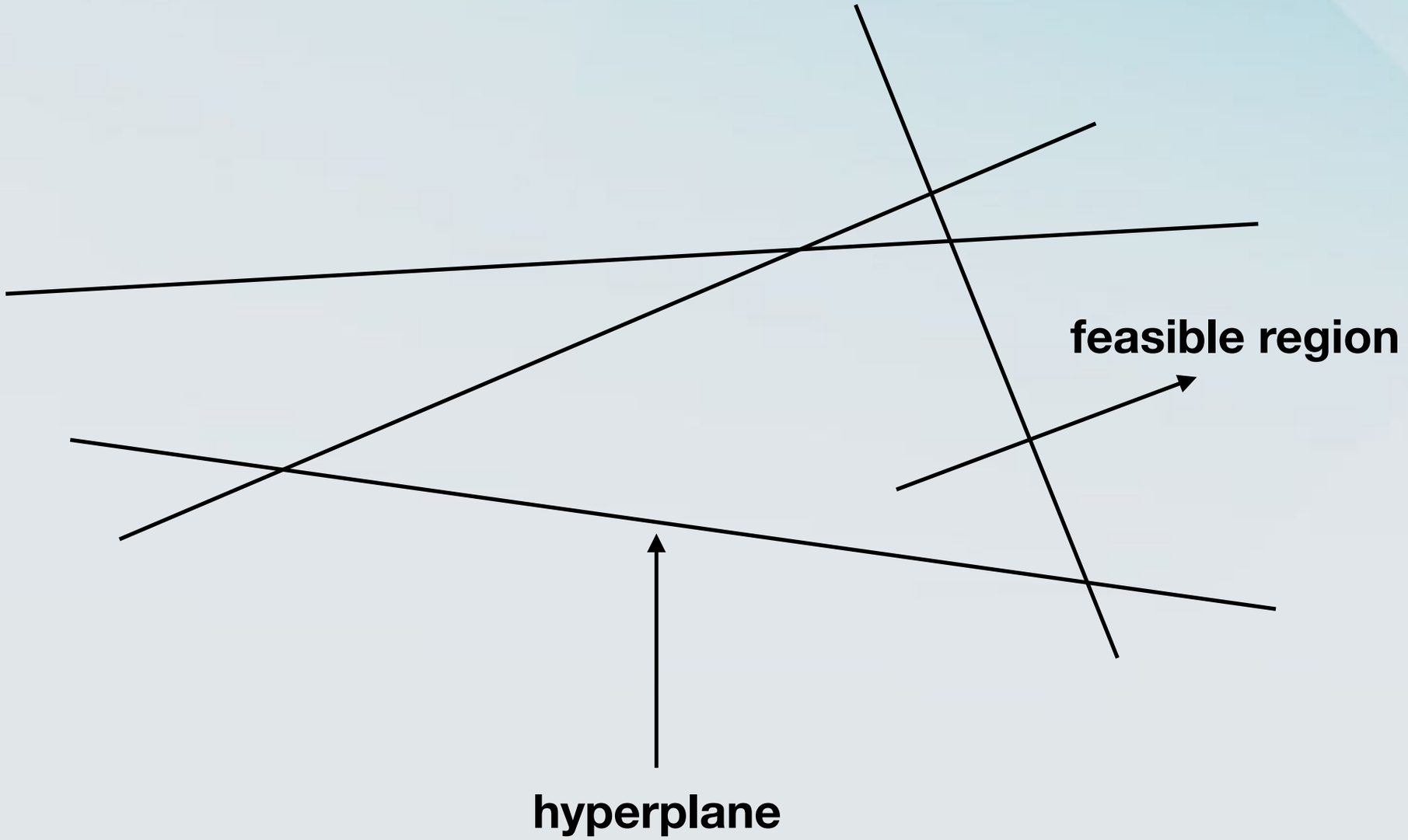
Feasible region



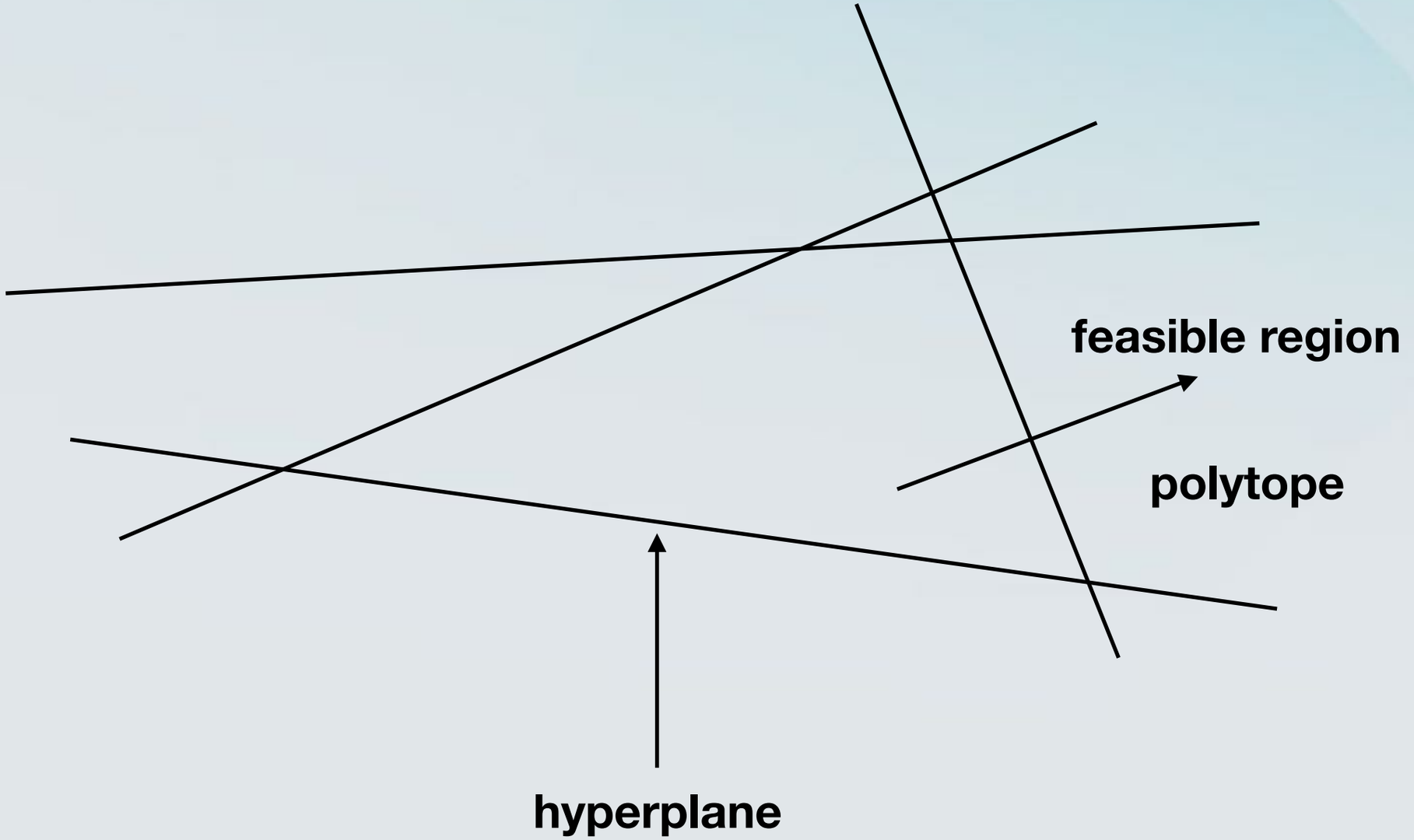
Feasible region



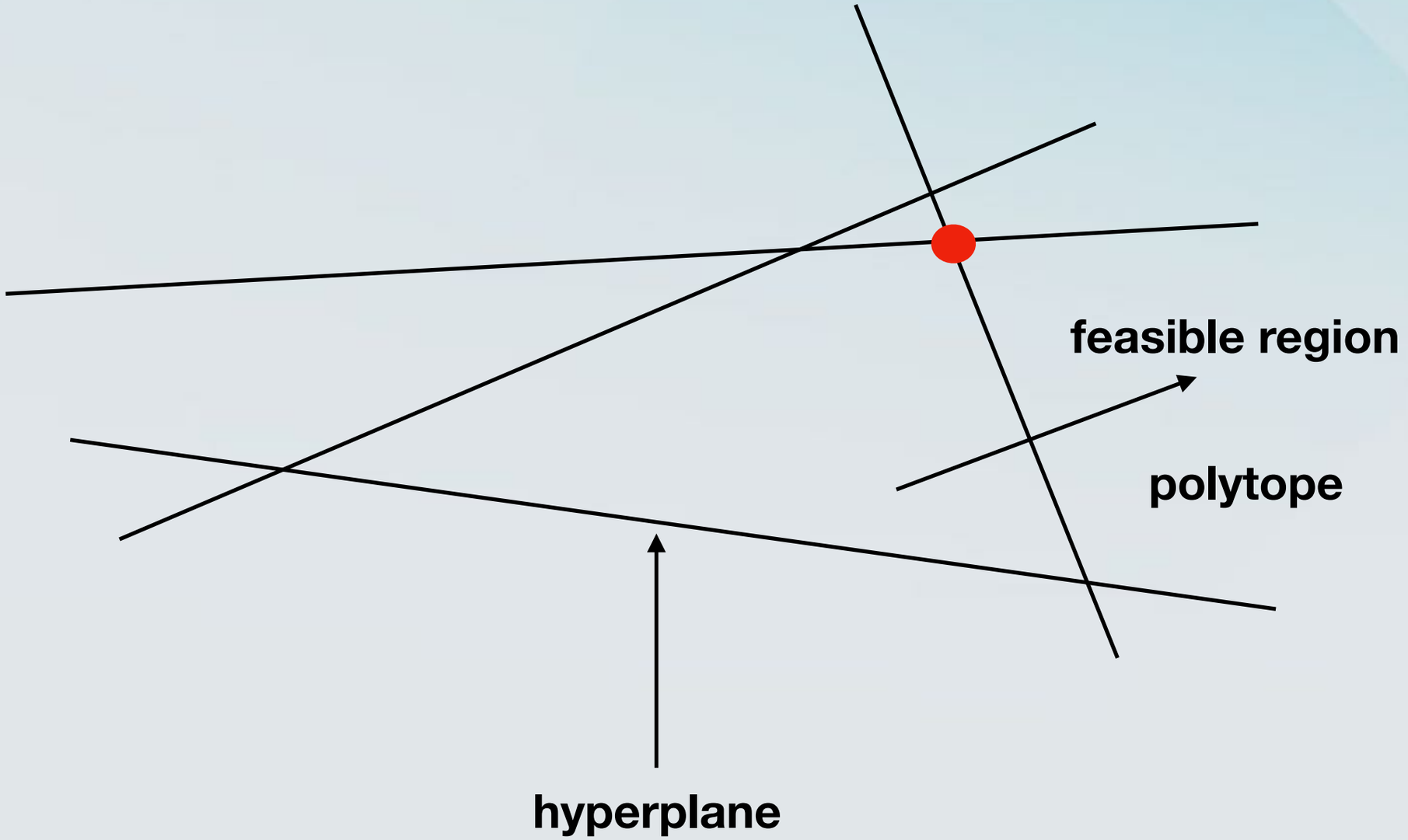
Feasible region



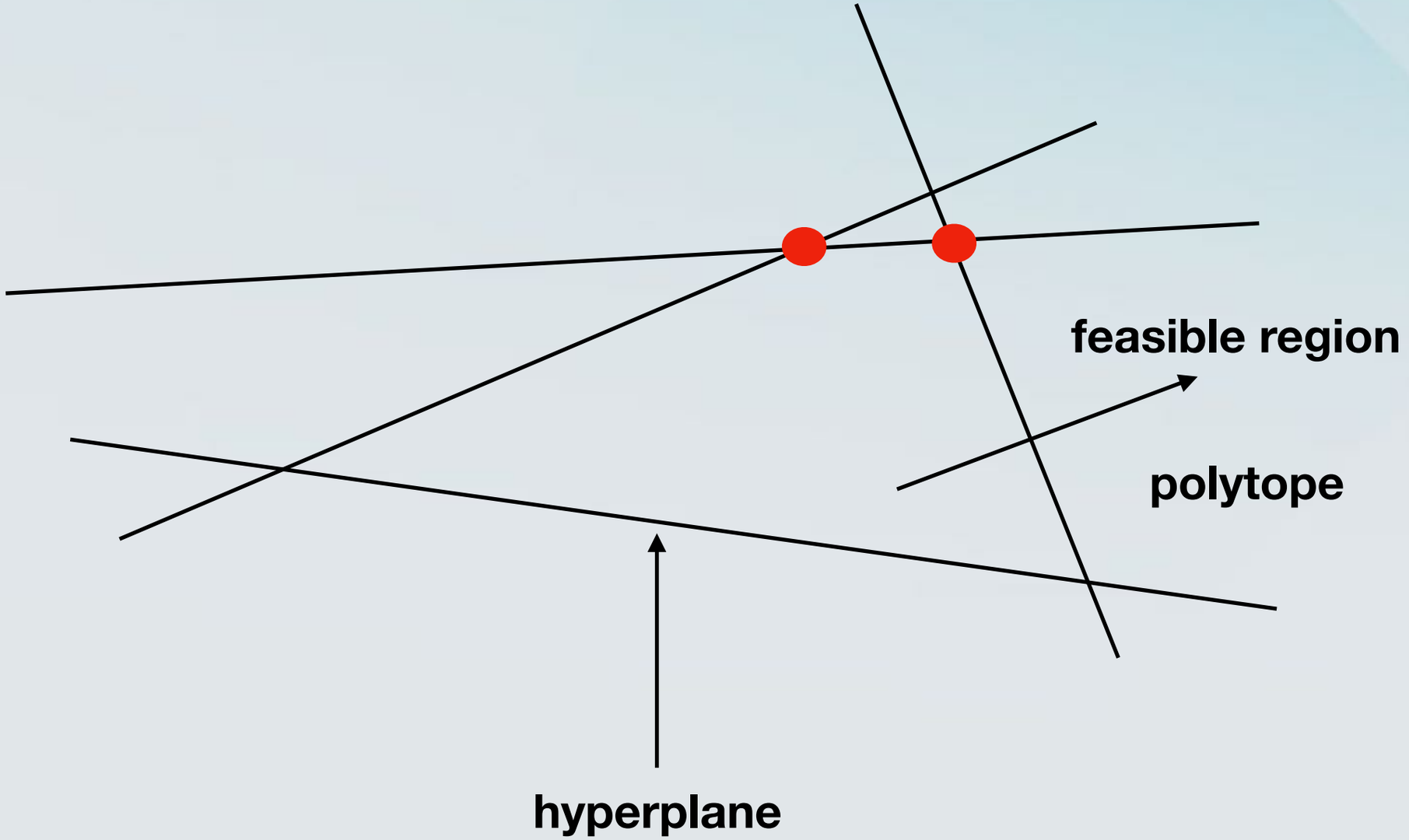
Feasible region



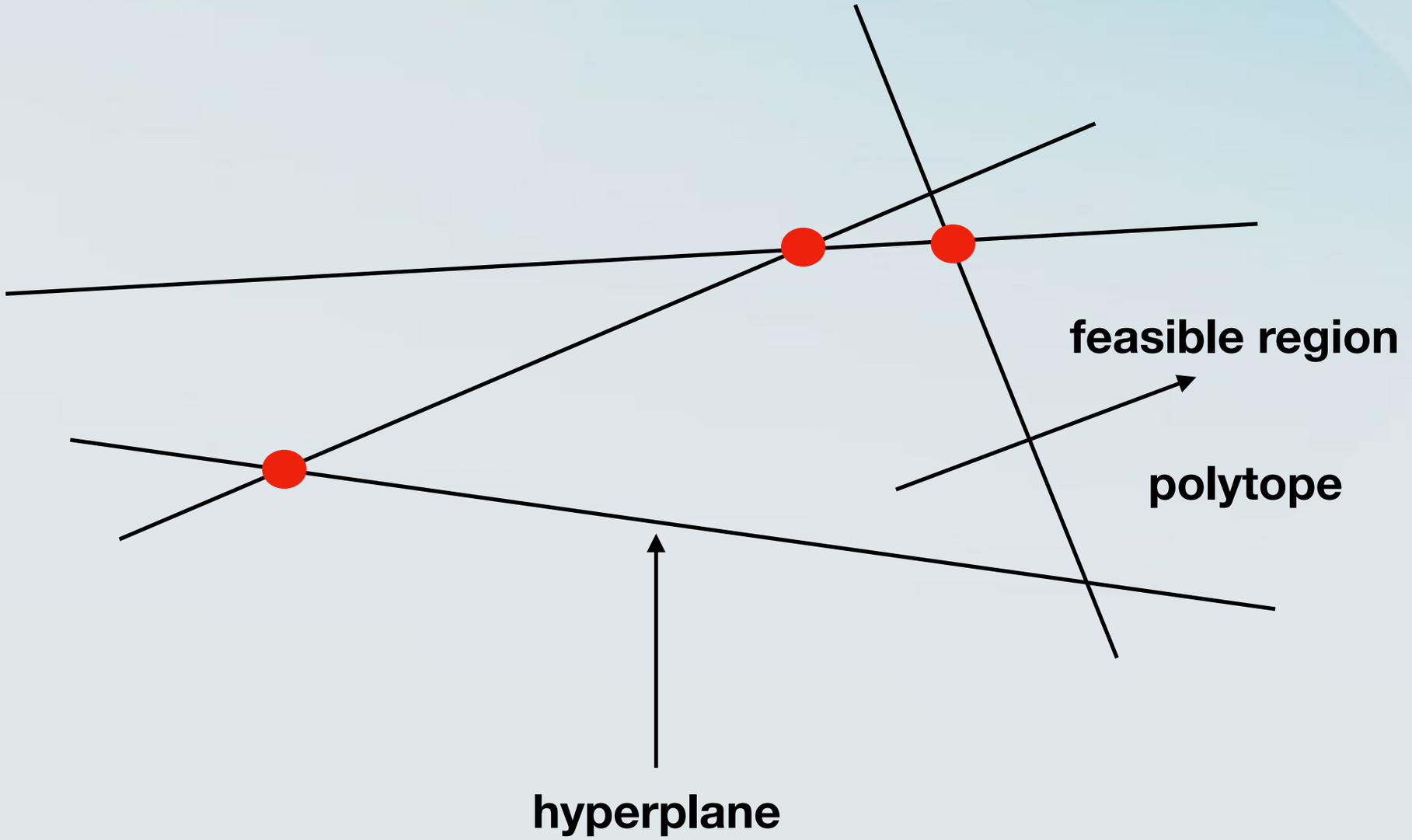
Feasible region



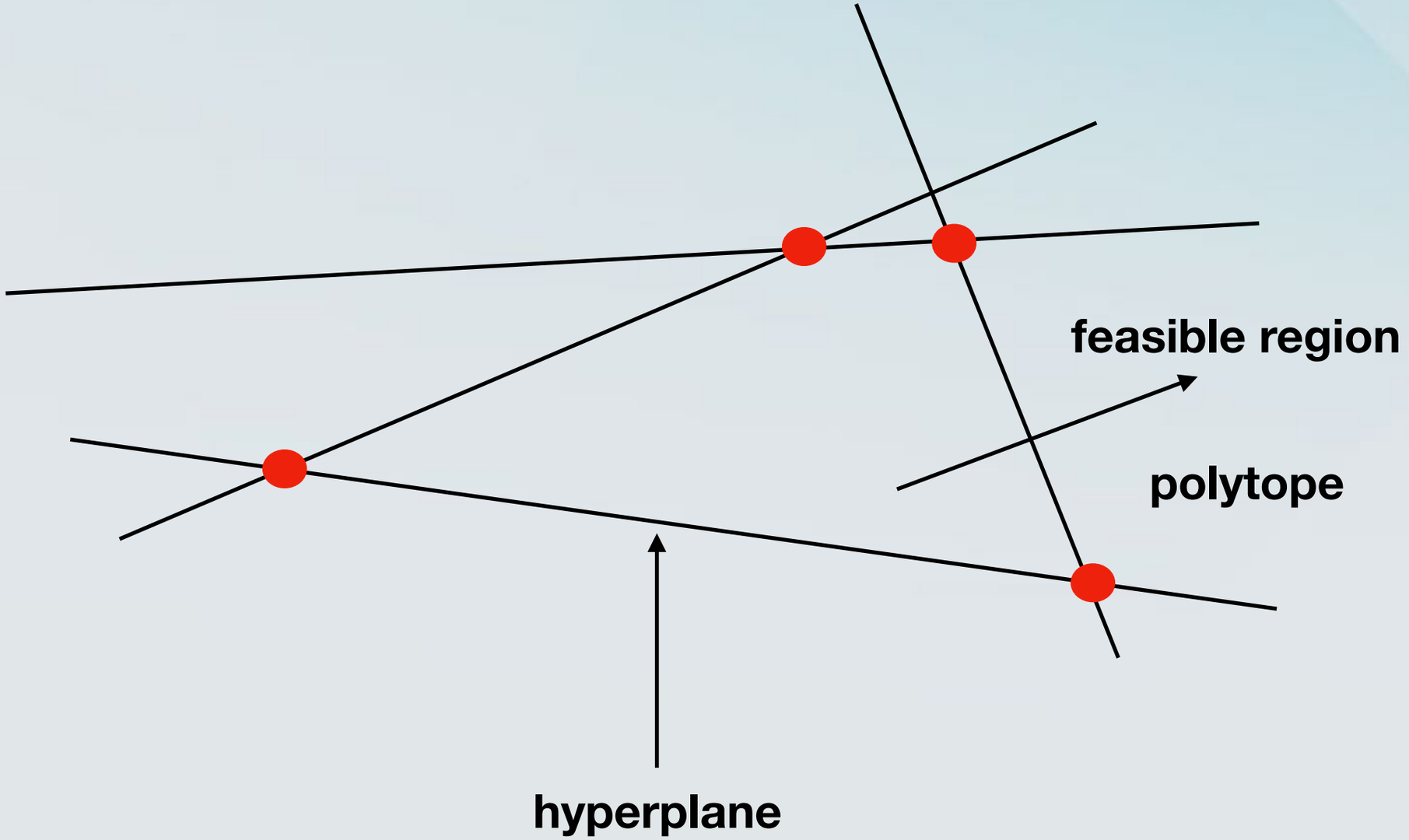
Feasible region



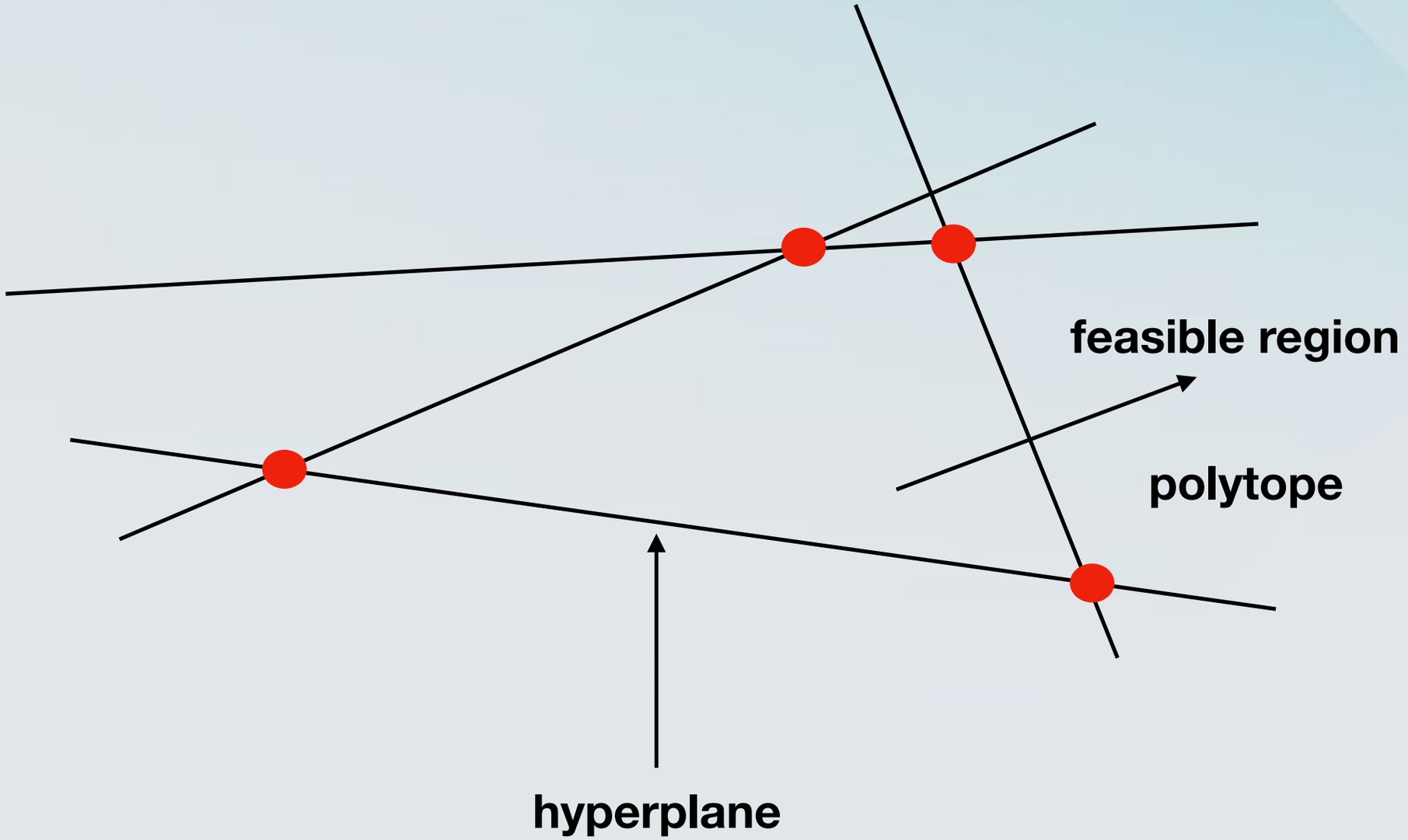
Feasible region



Feasible region



Feasible region



candidate optimal solution

Solving the linear program

- To find the optimal solution, it suffices to examine the *corners* of the feasible region.
- These are the intersection points of the lines defined by the constraints.

Solving the linear program

- To find the optimal solution, it suffices to examine the *corners* of the feasible region.
- These are the intersection points of the lines defined by the constraints.
- This is what the Simplex method does, via *pivoting*.

Solving the linear program

- To find the optimal solution, it suffices to examine the *corners* of the feasible region.
- These are the intersection points of the lines defined by the constraints.
- This is what the Simplex method does, via *pivoting*.
- Other algorithms for solving LPs:
  Ellipsoid Method, Interior Point Methods

Linear programming…

Linear programming…

- Is a large topic in itself.

Linear programming…

- Is a large topic in itself.
- Is covered in more detail in the COMP557 module.

Linear programming…

- Is a large topic in itself.
- Is covered in more detail in the COMP557 module.
- What we will need here:

Linear programming…

- Is a large topic in itself.
- Is covered in more detail in the COMP557 module.
- What we will need here:
  - If we manage to formulate a problem as a linear program, then we can solve it in polynomial time.

