Algorithms and Data Structures

Modelling with Linear Programs

Modelling

Modelling

Knowing how to solve linear programs is very useful.

Modelling

Knowing how to solve linear programs is very useful.

We might never have to solve one by hand, but we can understand the basic principles of linear programming.

Modelling

Knowing how to solve linear programs is very useful.

We might never have to solve one by hand, but we can understand the basic principles of linear programming.

In practice: There are many fantastic LP solvers (e.g., CPLEX, Gurobi, whatever-your-favourite-library-of-your-favourite-programming-language-uses, etc).

Modelling

Knowing how to solve linear programs is very useful.

We might never have to solve one by hand, but we can understand the basic principles of linear programming.

In practice: There are many fantastic LP solvers (e.g., CPLEX, Gurobi, whatever-your-favourite-library-of-your-favourite-programming-language-uses, etc).

It suffices to formulate/model/express a problem as an LP and then ask one of those solvers for the solution.

Managing a Production Facility

Managing a Production Facility

Consider a production facility for a manufacturing company, which can produce products $1, \ldots, n$.

Managing a Production Facility

Consider a production facility for a manufacturing company, which can produce products $1,\ldots,n$.

The products are manufactured out of raw materials. There are m different raw materials $1,\ldots,m$.

Managing a Production Facility

Consider a production facility for a manufacturing company, which can produce products $1,\ldots,n$.

The products are manufactured out of raw materials. There are m different raw materials $1,\ldots,m$.

For each raw material i, there is a known amount b_i in stock.

Managing a Production Facility

Consider a production facility for a manufacturing company, which can produce products $1, \ldots, n$.

The products are manufactured out of raw materials. There are m different raw materials $1, \ldots, m$.

For each raw material i, there is a known amount b_i in stock.

Each raw material i has a unit cost ρ_i.

Managing a Production Facility

Consider a production facility for a manufacturing company, which can produce products $1,\ldots,n$.

The products are manufactured out of raw materials. There are m different raw materials $1,\ldots,m$.

For each raw material i, there is a known amount b_i in stock.

Each raw material i has a unit cost ρ_i.

Each produce is made from known amounts of raw materials. Producing one unit of product j requires α_{ij} units of material i.

Managing a Production Facility

Consider a production facility for a manufacturing company, which can produce products $1,\ldots,n$.

The products are manufactured out of raw materials. There are m different raw materials $1,\ldots,m$.

For each raw material i, there is a known amount b_i in stock.

Each raw material i has a unit cost ρ_i.

Each produce is made from known amounts of raw materials. Producing one unit of product j requires α_{ij} units of material i.

Product j can be sold in the market for σ_j pounds per unit.

Managing a Production Facility

Consider a production facility for a manufacturing company, which can produce products $1, \ldots, n$.

The products are manufactured out of raw materials. There are m different raw materials $1, \ldots, m$.

For each raw material i, there is a known amount b_i in stock.

Each raw material i has a unit cost ρ_i.

Each produce is made from known amounts of raw materials. Producing one unit of product j requires α_{ij} units of material i.

Product j can be sold in the market for σ_j pounds per unit.

The production manager would like to use the materials in stock to extract as much revenue (= price - cost) as possible.

Step 1: Choosing the variables

Step 1: Choosing the variables

We already know: $b_i, \rho_i, \alpha_{ij}, \sigma_j$

Step 1: Choosing the variables

We already know: $b_i, \rho_i, \alpha_{ij}, \sigma_j$

These are constants or parameters of our problem, not variables.

Step 1: Choosing the variables

We already know: $b_i, \rho_i, \alpha_{ij}, \sigma_j$

These are constants or parameters of our problem, not variables.

The variables are chosen by us, trying to model the problem accurately.

Step 1: Choosing the variables

We already know: $b_i, \rho_i, \alpha_{ij}, \sigma_j$

These are constants or parameters of our problem, not variables.

The variables are chosen by us, trying to model the problem accurately.

What should we choose for the variables here?

Step 1: Choosing the variables

We already know: $b_i, \rho_i, \alpha_{ij}, \sigma_j$

These are constants or parameters of our problem, not variables.

The variables are chosen by us, trying to model the problem accurately.

What should we choose for the variables here?

x_j : number of units of product j that we will produce.

Step 2: Writing the objective function

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Revenue: Price - Cost

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Revenue: Price - Cost

What is the price of one unit of product j?

Managing a Production Facility

Consider a production facility for a manufacturing company, which can produce products $1, \ldots, n$.

The products are manufactured out of raw materials. There are m different raw materials $1, \ldots, m$.

For each raw material i, there is a known amount b_i in stock.

Each raw material i has a unit cost ρ_i.

Each produce is made from known amounts of raw materials. Producing one unit of product j requires α_{ij} units of material i.

Product j can be sold in the market for σ_j pounds per unit.

The production manager would like to use the materials in stock to extract as much revenue (= price - cost) as possible.

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Revenue: Price - Cost

What is the price of one unit of product j? σ_{ij}

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Revenue: Price - Cost

What is the price of one unit of product j? σ_{ij}

What is the cost of producing one unit of product j?

Managing a Production Facility

Consider a production facility for a manufacturing company, which can produce products $1,\ldots,n$.

The products are manufactured out of raw materials. There are m different raw materials $1,\ldots,m$.

For each raw material i, there is a known amount b_i in stock.

Each raw material i has a unit cost ρ_i.

Each produce is made from known amounts of raw materials. Producing one unit of product j requires α_{ij} units of material i.

Product j can be sold in the market for σ_j pounds per unit.

The production manager would like to use the materials in stock to extract as much revenue (= price - cost) as possible.

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Revenue: Price - Cost

What is the price of one unit of product j? σ_{ij}

What is the cost of producing one unit of product j? $\sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Revenue: Price - Cost

What is the price of one unit of product j? σ_{ij}

What is the cost of producing one unit of product j? $\sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

What is the revenue from one unit of j?

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Revenue: Price - Cost

What is the price of one unit of product j? σ_{ij}

What is the cost of producing one unit of product j? $\sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

What is the revenue from one unit of j? $c_j = \sigma_{ij} - \sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Revenue: Price - Cost

What is the price of one unit of product j? σ_{ij}

What is the cost of producing one unit of product j? $\sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

What is the revenue from one unit of j? $c_j = \sigma_{ij} - \sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

What is the revenue from all the units of of j?

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Revenue: Price - Cost

What is the price of one unit of product j? σ_{ij}

What is the cost of producing one unit of product j? $\sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

What is the revenue from one unit of j? $c_j = \sigma_{ij} - \sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

What is the revenue from all the units of of j? $c_j \cdot x_j$

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Revenue: Price - Cost

What is the price of one unit of product j? σ_{ij}

What is the cost of producing one unit of product j? $\sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

What is the revenue from one unit of j? $c_j = \sigma_{ij} - \sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

What is the revenue from all the units of of j? $c_j \cdot x_j$

What is the revenue in total?

Step 2: Writing the objective function

x_j : number of units of product j that we will produce.

Revenue: Price - Cost

What is the price of one unit of product j? σ_{ij}

What is the cost of producing one unit of product j? $\sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

What is the revenue from one unit of j? $c_j = \sigma_{ij} - \sum_{i=1}^{m} \alpha_{ij} \cdot \rho_i$

What is the revenue from all the units of of j? $c_j \cdot x_j$

What is the revenue in total? $\sum_{i=1}^{n} c_j x_j$

Our LP formulation

Maximise $\sum_{i=1}^{n} c_j x_j$

subject to $?$

Step 3: Writing the constraints

Step 3: Writing the constraints

x_j : number of units of product j that we will produce.

Step 3: Writing the constraints

x_j : number of units of product j that we will produce.

1. We cannot produce negative amounts:

Step 3: Writing the constraints

x_j : number of units of product j that we will produce.

1. We cannot produce negative amounts:

$$x_j \geq 0 \text{ for } j = 1, \ldots, n$$

Step 3: Writing the constraints

x_j : number of units of product j that we will produce.

1. We cannot produce negative amounts:

$$x_j \geq 0 \text{ for } j = 1, \ldots, n$$

2. We cannot produce more than the raw material allows:

Managing a Production Facility

Consider a production facility for a manufacturing company, which can produce products $1, \ldots, n$.

The products are manufactured out of raw materials. There are m different raw materials $1, \ldots, m$.

For each raw material i, there is a known amount b_i in stock.

Each raw material i has a unit cost ρ_i.

Each produce is made from known amounts of raw materials. Producing one unit of product j requires α_{ij} units of material i.

Product j can be sold in the market for σ_j pounds per unit.

The production manager would like to use the materials in stock to extract as much revenue (= price - cost) as possible.

Step 3: Writing the constraints

x_j : number of units of product j that we will produce.

1. We cannot produce negative amounts:

$$x_j \geq 0 \text{ for } j = 1, \ldots, n$$

2. We cannot produce more than the raw material allows:

Step 3: Writing the constraints

x_j : number of units of product j that we will produce.