Linear programming

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

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

$$x_j \geq 0, \quad j = 1, ..., n$$

Integer Linear programming

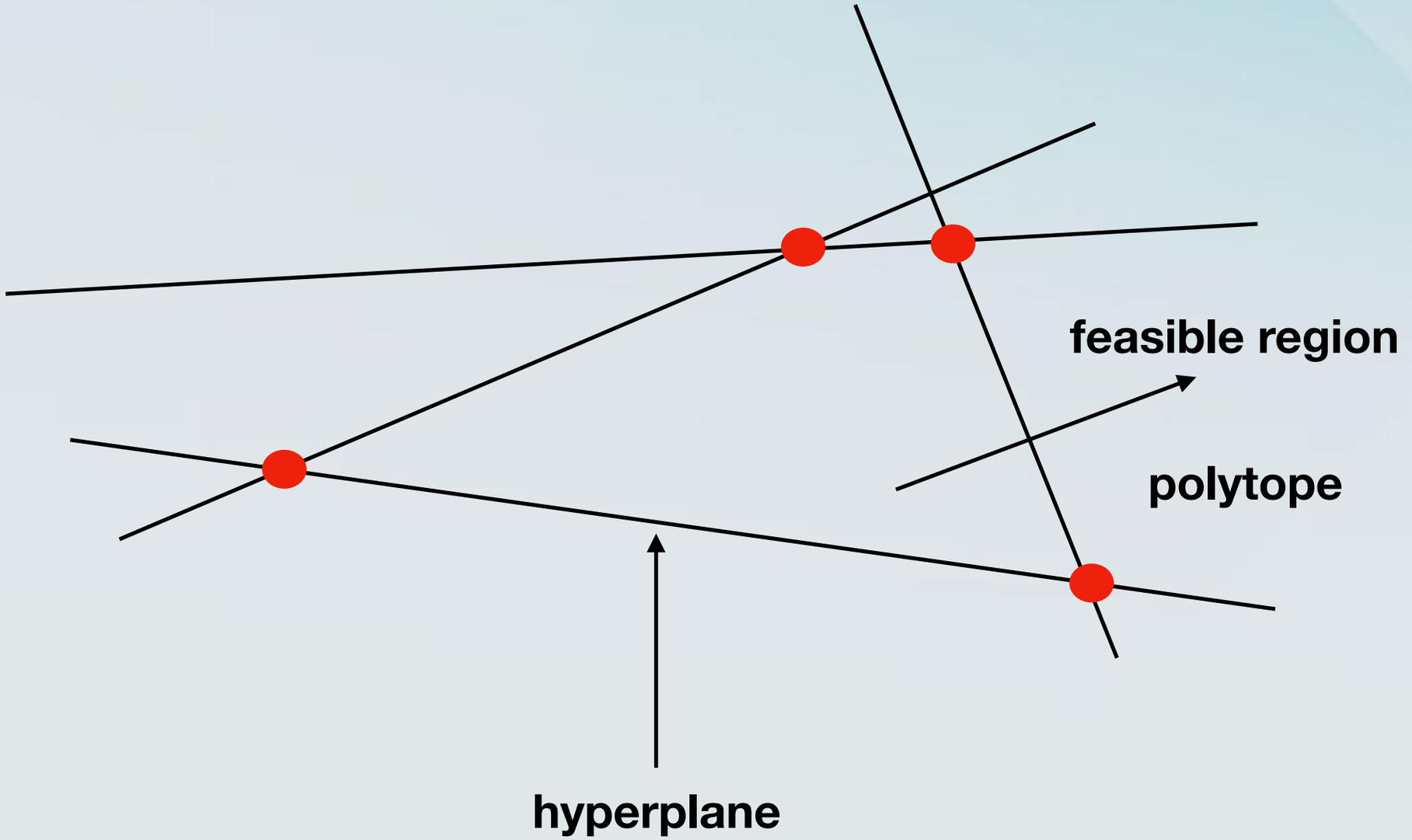
$$\text{maximise} \qquad \sum_{j=1}^{n} c_j x_j$$

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

$$x_j \geq 0, \quad j = 1, ..., n$$

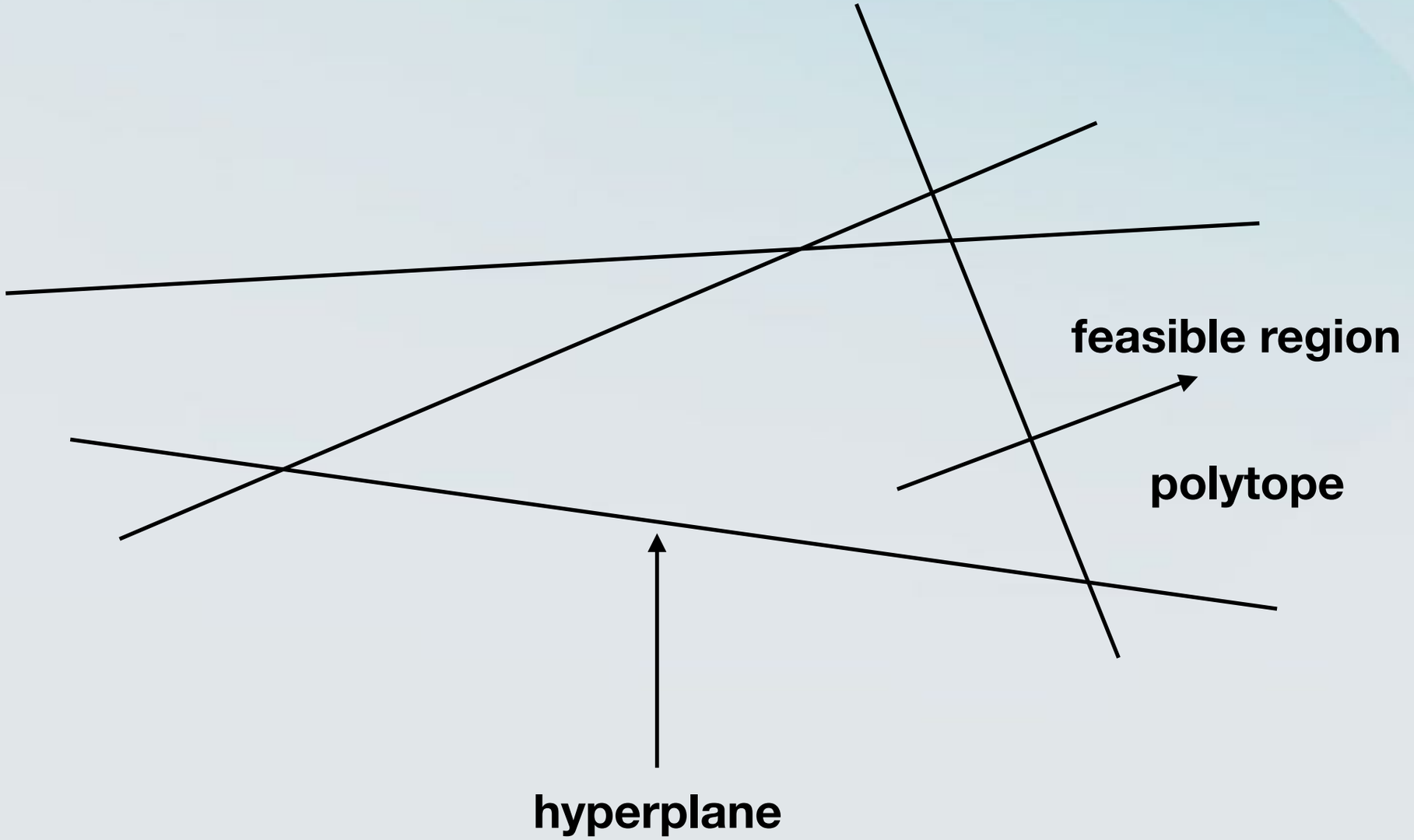
$$x_j \text{ is integer}$$

Feasible region

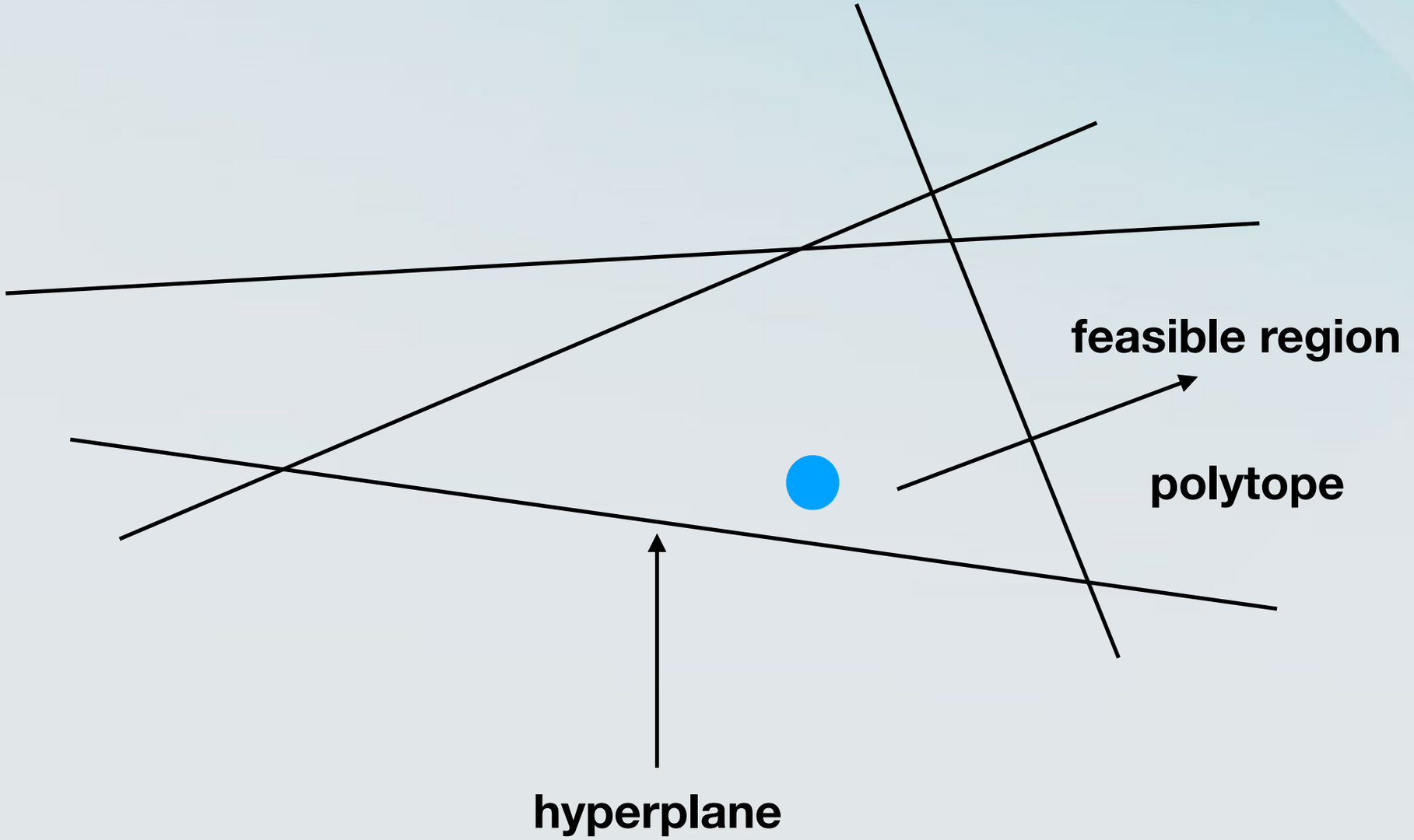


candidate optimal solution

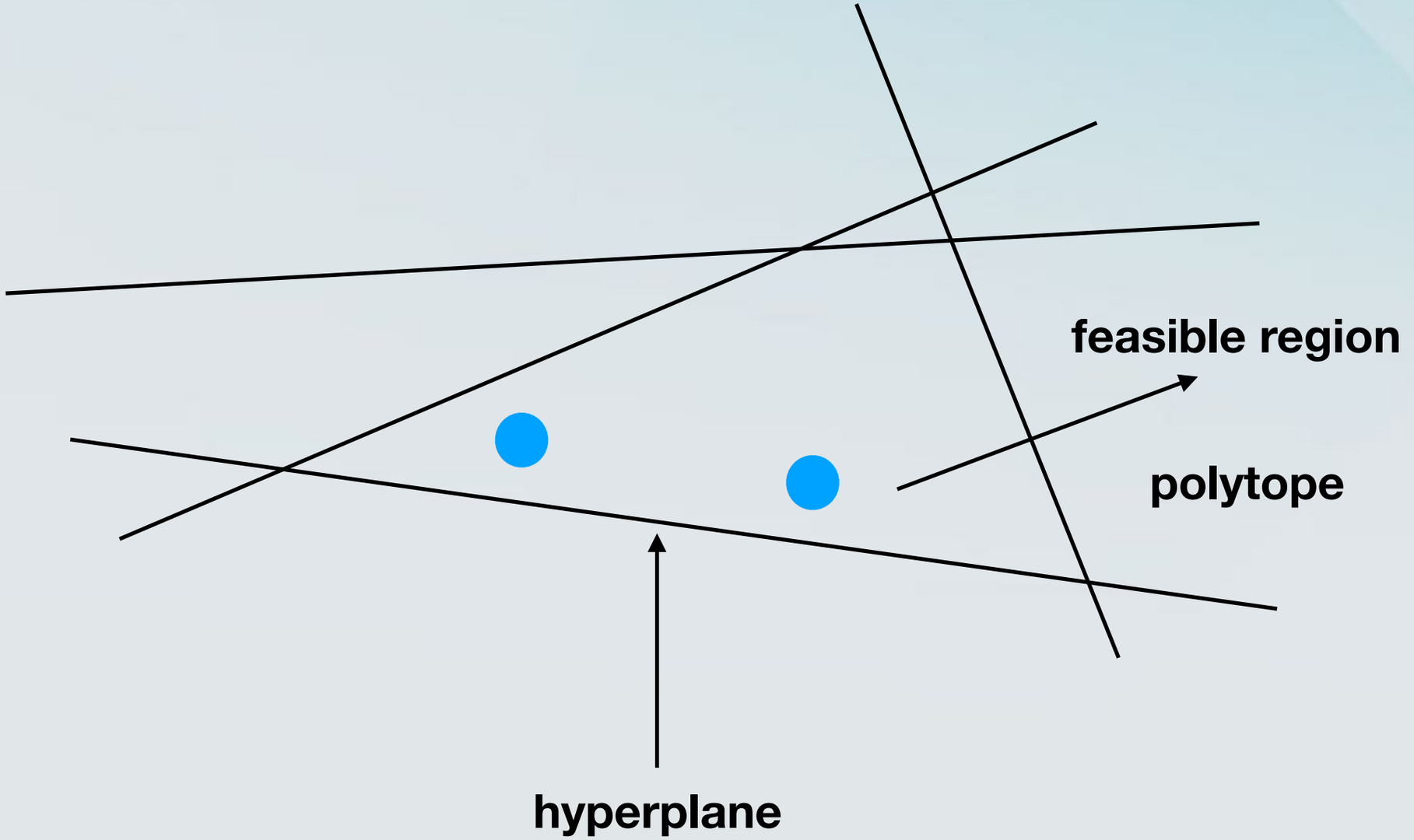
Feasible region



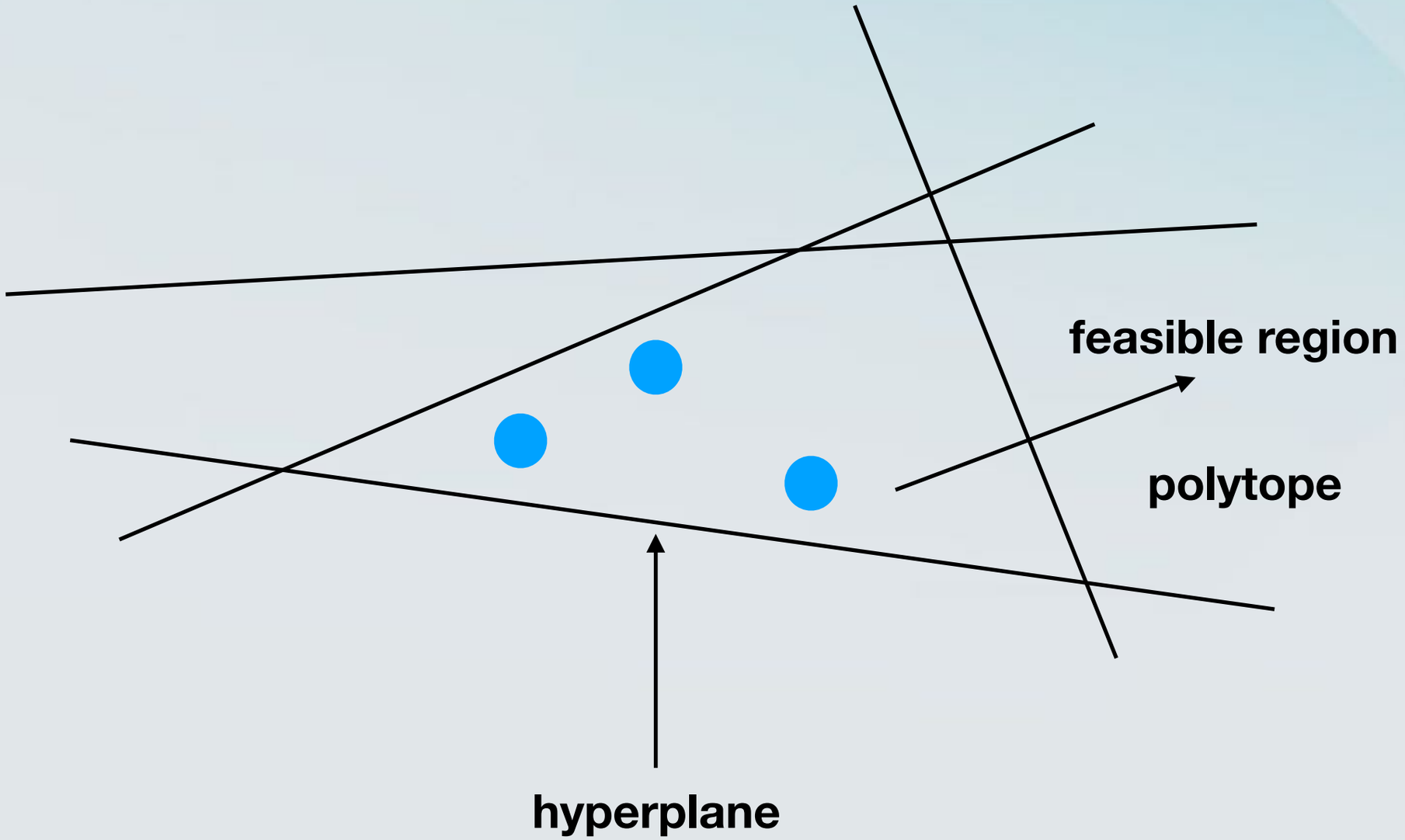
Feasible region



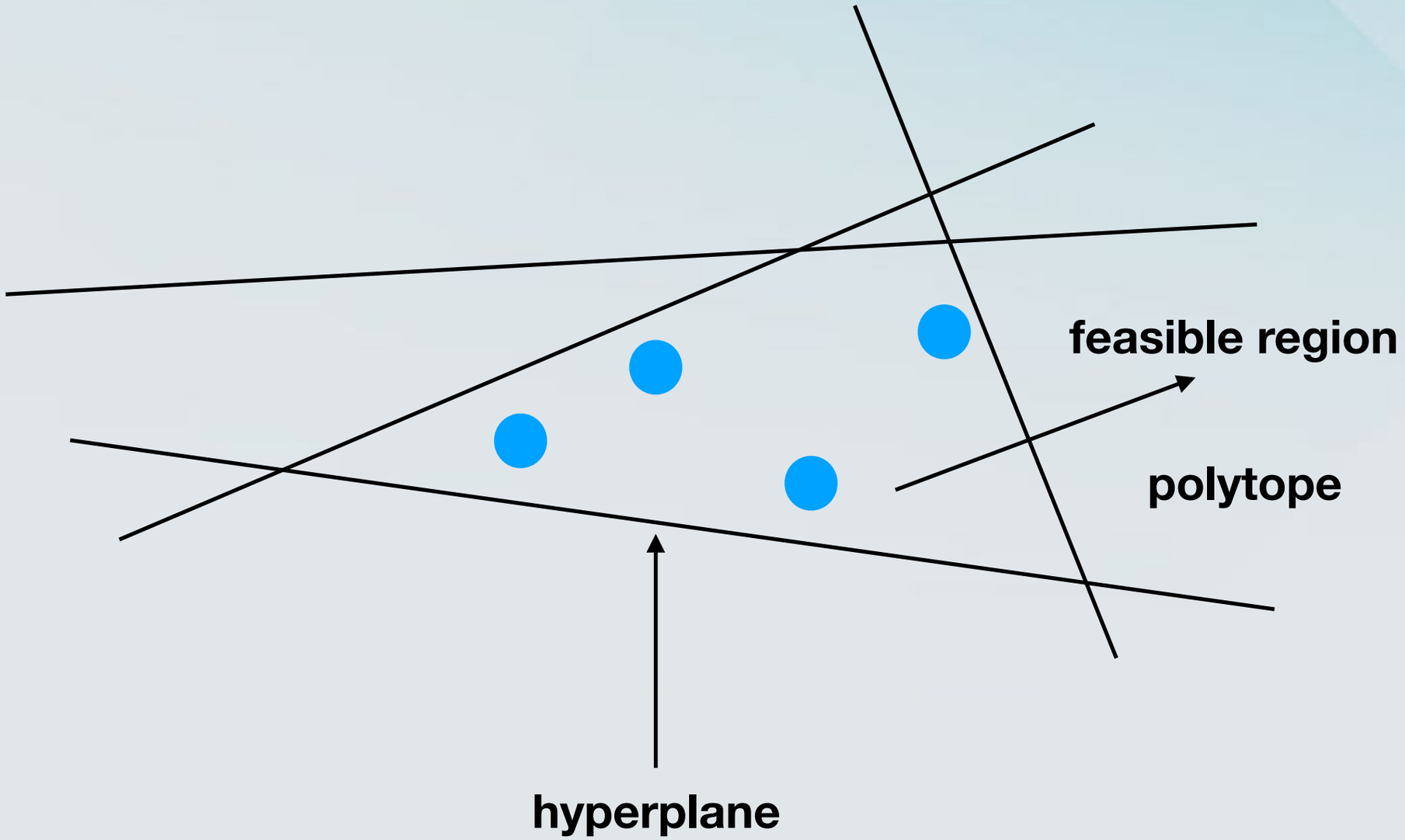
Feasible region



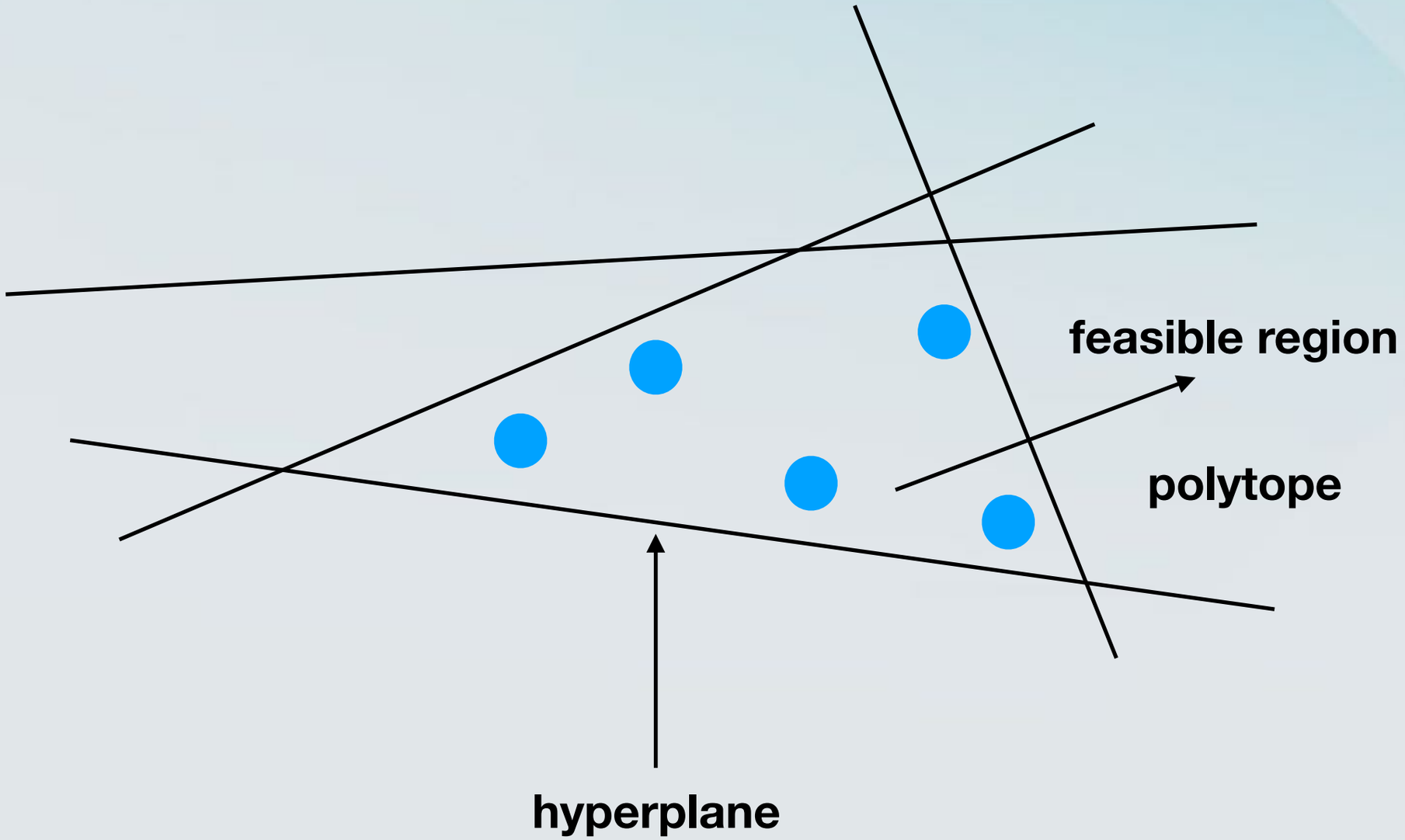
Feasible region



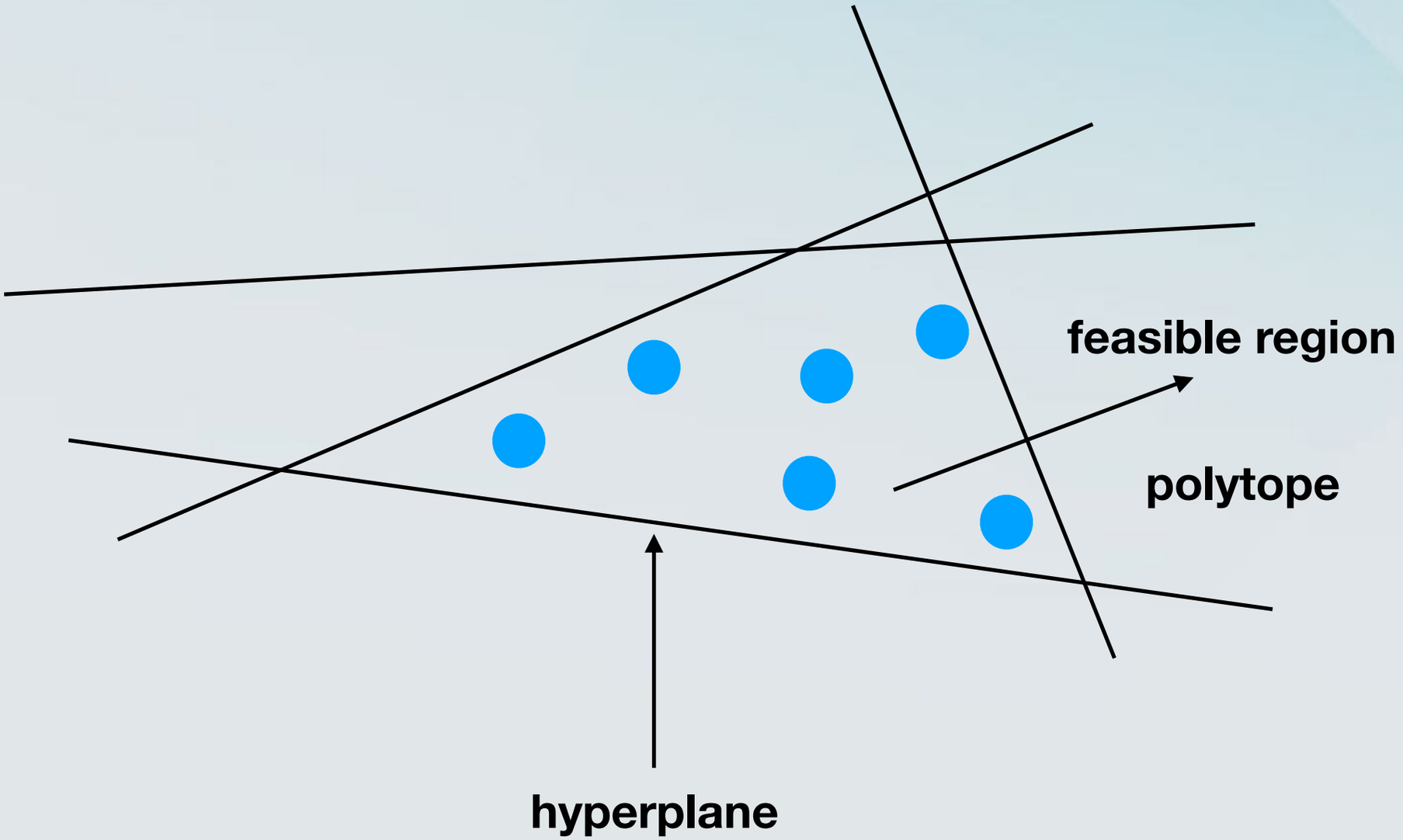
Feasible region



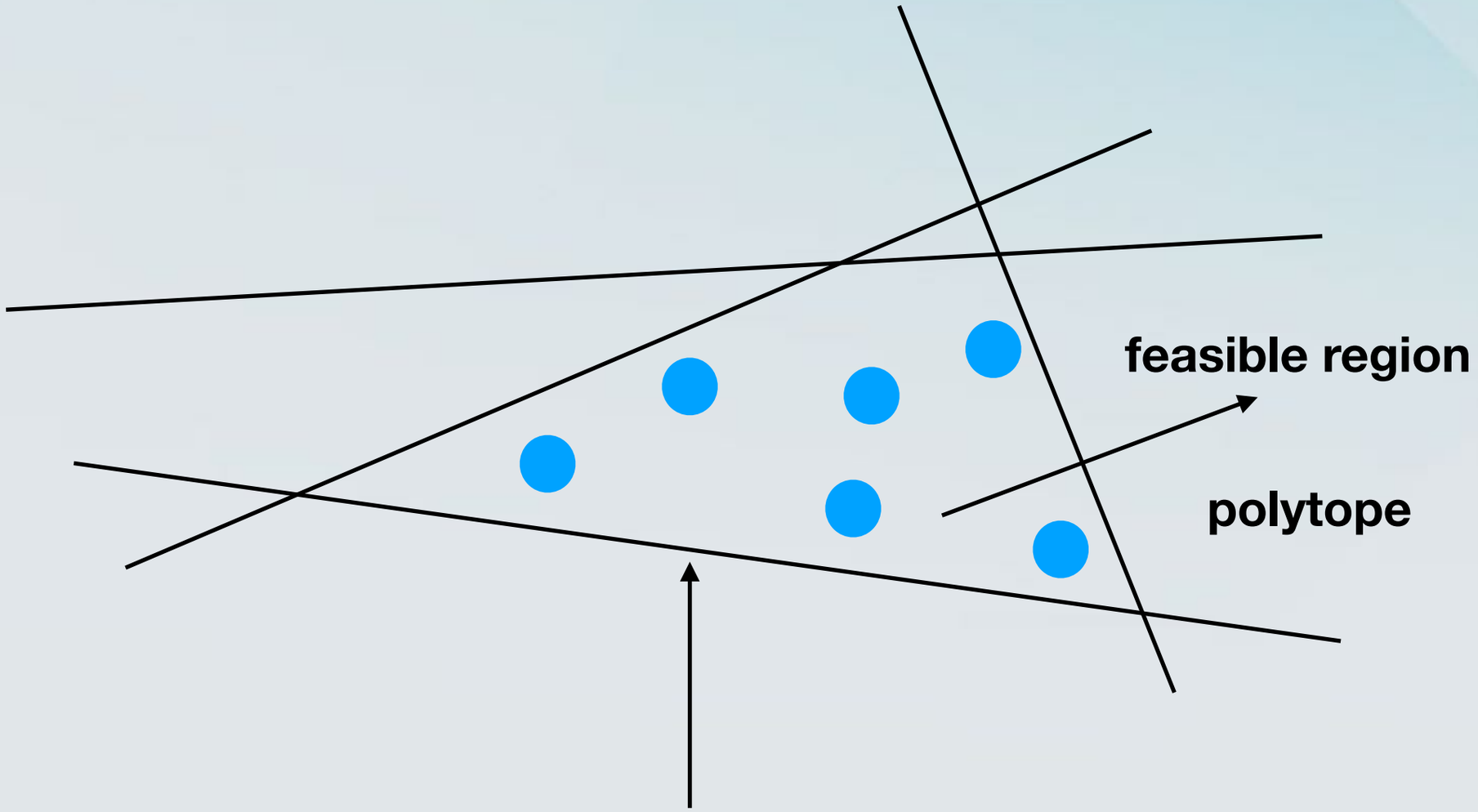
Feasible region



Feasible region



Feasible region



candidate optimal solution

Solving ILPs

Solving ILPs

- The corners are not necessarily integer solutions.

Solving ILPs

- The corners are not necessarily integer solutions.
  - It does not suffice to look at the corners.

Solving ILPs

- The corners are not necessarily integer solutions.
    - It does not suffice to look at the corners.
    - We can exhaustively try all possible integer solutions.

Solving ILPs

- The corners are not necessarily integer solutions.
  - It does not suffice to look at the corners.
  - We can exhaustively try all possible integer solutions.
  - Can we do something more clever?

Solving ILPs

- The corners are not necessarily integer solutions.
    - It does not suffice to look at the corners.
- We can exhaustively try all possible integer solutions.
- Can we do something more clever?
    - Yes, but in the worst-case, it will still take exponential time in many ILPs.

Solving ILPs

- The corners are not necessarily integer solutions.
    - It does not suffice to look at the corners.
    - We can exhaustively try all possible integer solutions.
    - Can we do something more clever?
        - Yes, but in the worst-case, it will still take exponential time in many ILPs.
    - Generally speaking, ILP solving is NP-hard.

Summarising

- Linear Programs can be solved in polynomial time.

- Integer Linear Programs generally cannot be solved in polynomial time (unless P=NP).

Duality

- Suppose that we have a linear program, which we will refer to as *the primal*.