1. We cannot produce negative amounts:

$$x_j \geq 0 \text{ for } j = 1, \ldots, n$$

2. We cannot produce more than the raw material allows:

$$\sum_{j=1}^{m} \alpha_{ij} x_j \leq b_i \text{ for } i = 1, \ldots, m$$

Our LP formulation

Maximise $$\sum_{i=1}^{n} c_j x_j$$

subject to $$\sum_{j=1}^{n} \alpha_{ij} \cdot x_j \leq b_i \quad \textbf{for all } i = 1, \ldots, m$$

$$x_j \geq 0 \quad \textbf{for all } j = 1, \ldots, n$$

Linear Regression

Linear Regression

We have measured the height and weight of 100 people, collecting data points $h_1, \ldots, h_{100}$ and $w_1, \ldots, w_{100}$.

Linear Regression

We have measured the height and weight of 100 people, collecting data points $h_1, \ldots, h_{100}$ and $w_1, \ldots, w_{100}$.

We wish to come up with a formula that predicts the weight based on the height via a linear function, namely $w_i = ah_i + b$

Linear Regression

We have measured the height and weight of 100 people, collecting data points $h_1, \ldots, h_{100}$ and $w_1, \ldots, w_{100}$.

We wish to come up with a formula that predicts the weight based on the height via a linear function, namely $w_i = ah_i + b$

This formula does not fit our observations 100%, so we would like to find the values of a and b that best approximate our observations.

Linear Regression

We have measured the height and weight of 100 people, collecting data points $h_1, \ldots, h_{100}$ and $w_1, \ldots, w_{100}$.

We wish to come up with a formula that predicts the weight based on the height via a linear function, namely $w_i = ah_i + b$

This formula does not fit our observations 100%, so we would like to find the values of a and b that best approximate our observations.

In particular, we would like to minimise the worst-case error of the prediction, i.e., to minimise

$$e = \max_{i=1}^{100} | w_i - (ah_i + b) |$$

Linear Regression

In particular, we would like to minimise the worst-case error of the prediction, i.e., to minimise

$$e = \max_{i=1}^{100} | w_i - (ah_i + b) |$$

Linear Regression

In particular, we would like to minimise the worst-case error of the prediction, i.e., to minimise

$$e = \max_{i=1}^{100} |w_i - (ah_i + b)|$$

Let's try to devise a linear program for that.

Linear Regression

In particular, we would like to minimise the worst-case error of the prediction, i.e., to minimise

$$e = \max_{i=1}^{100} | w_i - (ah_i + b) |$$

Let's try to devise a linear program for that.

Variables?

A first attempt

minimise e

subject to $a, b \in \mathbb{R}$ (i.e., no constraint)

A first attempt

minimise $\max_{i=1}^{100} |w_i - (ah_i + b)|$

subject to $a, b \in \mathbb{R}$ (i.e., no constraint)

A first attempt

What is wrong with this?

minimise $\max_{i=1}^{100} |w_i - (ah_i + b)|$

subject to $a, b \in \mathbb{R}$ (i.e., no constraint)

Removing the max

Let $e = \max_{i=1}^{100} |w_i - (ah_i + b)|$

This means that for each $i = 1,\ldots,100$, we have that

$$|w_i - (ah_i + b)| \leq e$$

Our new attempt

minimise e

subject to $|w_i - (ah_i + b)| \leq e$ for all $i = 1, \ldots, 100$

Removing the absolute value

Removing the absolute value

Let's look at $|x|$.

Removing the absolute value

Let's look at $|x|$.

It holds that:

Removing the absolute value

Let's look at $|x|$.

It holds that:

$$|x| = \max\{x, -x\}$$

Removing the absolute value

Let's look at $|x|$.

It holds that:

$$|x| = \max\{x, -x\}$$

Let's look at the constraint $|x| \leq e$

Removing the absolute value

Let's look at $|x|$.

It holds that:

$$|x| = \max\{x, -x\}$$

Let's look at the constraint $|x| \leq e$

This means that $\max\{x, -x\} \leq e$

Removing the absolute value

Let's look at $|x|$.

It holds that:

$$|x| = \max\{x, -x\}$$

Let's look at the constraint $|x| \leq e$

This means that $\max\{x, -x\} \leq e$

Which means that both $x \leq e$ and $-x \leq e$

Our new attempt

minimise e

subject to $|w_i - (ah_i + b)| \leq e$ for all $i = 1,\ldots,100$

Our new LP

minimise e

subject to $w_i - (ah_i + b) \leq e$ for all $i = 1,\ldots,100$

$-w_i + (ah_i + b) \leq e$ for all $i = 1,\ldots,100$

Linear Regression

Linear Regression

What if we wish to minimise the average error of the prediction, i.e., to minimise

$$e = \frac{1}{100} \sum_{i=1}^{100} | w_i - (ah_i + b) |$$

A first attempt

minimise $\frac{1}{100}\sum_{i=1}^{100} |w_i - (ah_i + b)|$

subject to $a, b \in \mathbb{R}$ (i.e., no constraint)

Removing the absolute value

Removing the absolute value

Let's look at $|x|$. Let $x' = |x|$.

Removing the absolute value

Let's look at $|x|$. Let $x' = |x|$.

It holds that:

Removing the absolute value

Let's look at $|x|$. Let $x' = |x|$.

It holds that:

$$x' = |x| = \max\{x, -x\}$$

Removing the absolute value

Let's look at $|x|$. Let $x' = |x|$.

It holds that:

$$x' = |x| = \max\{x, -x\}$$

i.e., $x \leq x'$ and $-x \leq x'$

Removing the absolute value

Let's look at $|x|$. Let $x' = |x|$.

It holds that:

$$x' = |x| = \max\{x, -x\}$$

i.e., $x \leq x'$ and $-x \leq x'$

We can add those as constraints.

Our new LP

minimise $\frac{1}{100}\sum_{i=1}^{100} x_i$

subject to $w_i - ah_i + b \leq x_i$ for all $i = 1,\ldots,100$

$-w_i + ah_i + b \leq x_i$ for all $i = 1,\ldots,100$

Our new LP

Should we worry that the LP will choose some x_i such that
$x_i > \max\{-w_i + ah_i + b, w_i - ah_i + b\}$?