Duality

- Suppose that we have a linear program, which we will refer to as *the primal*.

- We will construct another linear program, which we will refer to as *the dual*.

Duality

- Suppose that we have a linear program, which we will refer to as *the primal*.

- We will construct another linear program, which we will refer to as *the dual*.

- The *variables* of the primal become the *constraints* of the dual and vice-versa.

Duality

- Suppose that we have a linear program, which we will refer to as *the primal*.

- We will construct another linear program, which we will refer to as *the dual*.

- The *variables* of the primal become the *constraints* of the dual and vice-versa.

- Maximisation becomes minimisation.

Duality

- Suppose that we have a linear program, which we will refer to as *the primal*.

- We will construct another linear program, which we will refer to as *the dual*.

- The *variables* of the primal become the *constraints* of the dual and vice-versa.

- Maximisation becomes minimisation.

- The two linear programs will have a very important connection.

Duality

The Primal

$$\begin{aligned} \text{maximise} \quad & \sum_{j=1}^{n} c_j x_j \\ \text{subject to} \quad & \sum_{j=1\in}^{n} \alpha_{ij} x_j \leq b_i, \quad i = 1, ..., m \\ & x_j \geq 0, \quad j = 1, ..., n \end{aligned}$$

The Dual

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} b_i y_i \\ \text{subject to} \quad & \sum_{i=1}^{m} \alpha_{ij} y_i \geq c_j, \quad i = 1, ..., n \\ & y_j \geq 0, \qquad j = 1, ..., m \end{aligned}$$

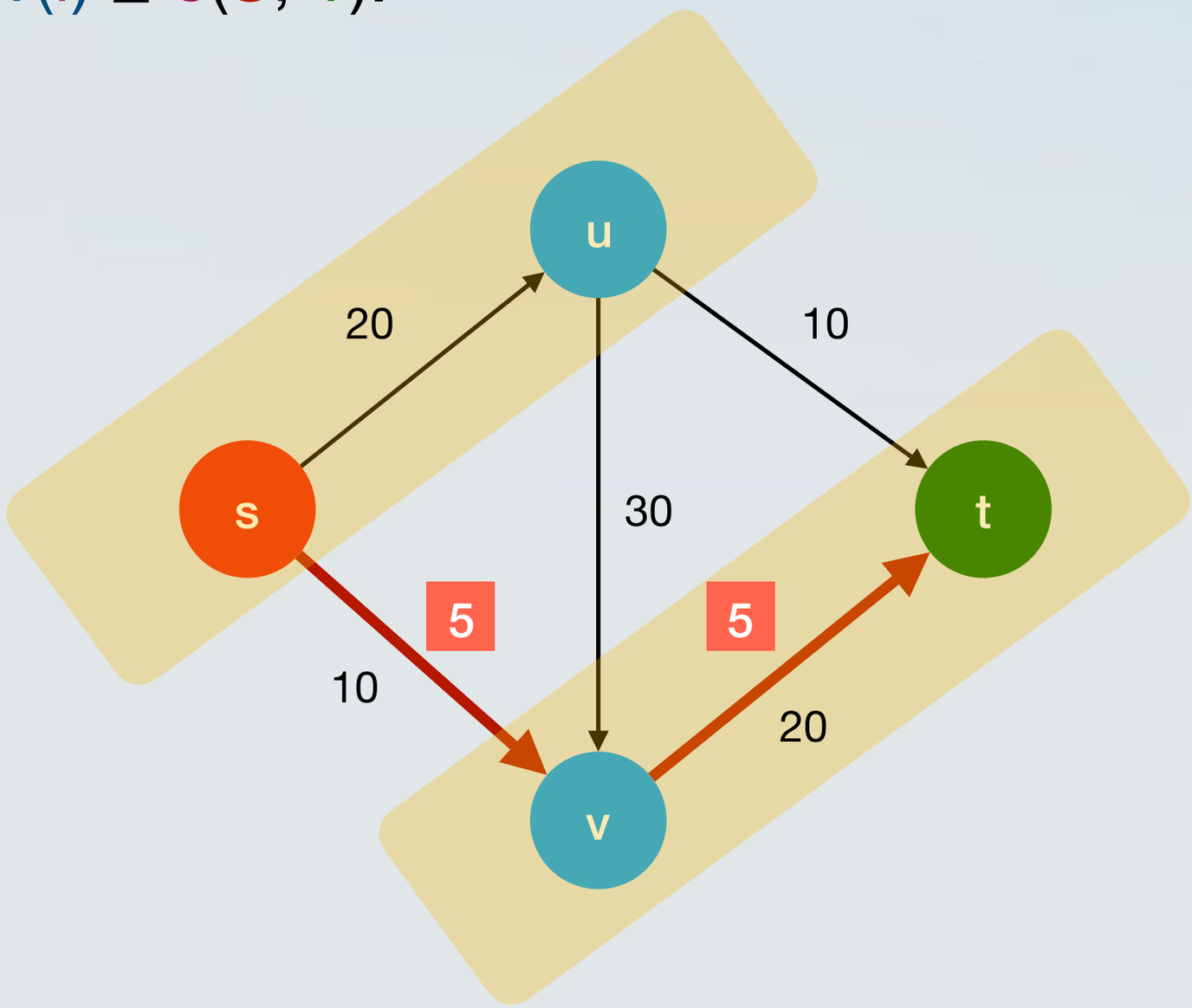
Weak Duality

- Let x be any feasible solution to the Primal and let y be any feasible solution to the Dual. Then we have that

$$\text{value}(x) \leq \text{value}(y)$$

Weak Duality

- Fact 3: Let f by any (s-t) flow and (S, T) be any (s-t) cut. Then v(f) ≤ c(S, T).



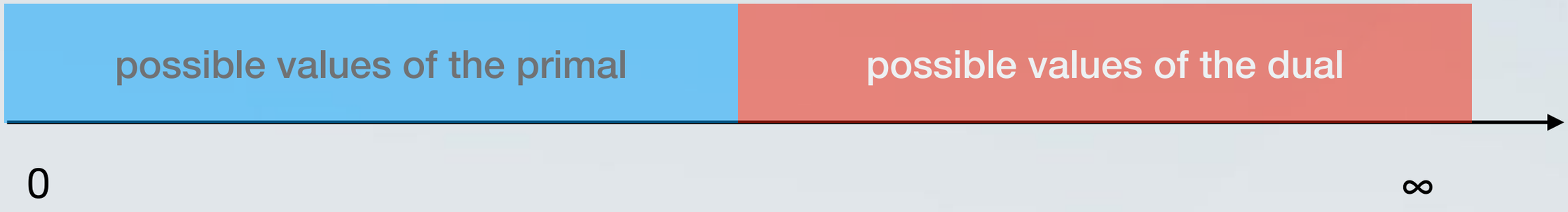
Strong Duality

- Let x be any feasible solution to the Primal and let y be any feasible solution to the Dual. If

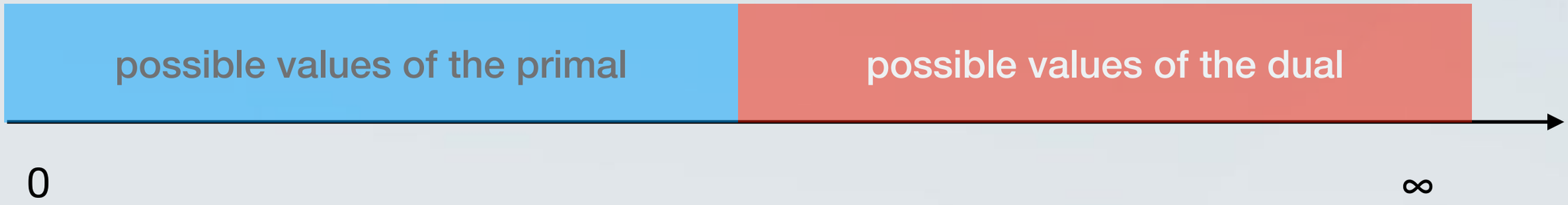
$$\text{value}(x) = \text{value}(y)$$

then x and y are both optimal solutions.

Pictorially

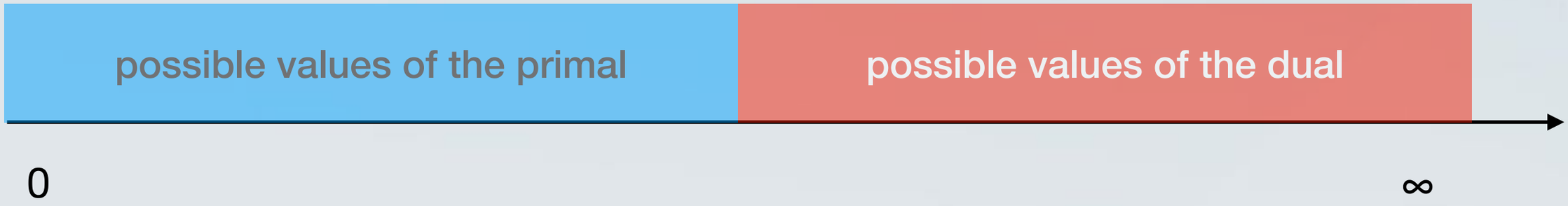


Pictorially



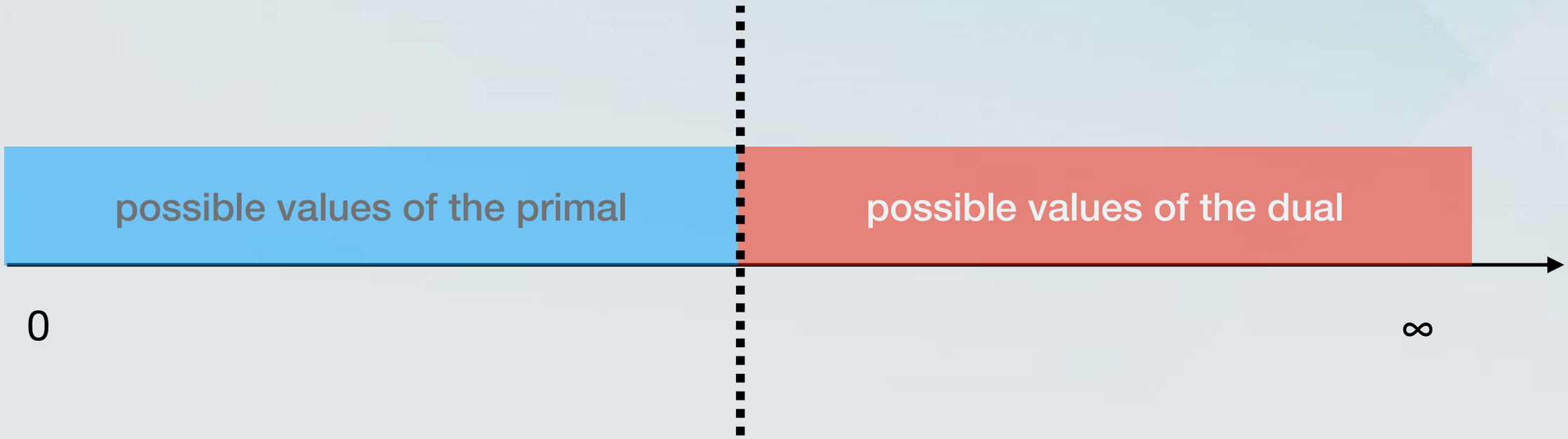
- How can we prove that a solution x to the primal is maximum?

Pictorially



- How can we prove that a solution x to the primal is maximum?
- Find a solution y to the dual with value(y) = value(x)

Pictorially



- How can we prove that a solution x to the primal is maximum?

- Find a solution y to the dual with value(y) = value(x)

The Max-Flow Min-Cut Theorem

- Theorem: In every flow network, the value of the maximum flow is *equal* to the capacity of the minimum cut.

- This is a consequence of the *strong duality theorem* for linear programs!

As a matter of fact…

As a matter of fact…

- The Simplex method starts from some feasible solution (possibly the point (0, …. , 0)).

As a matter of fact…

- The Simplex method starts from some feasible solution (possibly the point (*0*, …. , *0*)).
- It improves the solution in every step (via *pivoting*) until it can no longer be improved.

As a matter of fact…

- The Simplex method starts from some feasible solution (possibly the point $(0, \ldots, 0)$).

- It improves the solution in every step (via *pivoting*) until it can no longer be improved.

- To prove that the solution is optimal, we use duality.

As a matter of fact…

- The Simplex method starts from some feasible solution (possibly the point $(0, \ldots, 0)$).
- It improves the solution in every step (via *pivoting*) until it can no longer be improved.
- To prove that the solution is optimal, we use duality.
- Do you know of any other algorithms for which the same principle applies?

Maximum Flow as an LP

- We can write the maximum flow problem as a linear problem.
- "Maximise the flow, subject to capacity and flow conservation constraints".

$$
\begin{aligned}
\text{maximise} \quad & \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} \\
\text{subject to} \quad & f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V \\
& \sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\} \\
& f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V
\end{aligned}
$$

Maximum Flow as an LP

- We can write the maximum flow problem as a linear problem.
- "Maximise the flow, subject to capacity and flow conservation constraints".

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Maximum Flow as an LP

- We can write the maximum flow problem as a linear problem.
- "Maximise the flow, subject to capacity and flow conservation constraints".

total flow out of s

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Maximum Flow as an LP

- We can write the maximum flow problem as a linear problem.
- "Maximise the flow, subject to capacity and flow conservation constraints".

total flow out of s

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Maximum Flow as an LP

- We can write the maximum flow problem as a linear problem.
- "Maximise the flow, subject to capacity and flow conservation constraints".

total flow out of s — total flow into s

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Maximum Flow as an LP

- We can write the maximum flow problem as a linear problem.
- "Maximise the flow, subject to capacity and flow conservation constraints".

total flow out of s — total flow into s

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Maximum Flow as an LP

- We can write the maximum flow problem as a linear problem.
- "Maximise the flow, subject to capacity and flow conservation constraints".

total flow out of s — total flow into s

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

capacity constraint

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Maximum Flow as an LP

- We can write the maximum flow problem as a linear problem.
- "Maximise the flow, subject to capacity and flow conservation constraints".

total flow out of s — total flow into s

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

capacity constraint

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Maximum Flow as an LP

- We can write the maximum flow problem as a linear problem.
- "Maximise the flow, subject to capacity and flow conservation constraints".

total flow out of s — total flow into s

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

capacity constraint

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

flow conservation

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Maximum Flow as an LP

- We can write the maximum flow problem as a linear problem.
- "Maximise the flow, subject to capacity and flow conservation constraints".

total flow out of s — total flow into s

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

capacity constraint

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

flow conservation

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Maximum Flow as an LP

- We can write the maximum flow problem as a linear problem.
- "Maximise the flow, subject to capacity and flow conservation constraints".

total flow out of s — total flow into s

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

capacity constraint

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

flow conservation

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

non-negative flow

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Constructing the dual

$$\begin{aligned}
\text{minimise} \quad & \sum_{(u,v)\in E} c_{uv} d_{uv} \\
\text{subject to} \quad & d_{uv} - z_u + z_v \geq 0 \quad \text{for each } (u,v) \in E, u \neq s, v \neq t \\
& d_{su} + z_v \geq 1 \quad \text{for each } (s,u) \in E \\
& d_{ut} - z_u \geq 0 \quad \text{for each } (u,t) \in E \\
& d_{uv} \geq 0, \quad \text{for each } (u,v) \in E \\
& z_u \geq 0 \quad \text{for each } u \in V - \{t,s\}
\end{aligned}$$

Constructing the dual

This is *1* if u is in S and v is in T and *0* otherwise.

$$\text{minimise} \quad \sum_{(u,v)\in E} c_{uv} d_{uv}$$

$$\begin{aligned}
\text{subject to} \quad d_{uv} - z_u + z_v &\geq 0 \quad \text{for each } (u,v) \in E, u \neq s, v \neq t \\
d_{su} + z_v &\geq 1 \quad \text{for each } (s,u) \in E \\
d_{ut} - z_u &\geq 0 \quad \text{for each } (u,t) \in E \\
d_{uv} &\geq 0, \quad \text{for each } (u,v) \in E \\
z_u &\geq 0 \quad \text{for each } u \in V - \{t,s\}
\end{aligned}$$

Constructing the dual

This is *1* if u is in S and v is in T and *0* otherwise.

$$\text{minimise} \quad \sum_{(u,v)\in E} c_{uv} d_{uv}$$

$$\begin{aligned}
\text{subject to} \quad d_{uv} - z_u + z_v &\geq 0 \quad \text{for each } (u,v) \in E, u \neq s, v \neq t \\
d_{su} + z_v &\geq 1 \quad \text{for each } (s,u) \in E \\
d_{ut} - z_u &\geq 0 \quad \text{for each } (u,t) \in E \\
d_{uv} &\geq 0, \quad \text{for each } (u,v) \in E \\
z_u &\geq 0 \quad \text{for each } u \in V - \{t,s\} \\
d_{uv} &\in \{0,1\}, \quad \text{for each } (u,v) \in E
\end{aligned}$$

Constructing the dual

This is *1* if u is in S and v is in T and *0* otherwise.

This is *1* if u is in S and *0* otherwise.

$$\text{minimise} \quad \sum_{(u,v)\in E} c_{uv} d_{uv}$$

$$\begin{aligned}
\text{subject to} \quad d_{uv} - z_u + z_v &\geq 0 \quad \text{for each } (u,v) \in E, u \neq s, v \neq t \\
d_{su} + z_v &\geq 1 \quad \text{for each } (s,u) \in E \\
d_{ut} - z_u &\geq 0 \quad \text{for each } (u,t) \in E \\
d_{uv} &\geq 0, \quad \text{for each } (u,v) \in E \\
z_u &\geq 0 \quad \text{for each } u \in V - \{t,s\} \\
d_{uv} &\in \{0,1\}, \quad \text{for each } (u,v) \in E
\end{aligned}$$

Constructing the dual

This is *1* if u is in S and v is in T and *0* otherwise.

This is *1* if u is in S and *0* otherwise.

$$\text{minimise} \quad \sum_{(u,v)\in E} c_{uv} d_{uv}$$

$$\begin{aligned}
\text{subject to} \quad d_{uv} - z_u + z_v &\geq 0 \quad \text{for each } (u,v) \in E, u \neq s, v \neq t \\
d_{su} + z_v &\geq 1 \quad \text{for each } (s,u) \in E \\
d_{ut} - z_u &\geq 0 \quad \text{for each } (u,t) \in E \\
d_{uv} &\geq 0, \quad \text{for each } (u,v) \in E \\
z_u &\geq 0 \quad \text{for each } u \in V - \{t,s\} \\
d_{uv} &\in \{0,1\}, \quad \text{for each } (u,v) \in E \\
z_u &\in \{0,1\}, \quad \text{for each } u \in V - \{s,t\}
\end{aligned}$$

Constructing the dual

This is *1* if u is in S and v is in T and *0* otherwise.

This is *1* if u is in S and *0* otherwise.

$$\text{minimise} \quad \sum_{(u,v)\in E} c_{uv} d_{uv}$$

$$\text{subject to} \quad d_{uv} - z_u + z_v \geq 0 \quad \text{for each } (u,v) \in E, u \neq s, v \neq t$$

$$d_{su} + z_v \geq 1 \quad \text{for each } (s,u) \in E$$

$$d_{ut} - z_u \geq 0 \quad \text{for each } (u,t) \in E$$

$$d_{uv} \geq 0, \quad \text{for each } (u,v) \in E$$

$$z_u \geq 0 \quad \text{for each } u \in V - \{t,s\}$$

$$d_{uv} \in \{0,1\}, \quad \text{for each } (u,v) \in E$$

$$z_u \in \{0,1\}, \quad \text{for each } u \in V - \{s,t\}$$

If u is in S and v is in T, then d_{uv} must be *1*.

Constructing the dual

This is *1* if u is in S and v is in T and *0* otherwise.

This is *1* if u is in S and *0* otherwise.

$$\text{minimise} \quad \sum_{(u,v)\in E} c_{uv} d_{uv}$$

$$\begin{aligned}
\text{subject to} \quad & d_{uv} - z_u + z_v \geq 0 \quad \text{for each } (u,v) \in E, u \neq s, v \neq t \\
& d_{su} + z_v \geq 1 \quad \text{for each } (s,u) \in E \\
& d_{ut} - z_u \geq 0 \quad \text{for each } (u,t) \in E \\
& d_{uv} \geq 0, \quad \text{for each } (u,v) \in E \\
& z_u \geq 0 \quad \text{for each } u \in V - \{t,s\}
\end{aligned}$$

If u is in S and v is in T, then d_{uv} must be *1*.

If v is in T then d_{sv} must be *1*.

$$d_{uv} \in \{0,1\}, \quad \text{for each } (u,v) \in E$$
$$z_u \in \{0,1\}, \quad \text{for each } u \in V - \{s,t\}$$

Constructing the dual

This is *1* if u is in S and v is in T and *0* otherwise.