minimise $\frac{1}{100}\sum_{i=1}^{100} x_i$

subject to $w_i - ah_i + b \leq x_i$ for all $i = 1,\ldots,100$

$-w_i + ah_i + b \leq x_i$ for all $i = 1,\ldots,100$

Integer Linear programming

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

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

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

$$x_j \text{ is integer}$$

Integer Linear programming

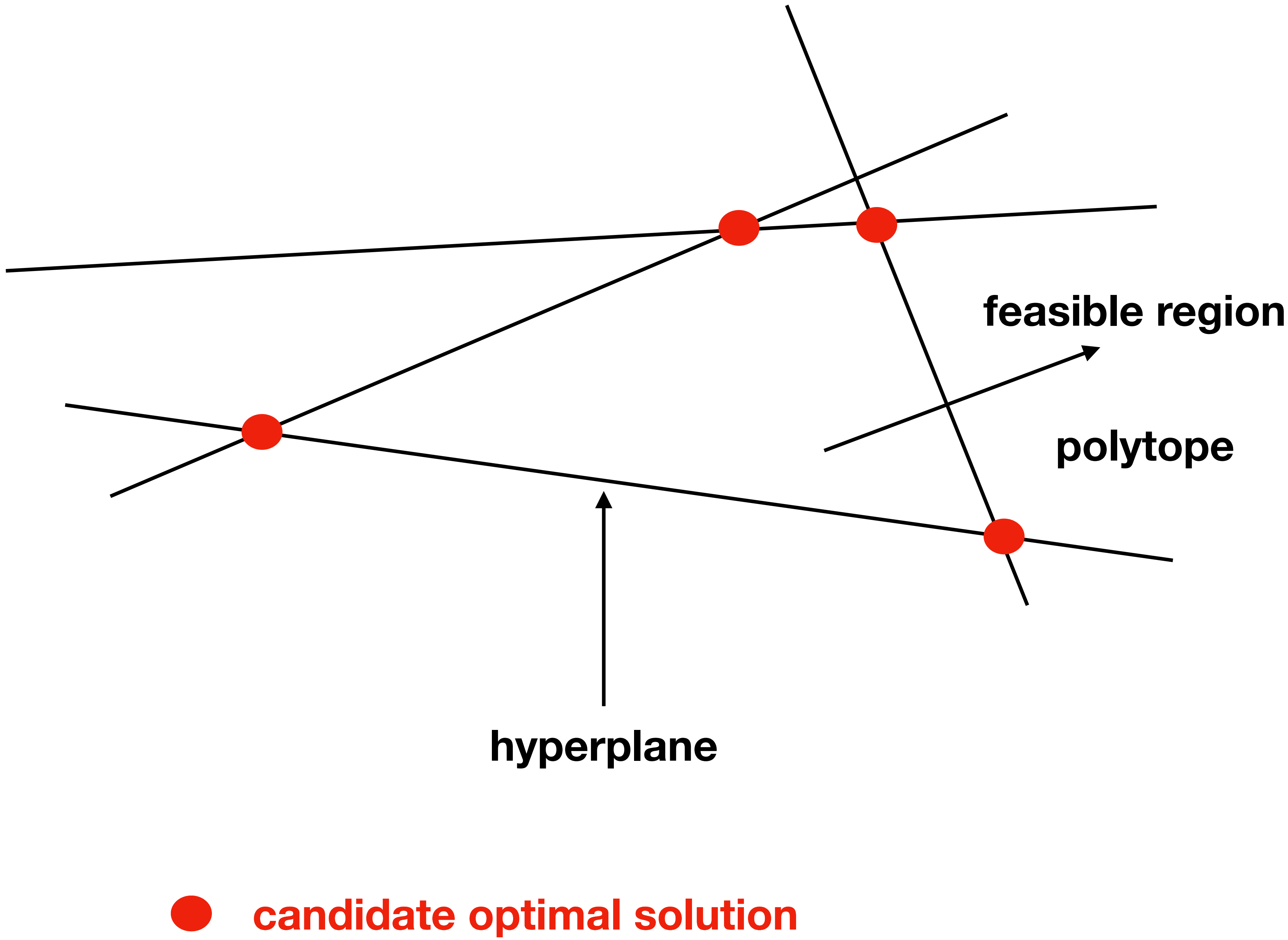
maximise $\sum_{j=1}^{n} c_j x_j$

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

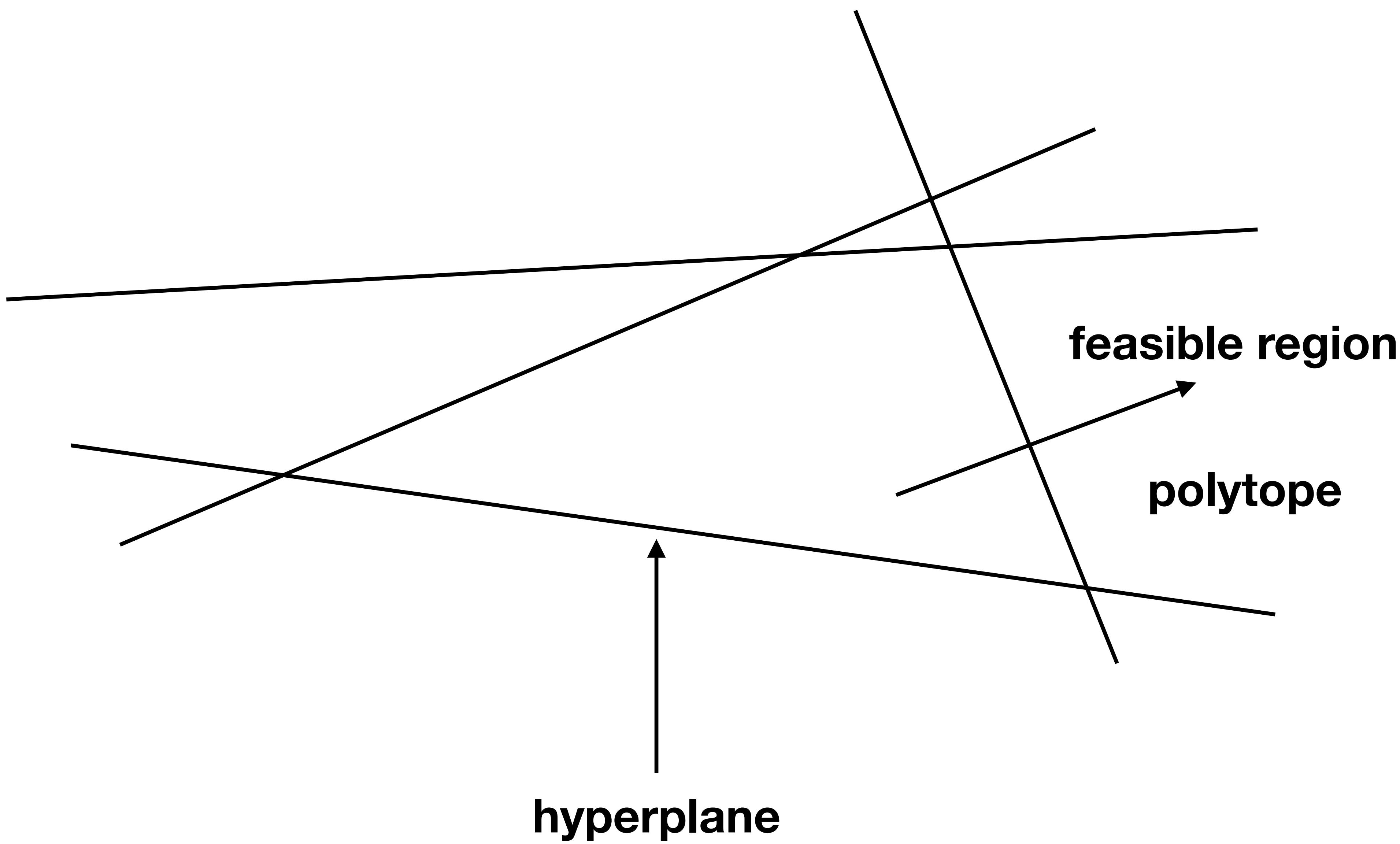
$x_j \geq 0, \quad j = 1, ..., n$

x_j is integer

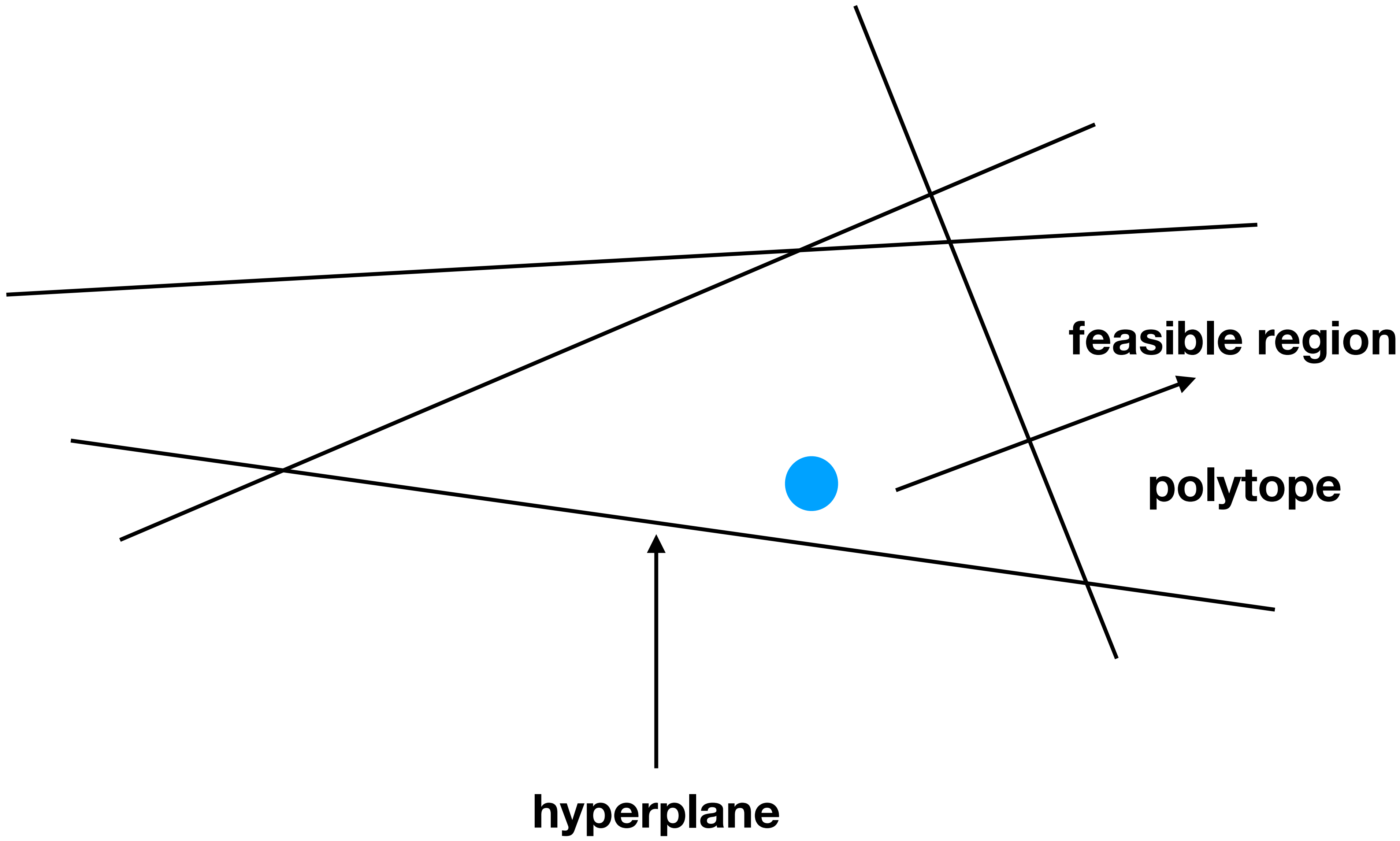
Feasible region



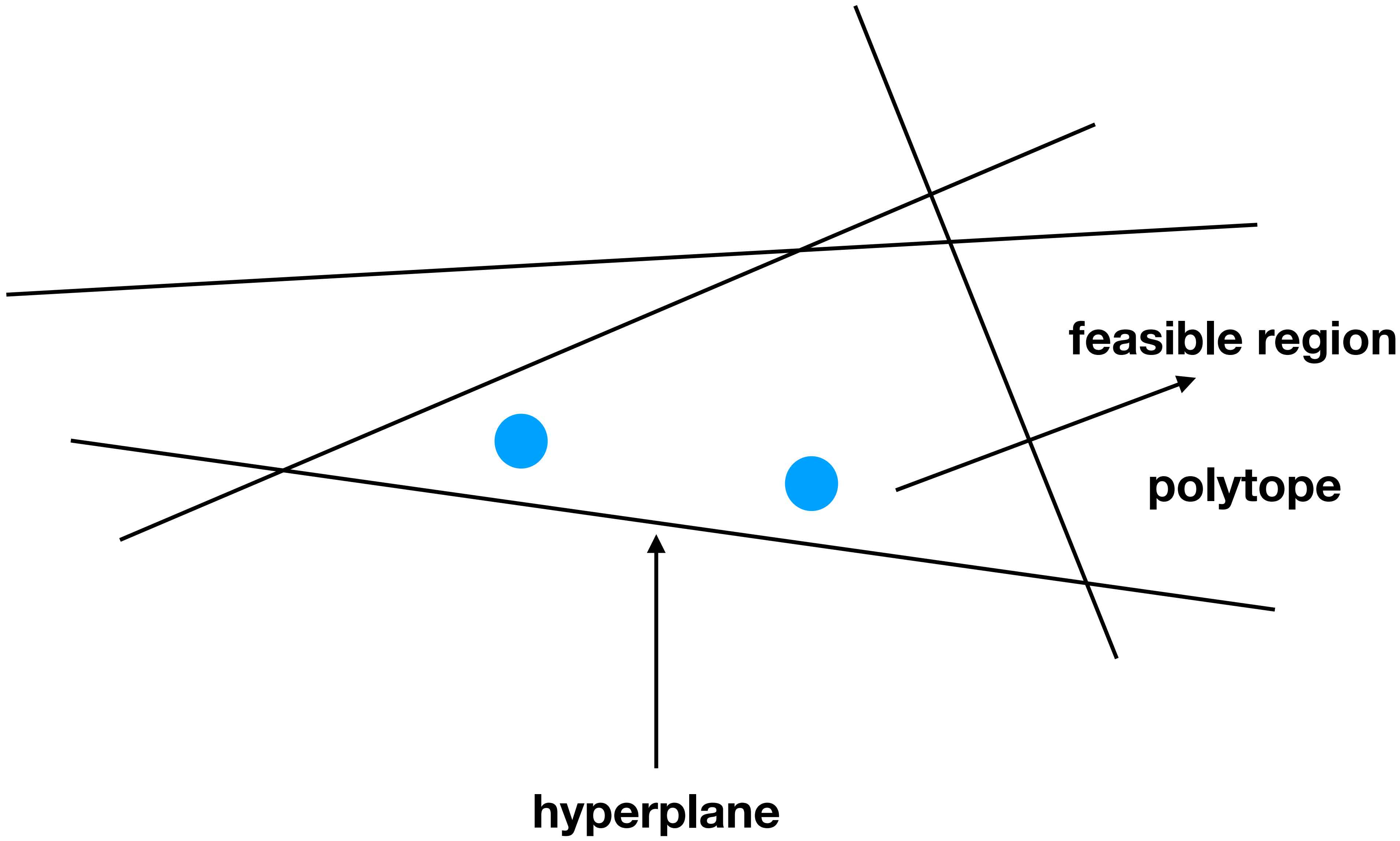
Feasible region



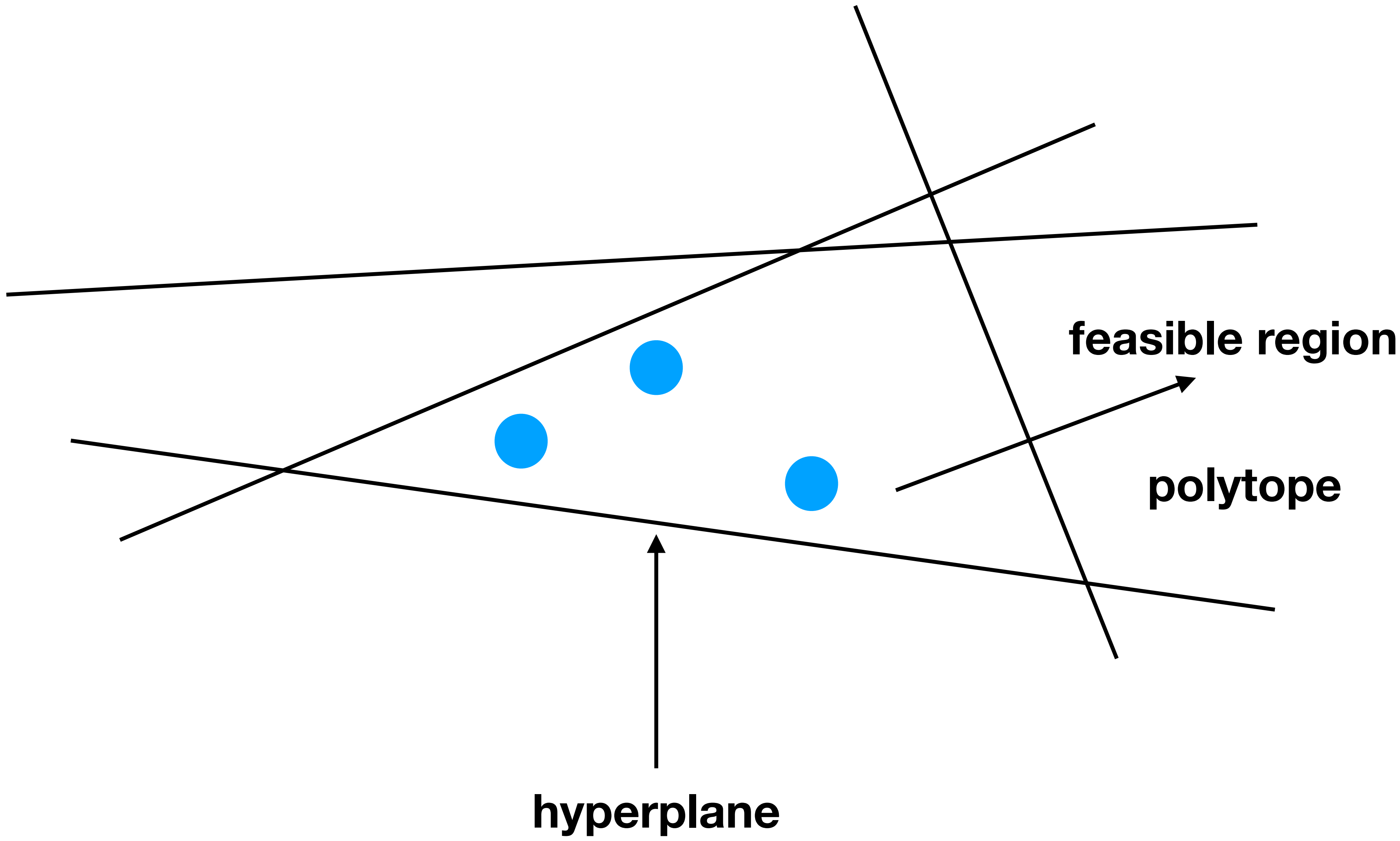
Feasible region



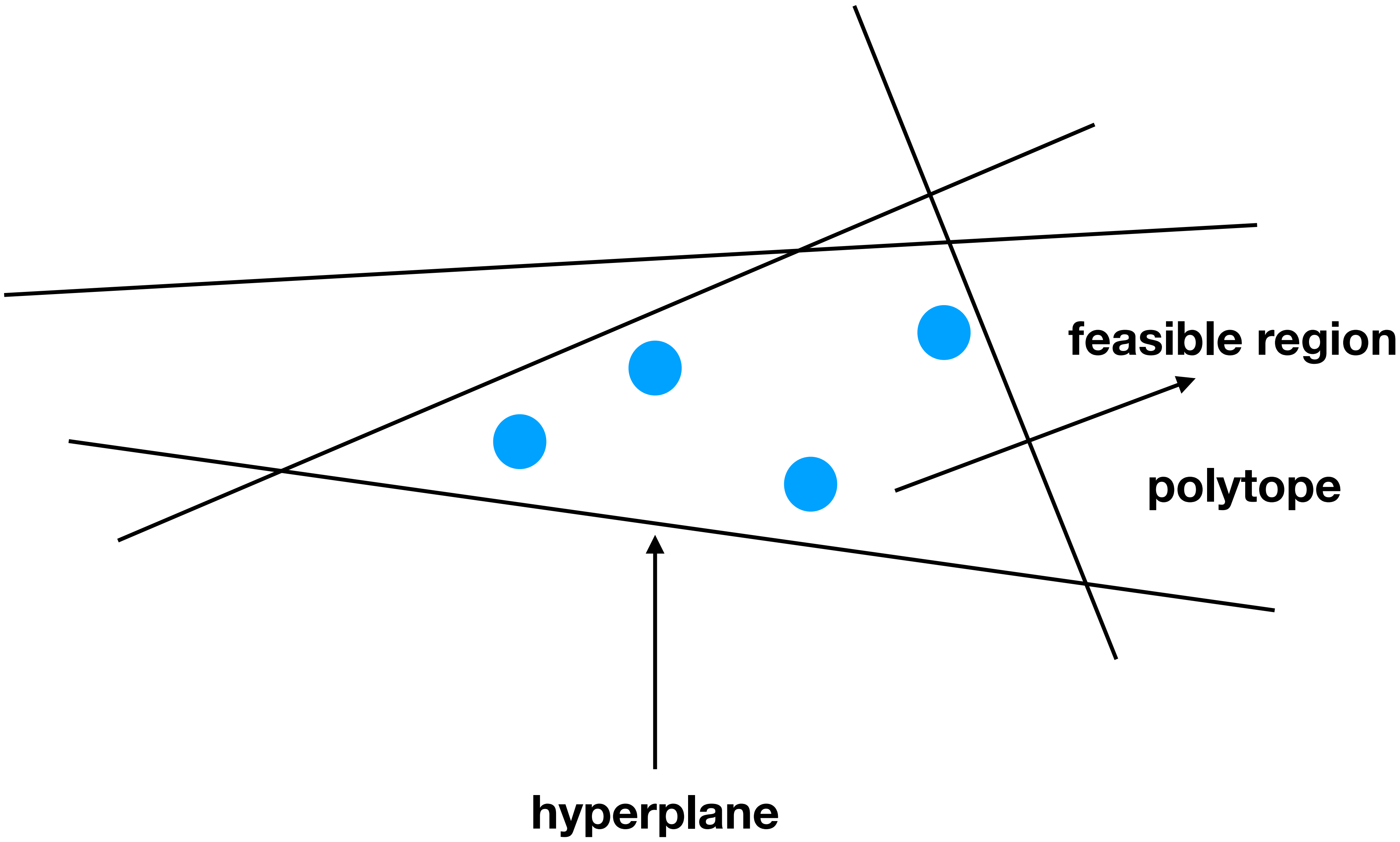
Feasible region



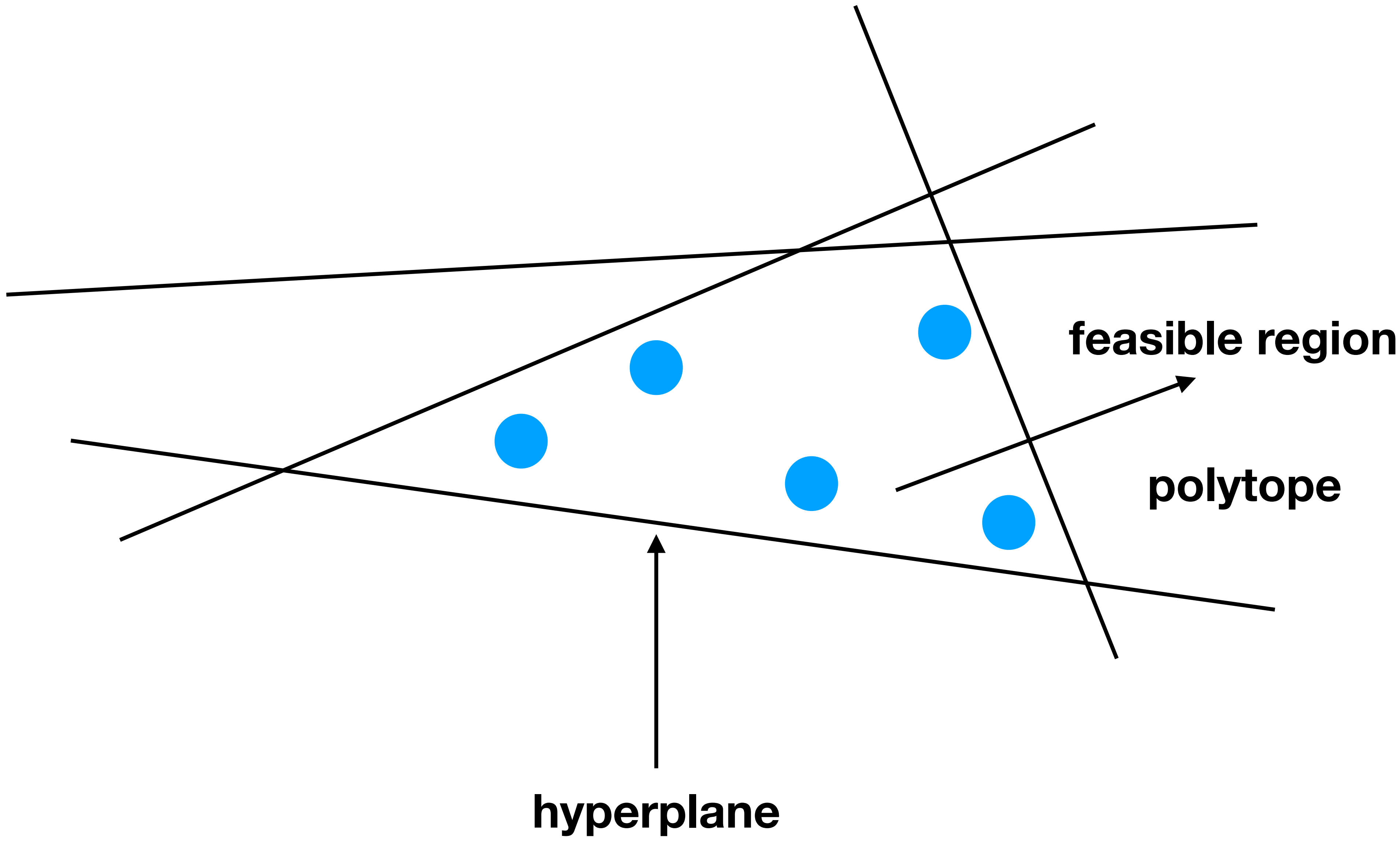
Feasible region



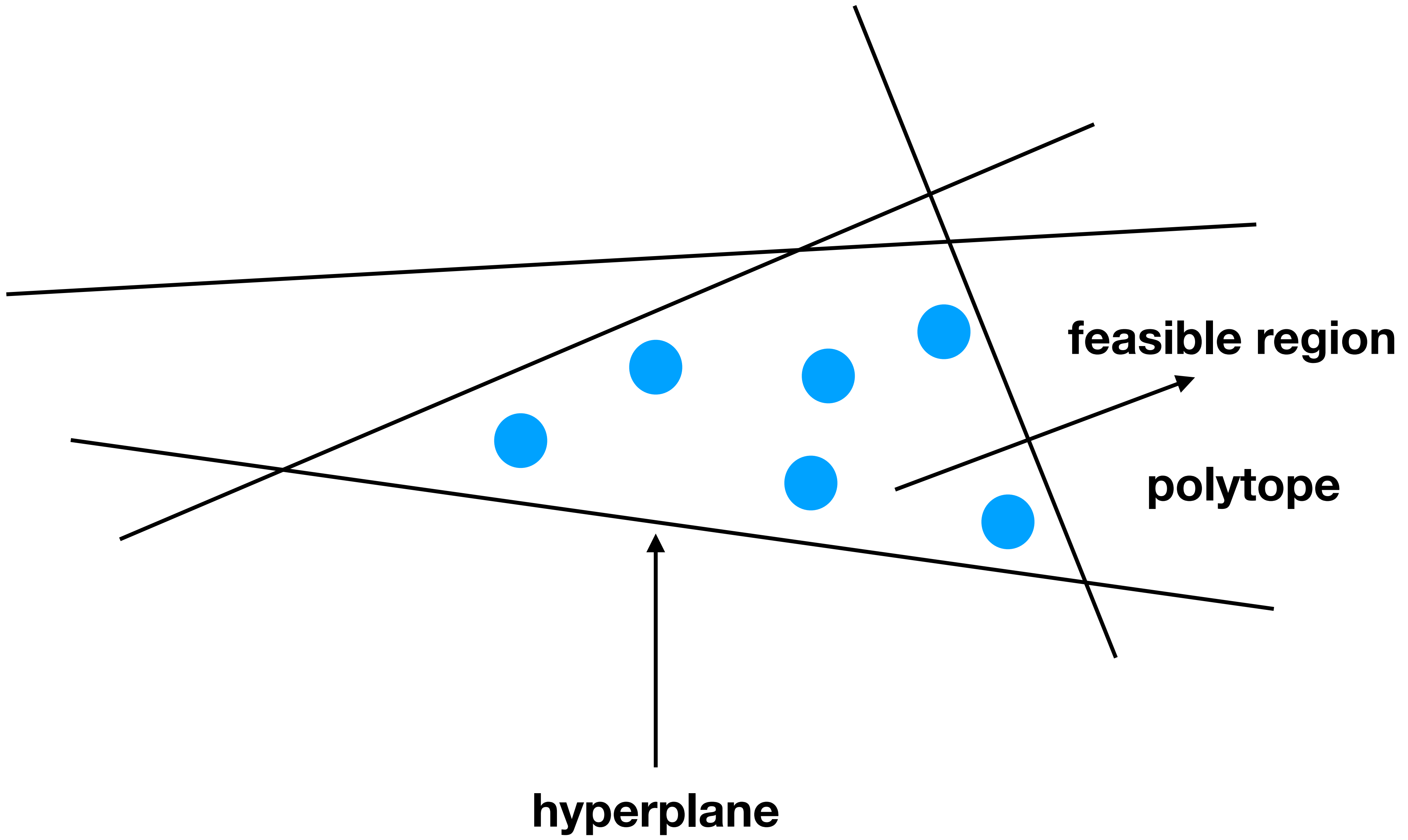
Feasible region



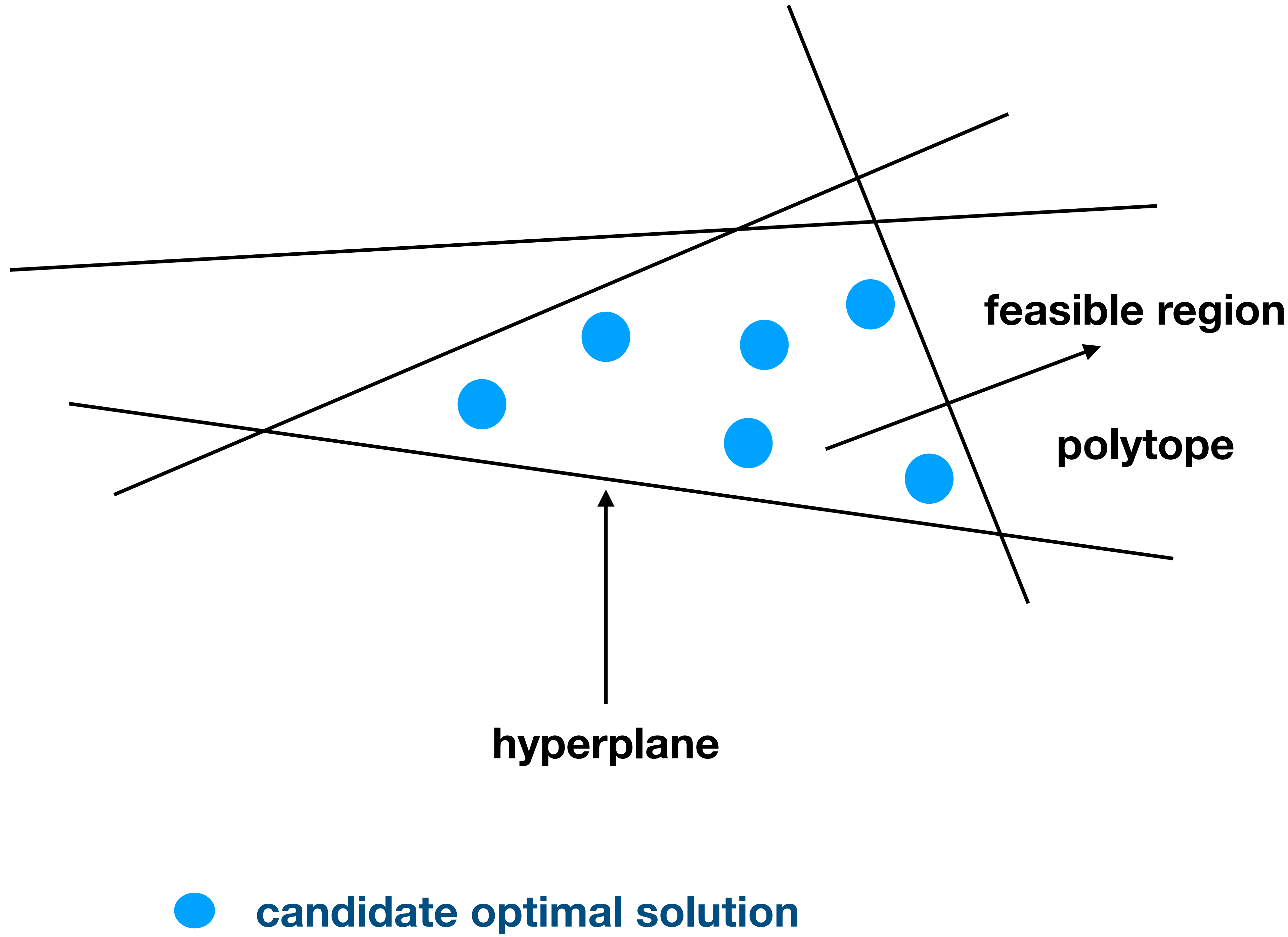
Feasible region



Feasible region



Feasible region



Modelling

Modelling

ILPs cannot be solved efficiently in general.

Modelling

ILPs cannot be solved efficiently in general.

The state-of-the-art solvers perform a lot of clever optimisations to solve them as fast as possible.

Modelling

ILPs cannot be solved efficiently in general.

The state-of-the-art solvers perform a lot of clever optimisations to solve them as fast as possible.

Some ILPs might take a really long time, but some may be solved in reasonable time.

Modelling

ILPs cannot be solved efficiently in general.

The state-of-the-art solvers perform a lot of clever optimisations to solve them as fast as possible.