This is *1* if u is in S and *0* otherwise.

$$\text{minimise} \quad \sum_{(u,v)\in E} c_{uv} d_{uv}$$

$$\begin{aligned}
\text{subject to} \quad & d_{uv} - z_u + z_v \geq 0 \quad \text{for each } (u,v) \in E, u \neq s, v \neq t \\
& d_{su} + z_v \geq 1 \quad \text{for each } (s,u) \in E \\
& d_{ut} - z_u \geq 0 \quad \text{for each } (u,t) \in E \\
& d_{uv} \geq 0, \quad \text{for each } (u,v) \in E \\
& z_u \geq 0 \quad \text{for each } u \in V - \{t,s\}
\end{aligned}$$

If u is in S and v is in T, then d_{uv} must be *1*.

If v is in T then d_{sv} must be *1*.

If u is in S then d_{ut} must be *1*.

$$d_{uv} \in \{0,1\}, \quad \text{for each } (u,v) \in E$$

$$z_u \in \{0,1\}, \quad \text{for each } u \in V - \{s,t\}$$

Minimum Cut as an ILP

$$\text{minimise} \quad \sum_{(u,v)\in E} c_{uv} d_{uv}$$

$$\begin{aligned}
\text{subject to} \quad d_{uv} - z_u + z_v &\geq 0 \quad \text{for each } (u,v) \in E, u \neq s, v \neq t \\
d_{su} + z_v &\geq 1 \quad \text{for each } (s,u) \in E \\
d_{ut} - z_u &\geq 0 \quad \text{for each } (u,t) \in E \\
d_{uv} &\geq 0, \quad \text{for each } (u,v) \in E \\
z_u &\geq 0 \quad \text{for each } u \in V - \{t,s\} \\
d_{uv} &\in \{0,1\}, \quad \text{for each } (u,v) \in E \\
z_u &\in \{0,1\}, \quad \text{for each } u \in V - \{s,t\}
\end{aligned}$$

LP-relaxation

- An *LP-relaxation* of an Integer Linear Program is a linear program which is identical to the ILP, except all the integrality constraints have been removed (“*relaxed*”), or replaced with non-integral constraints.

Minimum Cut as an ILP

$$\text{minimise} \quad \sum_{(u,v)\in E} c_{uv} d_{uv}$$

$$\begin{aligned}
\text{subject to} \quad d_{uv} - z_u + z_v &\geq 0 \quad \text{for each } (u,v) \in E, u \neq s, v \neq t \\
d_{su} + z_v &\geq 1 \quad \text{for each } (s,u) \in E \\
d_{ut} - z_u &\geq 0 \quad \text{for each } (u,t) \in E \\
d_{uv} &\geq 0, \quad \text{for each } (u,v) \in E \\
z_u &\geq 0 \quad \text{for each } u \in V - \{t,s\} \\
d_{uv} &\in \{0,1\}, \quad \text{for each } (u,v) \in E \\
z_u &\in \{0,1\}, \quad \text{for each } u \in V - \{s,t\}
\end{aligned}$$

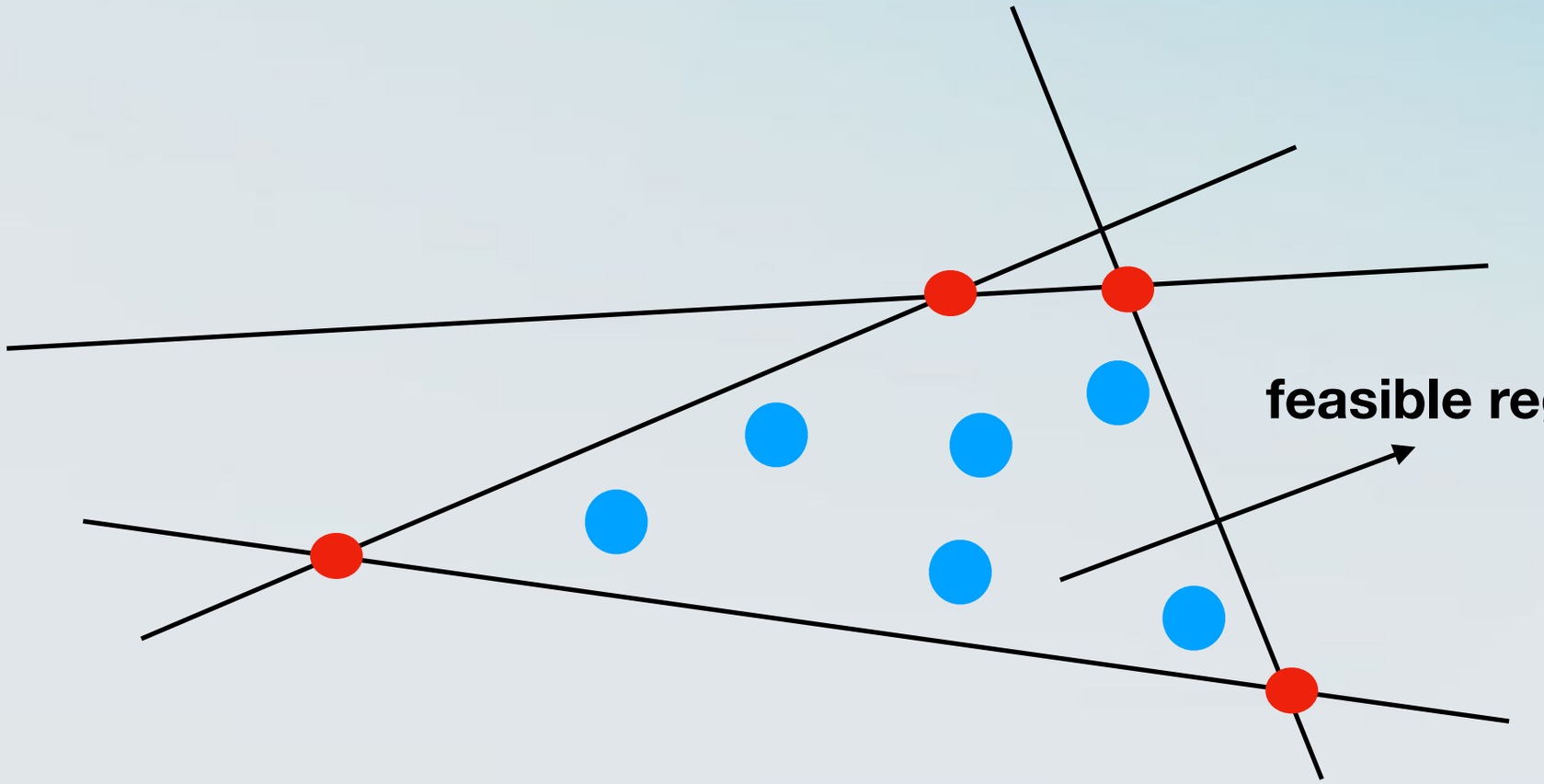
Minimum Cut as an ILP

LP-relaxation

$$\begin{aligned}
\text{minimise} \quad & \sum_{(u,v)\in E} c_{uv} d_{uv} \\
\text{subject to} \quad & d_{uv} - z_u + z_v \geq 0 \quad \text{for each } (u,v) \in E, u \neq s, v \neq t \\
& d_{su} + z_v \geq 1 \quad \text{for each } (s,u) \in E \\
& d_{ut} - z_u \geq 0 \quad \text{for each } (u,t) \in E \\
& d_{uv} \geq 0, \quad \text{for each } (u,v) \in E \\
& z_u \geq 0 \quad \text{for each } u \in V - \{t,s\}
\end{aligned}$$

$$d_{uv} \in \{0,1\}, \quad \text{for each } (u,v) \in E$$
$$z_u \in \{0,1\}, \quad \text{for each } u \in V - \{s,t\}$$

ILP vs LP-relaxation



candidate optimal solution for ILP

candidate optimal solution for LP-relaxation

ILP vs LP-relaxation

ILP vs LP-relaxation

- For a maximisation problem:

ILP vs LP-relaxation

- For a maximisation problem:
- The optimal value of the ILP is not larger than the optimal value of the LP-relaxation.

ILP vs LP-relaxation

- For a maximisation problem:

- The optimal value of the ILP is not larger than the optimal value of the LP-relaxation.

- The ratio

  max_value(LP-relaxation) / max_value(LP)

  is called the integrality gap of the LP-formulation.

Let’s put some facts together.

Let’s put some facts together.

- The Max-Flow LP and the Min-Cut LP-relaxation are duals of each other.

Let’s put some facts together.

- The Max-Flow LP and the Min-Cut LP-relaxation are duals of each other.
- By *strong duality*, it holds that the optimal value of the Max-Flow LP is equal to the optimal value of the Min-Cut LP-relaxation.

Let’s put some facts together.

- The Max-Flow LP and the Min-Cut LP-relaxation are duals of each other.
- By *strong duality*, it holds that the optimal value of the Max-Flow LP is equal to the optimal value of the Min-Cut LP-relaxation.
- By the *Max-Flow-Min-Cut Theorem,* the value of the maximum flow is *equal* to the capacity of the minimum cut.

Let’s put some facts together.

- The Max-Flow LP and the Min-Cut LP-relaxation are duals of each other.
- By *strong duality*, it holds that the optimal value of the Max-Flow LP is equal to the optimal value of the Min-Cut LP-relaxation.
- By the *Max-Flow-Min-Cut Theorem,* the value of the maximum flow is *equal* to the capacity of the minimum cut.
- That can only mean one thing:

Let’s put some facts together.

- The Max-Flow LP and the Min-Cut LP-relaxation are duals of each other.
- By *strong duality*, it holds that the optimal value of the Max-Flow LP is equal to the optimal value of the Min-Cut LP-relaxation.
- By the *Max-Flow-Min-Cut Theorem,* the value of the maximum flow is *equal* to the capacity of the minimum cut.
- That can only mean one thing:
  - The value of the Min-Cut LP-relaxation is equal to the value of the Min-Cut LP.

Let’s put some facts together.

- The Max-Flow LP and the Min-Cut LP-relaxation are duals of each other.
- By *strong duality*, it holds that the optimal value of the Max-Flow LP is equal to the optimal value of the Min-Cut LP-relaxation.
- By the *Max-Flow-Min-Cut Theorem,* the value of the maximum flow is *equal* to the capacity of the minimum cut.
- That can only mean one thing:
  - The value of the Min-Cut LP-relaxation is equal to the value of the Min-Cut LP.
  - In other words, the Min-Cut LP-formulation has integrality gap *1*.

Let’s put some facts together.

- The Max-Flow LP and the Min-Cut LP-relaxation are duals of each other.
- By *strong duality*, it holds that the optimal value of the Max-Flow LP is equal to the optimal value of the Min-Cut LP-relaxation.
- By the *Max-Flow-Min-Cut Theorem,* the value of the maximum flow is *equal* to the capacity of the minimum cut.
- That can only mean one thing:
  - The value of the Min-Cut LP-relaxation is equal to the value of the Min-Cut LP.
  - In other words, the Min-Cut LP-formulation has integrality gap *1*.
  - In other words, the Min-Cut LP has *an integer optimal solution*.

Back to Maximum Flow

- What if we wanted an integer flow instead of any flow?

$$
\begin{aligned}
\text{maximise} \quad & \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} \\
\text{subject to} \quad & f_{uv} \leq c_{uv}, && \text{for each} \quad u, v \in V \\
& \sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, && \text{for each } u \in V - \{s, t\} \\
& f_{uv} \geq 0, && \text{for each} \quad u, v \in V
\end{aligned}
$$

Back to Maximum Flow

- What if we wanted an integer flow instead of any flow?

$$\begin{aligned}
\text{maximise} \quad & \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs} \\
\text{subject to} \quad & f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V \\
& \sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\} \\
& f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V \\
& f_{uv} \in \mathbb{R}, \quad \text{for each } u, v \in V
\end{aligned}$$

Back to Maximum Flow

- Does the LP-relaxation of this ILP always have an integer solution?

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

$$f_{uv} \in \mathbb{R}, \quad \text{for each } u, v \in V$$

LPs for Max-Flow and Min-Cut

LPs for Max-Flow and Min-Cut

- The Max-Flow problem for integer flows can be written as an ILP.

LPs for Max-Flow and Min-Cut

- The Max-Flow problem for integer flows can be written as an ILP.
- The Min-Cut problem can be written as an ILP (cuts are always integers).

LPs for Max-Flow and Min-Cut

- The Max-Flow problem for integer flows can be written as an ILP.
- The Min-Cut problem can be written as an ILP (cuts are always integers).
- We can write the LP-relaxations of those two ILPs.

LPs for Max-Flow and Min-Cut

- The Max-Flow problem for integer flows can be written as an ILP.

- The Min-Cut problem can be written as an ILP (cuts are always integers).

- We can write the LP-relaxations of those two ILPs.

  - For Max-Flow, it finds the maximum fractional flow.

LPs for Max-Flow and Min-Cut

- The Max-Flow problem for integer flows can be written as an ILP.

- The Min-Cut problem can be written as an ILP (cuts are always integers).

- We can write the LP-relaxations of those two ILPs.

  - For Max-Flow, it finds the maximum fractional flow.

  - For Min-Cut, it finds the minimum “fractional” cut.

LPs for Max-Flow and Min-Cut

- The Max-Flow problem for integer flows can be written as an ILP.

- The Min-Cut problem can be written as an ILP (cuts are always integers).

- We can write the LP-relaxations of those two ILPs.
  - For Max-Flow, it finds the maximum fractional flow.
  - For Min-Cut, it finds the minimum "fractional" cut.