Some ILPs might take a really long time, but some may be solved in reasonable time.

For many problems the ILP formulation beats all the other alternatives.

Modelling

ILPs cannot be solved efficiently in general.

The state-of-the-art solvers perform a lot of clever optimisations to solve them as fast as possible.

Some ILPs might take a really long time, but some may be solved in reasonable time.

For many problems the ILP formulation beats all the other alternatives.

Modelling as ILPs is a very useful skill.

Flight Scheduling

Flight Scheduling

A route is a journey from one destination to another (e.g., Edinburgh to Athens).

Flight Scheduling

A route is a journey from one destination to another (e.g., Edinburgh to Athens).

Each route might consist of several legs (e.g., Edinburgh to London departing at 6.30, London to Frankfurt departing at 10.00, Frankfurt to Athens departing at 16.00).

Flight Scheduling

A route is a journey from one destination to another (e.g., Edinburgh to Athens).

Each route might consist of several legs (e.g., Edinburgh to London departing at 6.30, London to Frankfurt departing at 10.00, Frankfurt to Athens departing at 16.00).

AlgoAir has a set of n specified routes and a set of m specified legs. The routes are denoted by $1,\ldots,n$ and for each leg, we have a parameter a_{ij} that specifies whether the leg is part of route j.

Flight Scheduling

A route is a journey from one destination to another (e.g., Edinburgh to Athens).

Each route might consist of several legs (e.g., Edinburgh to London departing at 6.30, London to Frankfurt departing at 10.00, Frankfurt to Athens departing at 16.00).

AlgoAir has a set of n specified routes and a set of m specified legs. The routes are denoted by $1,\ldots,n$ and for each leg, we have a parameter a_{ij} that specifies whether the leg is part of route j.

Every route j has an associated cost c_j.

Flight Scheduling

A route is a journey from one destination to another (e.g., Edinburgh to Athens).

Each route might consist of several legs (e.g., Edinburgh to London departing at 6.30, London to Frankfurt departing at 10.00, Frankfurt to Athens departing at 16.00).

AlgoAir has a set of n specified routes and a set of m specified legs. The routes are denoted by $1,\ldots,n$ and for each leg, we have a parameter a_{ij} that specifies whether the leg is part of route j.

Every route j has an associated cost c_j.

We would like to find a subset of the routes such that each leg is included in exactly one route.

Step 1: Choosing the variables

Step 1: Choosing the variables

We already know: c_j, a_{ij}

Step 1: Choosing the variables

We already know: c_j, a_{ij}

These are constants or parameters of our problem, not variables.

Step 1: Choosing the variables

We already know: c_j, a_{ij}

These are constants or parameters of our problem, not variables.

Very commonly used in ILP problems: Indicator variables

Step 1: Choosing the variables

We already know: c_j, a_{ij}

These are constants or parameters of our problem, not variables.

Very commonly used in ILP problems: Indicator variables

An indicator variable is 1 if something happens and 0 otherwise.

Step 1: Choosing the variables

We already know: c_j, a_{ij}

These are constants or parameters of our problem, not variables.