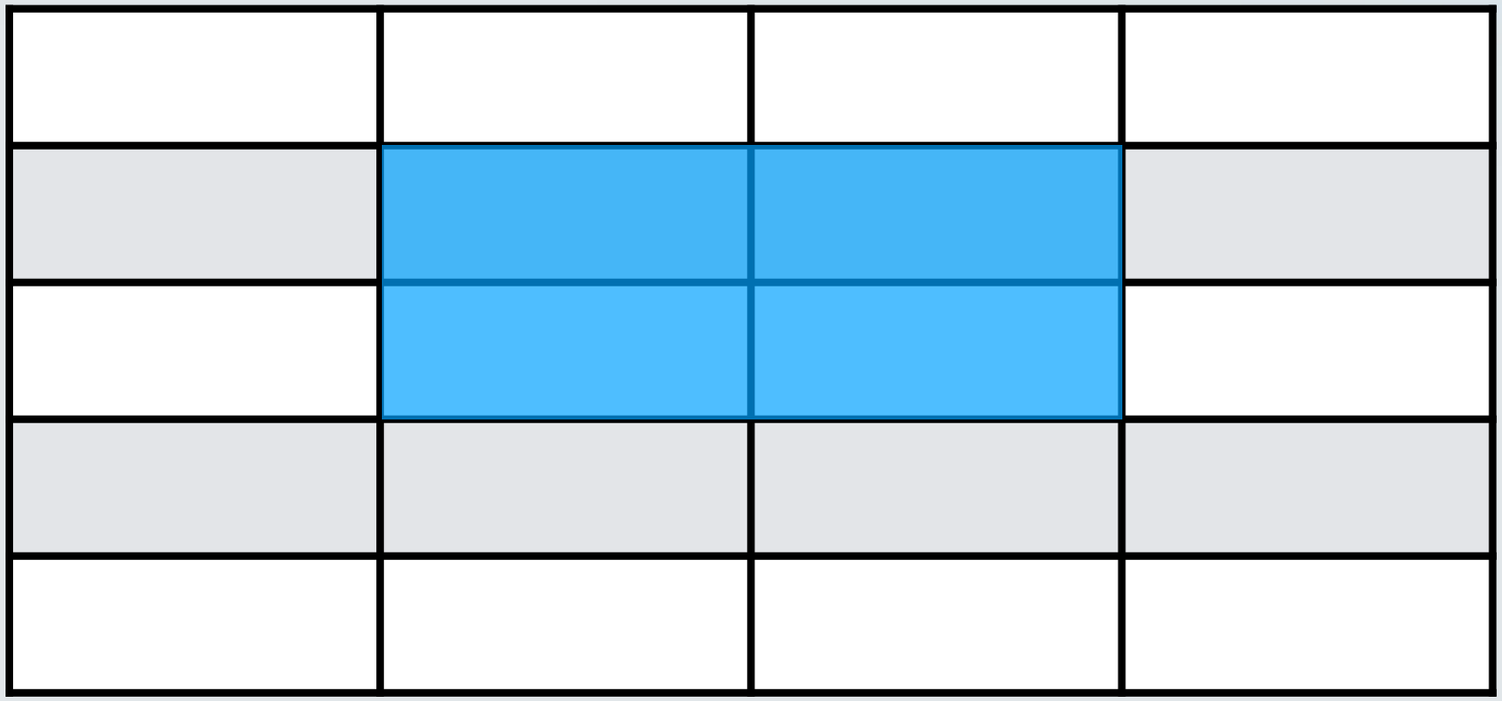
- If we solve those LP-relaxations, we will get *integer solutions*.

Totally Unimodular Matrices

- Let A be a real *m* x *n* matrix. Suppose that every square submatrix of A has determinant in {*0*, *+1*, *-1*}. Then A is *totally unimodular*.

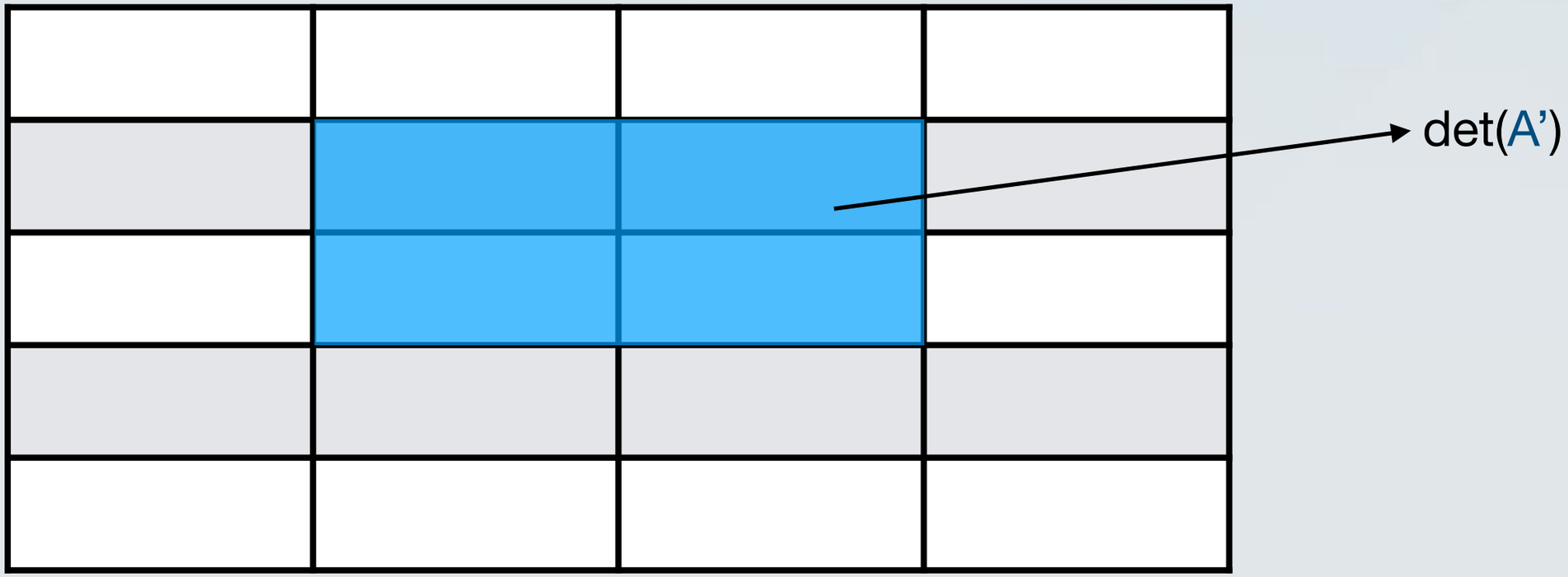
Totally Unimodular Matrices

- Let A be a real *m* x *n* matrix. Suppose that every square submatrix of A has determinant in {*0*, *+1*, *-1*}. Then A is *totally unimodular*.



Totally Unimodular Matrices

- Let A be a real *m* x *n* matrix. Suppose that every square submatrix of A has determinant in {*0*, *+1*, *-1*}. Then A is *totally unimodular*.



Total Unimodularity

- If the constraint matrix A is totally unimodular and b is an *integer vector*, then the LP has an *integer solution*.

- The Max-Flow and Min-Cut LP-relaxations admit integer solutions because their constraint matrices are totally unimodular.

$$\begin{aligned} \text{maximise} \quad & c^{\mathrm{T}} x \\ \text{subject to} \quad & Ax \leq b, \\ & x \geq 0 \end{aligned}$$

To be more precise

To be more precise

- Lemma: Suppose A is a totally unimodular matrix and b is an integer vector. Then every extreme point of

  P = {x: Ax < b} is integral.

To be more precise

- Lemma: Suppose A is a totally unimodular matrix and b is an integer vector. Then every extreme point of

  P = {x: Ax < b} is integral.

- Claim: Suppose A is totally unimodular. Then the matrix A’ = $(A\ -A\ I\ -\ I)^T$ is also totally unimodular. (tutorial)

To be more precise

- Lemma: Suppose A is a totally unimodular matrix and b is an integer vector. Then every extreme point of

  $P = \{x: Ax \leq b\}$ is integral.

- Claim: Suppose A is totally unimodular. Then the matrix $A' = (A\ -A\ I\ -\ I)^T$ is also totally unimodular. (tutorial)

- Corollary: Suppose A is a totally unimodular matrix and b is an integer vector. Then every extreme point of

  $P = \{x: Ax = b,\ 0 \leq x \leq c\}$ is integral.

Back to maximum flow

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Back to maximum flow

$$\text{maximise} \quad \sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$$

$$\text{subject to} \quad f_{uv} \leq c_{uv}, \ \text{for each} \quad u, v \in V$$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \ \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

Back to maximum flow

maximise $\sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$

subject to $f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$

$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$

$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$

Back to maximum flow

maximise $\sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$

subject to $f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$

$$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$$

$$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$$

$0 \leq x \leq c$

Back to maximum flow

maximise $\sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$

subject to $f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$

$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$

$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$

$0 \leq x \leq c$

Back to maximum flow

maximise $\sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$

subject to $f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$

$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$

$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$

$0 \leq x \leq c$

$Ax = 0$

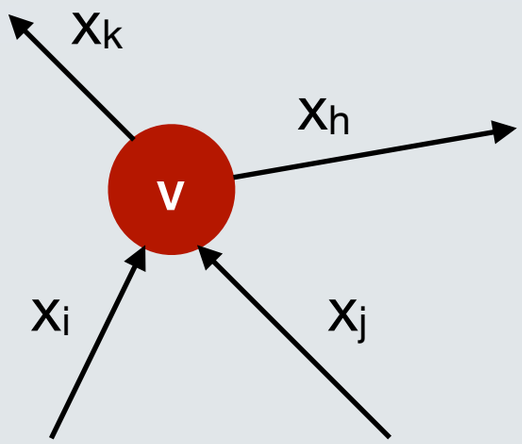
Back to maximum flow

maximise $\sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$

subject to $f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$

$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$

$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$



$0 \leq x \leq c$

$Ax = 0$

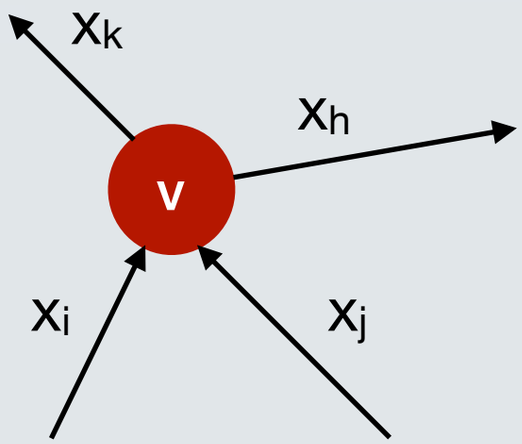
Back to maximum flow

maximise $\sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$

subject to $f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$

$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$

$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$



$x_k + x_h - x_i - x_j = 0$

$0 \leq x \leq c$

$Ax = 0$

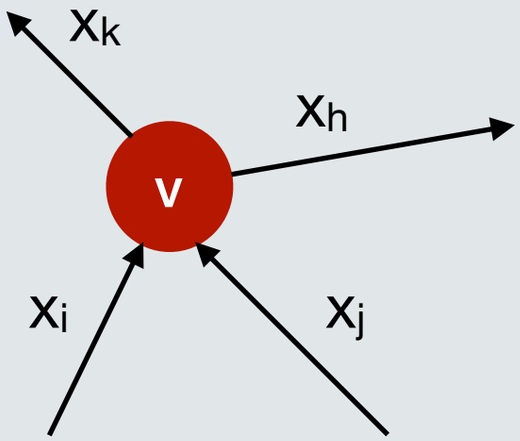
Back to maximum flow

maximise $\sum_{v \in V} f_{sv} - \sum_{v \in V} f_{vs}$

subject to $f_{uv} \leq c_{uv}, \quad \text{for each} \quad u, v \in V$

$\sum_{v \in V} f_{vu} = \sum_{v \in V} f_{uv}, \quad \text{for each } u \in V - \{s, t\}$

$f_{uv} \geq 0, \quad \text{for each} \quad u, v \in V$



$x_k + x_h - x_i - x_j = 0$

$0 \leq x \leq c$

$Ax = 0$

$A_{vi} = A_{vj} = -1$

$A_{vk} = A_{vh} = 1$

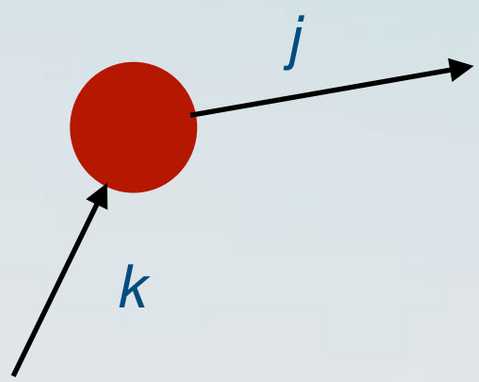
Back to maximum flow

- Consider the *incidence matrix* of the flow network (without s and t):
  - A_{ij} = *1* if edge *j* **starts** at node *i* in G_f.
  - A_{ij} = *-1* if edge *j* **ends** at node *i* in G_f.
  - A_{ij} = *0* otherwise.

Nodes/Edges		*j*	*k*
i		1	-1

Back to maximum flow

- Consider the *incidence matrix* of the flow network (without s and t):
  - A_{ij} = *1* if edge *j* **starts** at node *i* in G_f.
  - A_{ij} = *-1* if edge *j* **ends** at node *i* in G_f.
  - A_{ij} = *0* otherwise.



Nodes/Edges		*j*	*k*
i		1	-1

Back to maximum flow

- Consider the *incidence matrix* of the flow network (without s and t):

Nodes/Edges		*j*	*k*
i		1	-1

Back to maximum flow

- Consider the *incidence matrix* of the flow network (without s and t):
  - This is precisely the matrix A of the max flow LP.

Nodes/Edges		*j*	*k*
i		1	-1

Back to maximum flow

- Consider the *incidence matrix* of the flow network (without s and t):
  - This is precisely the matrix A of the max flow LP.
  - It suffices to prove that A is totally unimodular, by Corollary.

Nodes/Edges		*j*	*k*
i		1	-1

Back to maximum flow

- Consider the *incidence matrix* of the flow network (without s and t):
  - This is precisely the matrix A of the max flow LP.
  - It suffices to prove that A is totally unimodular, by Corollary.
  - Lemma: The incidence matrix of any directed graph is totally unimodular. (tutorial).

Nodes/Edges		*j*	*k*
i		1	-1