Very commonly used in ILP problems: Indicator variables

An indicator variable is 1 if something happens and 0 otherwise.

Here, we will let $x_j = 1$ if we select route j and $x_j = 0$ otherwise.

Step 2: Writing the objective function

Step 2: Writing the objective function

We want to minimise the total cost.

Step 2: Writing the objective function

We want to minimise the total cost.

The cost of route j is 1 if we schedule it, otherwise it is 0.

Step 2: Writing the objective function

We want to minimise the total cost.

The cost of route j is 1 if we schedule it, otherwise it is 0.

So what is the cost of route j?

Step 2: Writing the objective function

We want to minimise the total cost.

The cost of route j is 1 if we schedule it, otherwise it is 0.

So what is the cost of route j?

$$c_j \cdot x_j$$

Step 2: Writing the objective function

We want to minimise the total cost.

The cost of route j is 1 if we schedule it, otherwise it is 0.

So what is the cost of route j?

$$c_j \cdot x_j$$

What is the total cost of all the routes?

Step 2: Writing the objective function

We want to minimise the total cost.

The cost of route j is 1 if we schedule it, otherwise it is 0.

So what is the cost of route j?

$$c_j \cdot x_j$$

What is the total cost of all the routes?

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

Our LP formulation

Minimise $\sum_{i=1}^{n} c_j x_j$

subject to $?$

Step 3: Writing the constraints

Step 3: Writing the constraints

Every leg is included in exactly one route.

Step 3: Writing the constraints

Every leg is included in exactly one route.

Recall: $a_{ij} = 1$ iff leg i is part of the route (not necessarily included).

Step 3: Writing the constraints

Every leg is included in exactly one route.

Recall: $a_{ij} = 1$ iff leg i is part of the route (not necessarily included).

What is the total number of routes that i is a part of?

Step 3: Writing the constraints

Every leg is included in exactly one route.

Recall: $a_{ij} = 1$ iff leg i is part of the route (not necessarily included).

What is the total number of routes that i is a part of?

$$\sum_{j=1}^{n} a_{ij}$$

Step 3: Writing the constraints

Every leg is included in exactly one route.

Recall: $a_{ij} = 1$ iff leg i is part of the route (not necessarily included).

What is the total number of routes that i is a part of?

$$\sum_{j=1}^{n} a_{ij}$$

But some of these routes might not be included. What is the total number of *included* routes that i is a part of?

Step 3: Writing the constraints

Every leg is included in exactly one route.

Recall: $a_{ij} = 1$ iff leg i is part of the route (not necessarily included).

What is the total number of routes that i is a part of?

$$\sum_{j=1}^{n} a_{ij}$$

But some of these routes might not be included. What is the total number of *included* routes that i is a part of?

$$\sum_{j=1}^{n} a_{ij} x_j$$

Step 3: Writing the constraints

What is the total number of *included* routes that i is a part of?

$$\sum_{j=1}^{n} a_{ij} x_j$$

Step 3: Writing the constraints

What is the total number of *included* routes that i is a part of?

$$\sum_{j=1}^{n} a_{ij}x_j$$

How many are these?

Step 3: Writing the constraints

What is the total number of *included* routes that i is a part of?

$$\sum_{j=1}^{n} a_{ij} x_j$$

How many are these?

One!

Step 3: Writing the constraints

What is the total number of *included* routes that i is a part of?

$$\sum_{j=1}^{n} a_{ij}x_j$$

How many are these?

One!

$$\sum_{j=1}^{n} a_{ij}x_j = 1$$

Our ILP formulation

Minimise $\sum_{i=1}^{n} c_j x_j$

subject to $\sum_{j=1}^{n} a_{ij} x_j$ **for** $i = 1, \ldots, m$

Our ILP formulation

Anything else?

Minimise $\sum_{i=1}^{n} c_j x_j$

subject to $\sum_{j=1}^{n} a_{ij} x_j$ **for** $i = 1, \ldots, m$

Our ILP formulation

Anything else?

Minimise $\sum_{i=1}^{n} c_j x_j$

subject to $\sum_{j=1}^{n} a_{ij} x_j$ **for** $i = 1, \ldots, m$

$x_j \in \{0,1\}$ **for** $j = 1, \ldots, n$

The Travelling Salesman Problem

The Travelling Salesman Problem

A salesman needs to visit n cities, denoted $0, 1, \ldots, n-1$.

The Travelling Salesman Problem

A salesman needs to visit n cities, denoted $0, 1, \ldots, n-1$.

He starts from city 0 and would like to visit each city *exactly once*.

The Travelling Salesman Problem

A salesman needs to visit n cities, denoted $0,1,\ldots,n-1$.

He starts from city 0 and would like to visit each city *exactly once*.

There is a known distance c_{ij} between any pair of cities i, j.

The Travelling Salesman Problem

A salesman needs to visit n cities, denoted $0,1,\ldots,n-1$.

He starts from city 0 and would like to visit each city *exactly once*.

There is a known distance c_{ij} between any pair of cities i, j.

The salesman would like to minimise the total distance travelled.

The Travelling Salesman Problem

The Travelling Salesman Problem

A tour can be described as a sequence of cities $0, s_1, s_2, \ldots, s_{n-1}$.

The Travelling Salesman Problem

A tour can be described as a sequence of cities $0, s_1, s_2, \ldots, s_{n-1}$.

The total number of possible tours is equal to the permutation of $n - 1$ elements, i.e., $(n-1)!$

The Travelling Salesman Problem

A tour can be described as a sequence of cities $0, s_1, s_2, \ldots, s_{n-1}$.

The total number of possible tours is equal to the permutation of $n - 1$ elements, i.e., $(n - 1)!$

Enumeration is obviously too slow. We will use an ILP formulation approach instead and rely on our clever solvers to be faster than enumeration.

Step 1: Choosing the variables

Step 1: Choosing the variables

We already know: c_{ij}.

Step 1: Choosing the variables

We already know: c_{ij}.

One idea: Let $x_j = 1$ if the tour visits city j and 0 otherwise.

Step 1: Choosing the variables

We already know: c_{ij}.

One idea: Let $x_j = 1$ if the tour visits city j and 0 otherwise.

A better idea: Let $x_{ij} = 1$ if the tour visits city i exactly after city j and 0 otherwise.

Step 1: Choosing the variables

We already know: c_{ij}.

One idea: Let $x_j = 1$ if the tour visits city j and 0 otherwise.

A better idea: Let $x_{ij} = 1$ if the tour visits city i exactly after city j and 0 otherwise.

Alternative interpretation: Think of the map as a fully connected graph with a node for every city and an edge between every two cities. Then $x_{ij} = 1$ if and only if the edge (i, j) is being used by the tour.

Step 2: Writing the objective function

Step 2: Writing the objective function

Relatively easy: We only pay the cost for those edges that we used.

Step 2: Writing the objective function

Relatively easy: We only pay the cost for those edges that we used.

Minimise $\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$

Step 3: Writing the constraints

Step 3: Writing the constraints

This is more tricky.

Step 3: Writing the constraints

This is more tricky.

Let’s start with the easier ones.

Step 3: Writing the constraints

This is more tricky.

Let’s start with the easier ones.

Once the salesman enters a city, how many cities can he visit in the next step?

Step 3: Writing the constraints

This is more tricky.

Let's start with the easier ones.

Once the salesman enters a city, how many cities can he visit in the next step?

Only one.

Step 3: Writing the constraints

This is more tricky.

Let's start with the easier ones.

Once the salesman enters a city, how many cities can he visit in the next step?

Only one.

$$\sum_{j \in V} x_{ij} = 1, \quad \text{for } i = 0, \ldots, n-1$$

Step 3: Writing the constraints

This is more tricky.

Let's start with the easier ones.

Once the salesman enters a city, how many cities can he visit in the next step?

Only one.

$$\sum_{j \in V} x_{ij} = 1, \quad \text{for } i = 0, \dots, n-1$$

Once the salesman enters a city, from how many cities did he travel from?

Step 3: Writing the constraints

This is more tricky.

Let's start with the easier ones.

Once the salesman enters a city, how many cities can he visit in the next step?

Only one.

$$\sum_{j \in V} x_{ij} = 1, \quad \text{for } i = 0, \ldots, n-1$$

Once the salesman enters a city, from how many cities did he travel from?

Only one.

Step 3: Writing the constraints

This is more tricky.

Let's start with the easier ones.

Once the salesman enters a city, how many cities can he visit in the next step?

Only one.

$$\sum_{j \in V} x_{ij} = 1, \quad \text{for } i = 0, \ldots, n-1$$

Once the salesman enters a city, from how many cities did he travel from?

Only one.

$$\sum_{i \in V} x_{ij} = 1, \quad \text{for } j = 0, \ldots, n-1$$

Our developing ILP

Minimise $$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

subject to $$\sum_{j \in V} x_{ij} = 1, \quad \textbf{for } i = 0, \ldots, n-1$$

Our developing ILP

Minimise $$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

subject to $$\sum_{j \in V} x_{ij} = 1, \quad \textbf{for } i = 0, \ldots, n-1$$

$$\sum_{i \in V} x_{ij} = 1, \quad \textbf{for } j = 0, \ldots, n-1$$

Step 3: Writing the constraints

Step 3: Writing the constraints

Now to the more tricky ones.

Step 3: Writing the constraints

Now to the more tricky ones.

We need to make sure the cities are visited in a single tour, not in multiple disjoint subtours.

Step 3: Writing the constraints

Now to the more tricky ones.

We need to make sure the cities are visited in a single tour, not in multiple disjoint subtours.

New variables:

Step 3: Writing the constraints

Now to the more tricky ones.

We need to make sure the cities are visited in a single tour, not in multiple disjoint subtours.

New variables:

Let t_i be the number of the stop along the tour.

Step 3: Writing the constraints

Now to the more tricky ones.

We need to make sure the cities are visited in a single tour, not in multiple disjoint subtours.

New variables:

Let t_i be the number of the stop along the tour.

e.g. $t_3 = 4$ means that city 3 was visited 4th during the tour.

Relation to previous variables

Relation to previous variables

Let $x_{ij} = 1$ if the tour visits city i exactly after city j and 0 otherwise.

Relation to previous variables

Let $x_{ij} = 1$ if the tour visits city i exactly after city j and 0 otherwise.

So, when $x_{ij} = 1$, we want $t_j = t_i + 1$.

Relation to previous variables

Let $x_{ij} = 1$ if the tour visits city i exactly after city j and 0 otherwise.

So, when $x_{ij} = 1$, we want $t_j = t_i + 1$.

But at the same time, when $x_{ij} = 0$, we would like t_i to not impose any constraint on t_j.

Relation to previous variables

Let $x_{ij} = 1$ if the tour visits city i exactly after city j and 0 otherwise.

So, when $x_{ij} = 1$, we want $t_j = t_i + 1$.

Relation to previous variables

Let $x_{ij} = 1$ if the tour visits city i exactly after city j and 0 otherwise.

So, when $x_{ij} = 1$, we want $t_j = t_i + 1$.

Let's actually use $t_j \geq t_i + 1$ instead.

Relation to previous variables

Let $x_{ij} = 1$ if the tour visits city i exactly after city j and 0 otherwise.

So, when $x_{ij} = 1$, we want $t_j = t_i + 1$.

Let's actually use $t_j \geq t_i + 1$ instead.

This ensures merely that the ordering of cities in the tour is correct (j is visited after i). *If all cities are visited in single tour, we are ok*.

Relation to previous variables

Let $x_{ij} = 1$ if the tour visits city i exactly after city j and 0 otherwise.

So, when $x_{ij} = 1$, we want $t_j \geq t_i + 1$.

But at the same time, when $x_{ij} = 0$, we would like t_i to not impose any constraint on t_j.

The art of coming up with constraints

The art of coming up with constraints

What we want is the following:

The art of coming up with constraints

What we want is the following:

$t_j \geq t_i + 1$ if $x_{ij} = 1$

The art of coming up with constraints

What we want is the following:

$t_j \geq t_i + 1$ if $x_{ij} = 1$

Something at most as constraining as $t_j \geq 0$ if $x_{ij} = 0$

The art of coming up with constraints

What we want is the following:

$t_j \geq t_i + 1$ if $x_{ij} = 1$

Something at most as constraining as $t_j \geq 0$ if $x_{ij} = 0$

$\Rightarrow t_j \geq t_i + 1 - n$ if $x_{ij} = 0$

The art of coming up with constraints

What we want is the following:

$t_j \geq t_i + 1$ if $x_{ij} = 1$

Something at most as constraining as $t_j \geq 0$ if $x_{ij} = 0$

$\Rightarrow t_j \geq t_i + 1 - n$ if $x_{ij} = 0$

Putting them together: $t_j \geq t_i + 1 - n(1 - x_{ij})$

Our developing ILP

Minimise $\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$

subject to $\sum_{j \in V} x_{ij} = 1, \quad$ **for** $i = 0, \ldots, n-1$

Our developing ILP

Minimise $$\sum_{i\in V}\sum_{j\in V} c_{ij}x_{ij}$$

subject to $$\sum_{j\in V} x_{ij} = 1, \quad \textbf{for } i = 0,\ldots,n-1$$

$$\sum_{i\in V} x_{ij} = 1, \quad \textbf{for } j = 0,\ldots,n-1$$

Our developing ILP

Minimise $$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

subject to $$\sum_{j \in V} x_{ij} = 1, \quad \textbf{for } i = 0, \ldots, n-1$$

$$\sum_{i \in V} x_{ij} = 1, \quad \textbf{for } j = 0, \ldots, n-1$$

$$t_j \geq t_i + 1 - n(1 - x_{ij}) \quad \textbf{for } i \geq 0, j \geq 1, i \neq j$$

Our developing ILP

Minimise $$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

subject to $$\sum_{j \in V} x_{ij} = 1, \quad \textbf{for } i = 0, \ldots, n-1$$

$$\sum_{i \in V} x_{ij} = 1, \quad \textbf{for } j = 0, \ldots, n-1$$

$$t_j \geq t_i + 1 - n(1 - x_{ij}) \quad \textbf{for } i \geq 0, j \geq 1, i \neq j$$

$$t_0 = 0$$

Our developing ILP

Minimise $$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

subject to $$\sum_{j \in V} x_{ij} = 1, \quad \textbf{for } i = 0,\ldots,n-1$$

$$\sum_{i \in V} x_{ij} = 1, \quad \textbf{for } j = 0,\ldots,n-1$$

$$t_j \geq t_i + 1 - n(1 - x_{ij}) \quad \textbf{for } i \geq 0, j \geq 1, i \neq j$$

$$t_0 = 0$$

$$x_{ij} \in \{0,1\} \quad \textbf{for } i, j \in V$$

Our developing ILP

Minimise $$\sum_{i \in V} \sum_{j \in V} c_{ij} x_{ij}$$

subject to $$\sum_{j \in V} x_{ij} = 1, \quad \textbf{for } i = 0, \ldots, n-1$$

$$\sum_{i \in V} x_{ij} = 1, \quad \textbf{for } j = 0, \ldots, n-1$$

$$t_j \geq t_i + 1 - n(1 - x_{ij}) \quad \textbf{for } i \geq 0, j \geq 1, i \neq j$$

$$t_0 = 0$$

$$x_{ij} \in \{0,1\} \quad \textbf{for } i, j \in V$$
$$t_i \in \{0,1,2,\ldots,n-1\} \quad \textbf{for } i \in V$$

Relation to previous variables

Let $x_{ij} = 1$ if the tour visits city i exactly after city j and 0 otherwise.

So, when $x_{ij} = 1$, we want $t_j = t_i + 1$.

Let's actually use $t_j \geq t_i + 1$ instead.

This ensures merely that the ordering of cities in the tour is correct (j is visited after i). *If all cities are visited in single tour, we are ok*.

No subtours

Assume that we instead have two disjoint subtours.

Consider one of these subtours that does not include city 0, and let r be the number of cities visited by this subtour.

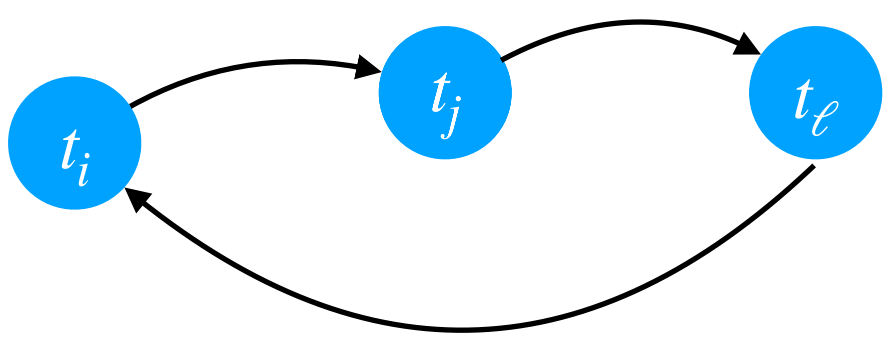
Consider the constraint $t_j \geq t_i + 1 - n(1 - x_{ij})$ and sum both sides for all the cities j in the subtour.

No subtours

Assume that we instead have two disjoint subtours.

Consider one of these subtours that does not include city 0, and let r be the number of cities visited by this subtour.

Consider the constraint $t_j \geq t_i + 1 - n(1 - x_{ij})$ and sum both sides for all the cities j in the subtour.

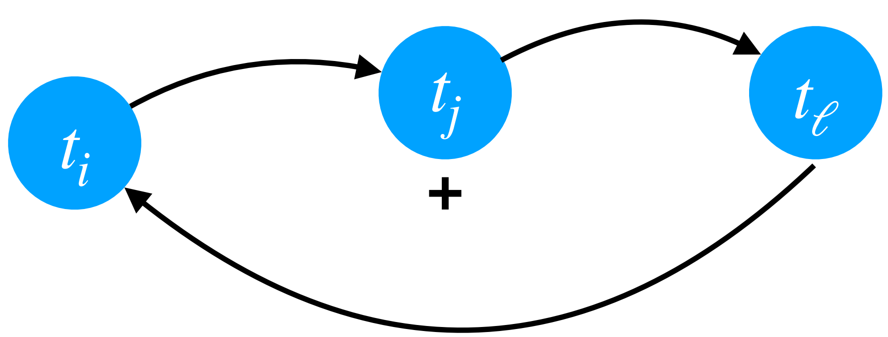


No subtours

Assume that we instead have two disjoint subtours.

Consider one of these subtours that does not include city 0, and let r be the number of cities visited by this subtour.

Consider the constraint $t_j \geq t_i + 1 - n(1 - x_{ij})$ and sum both sides for all the cities j in the subtour.

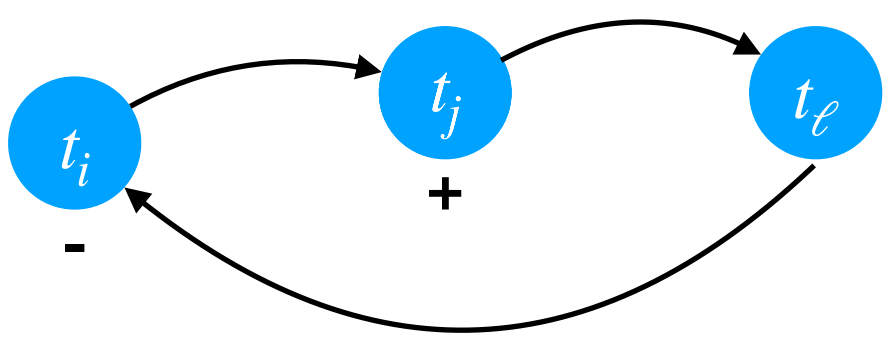


No subtours

Assume that we instead have two disjoint subtours.

Consider one of these subtours that does not include city 0, and let r be the number of cities visited by this subtour.

Consider the constraint $t_j \geq t_i + 1 - n(1 - x_{ij})$ and sum both sides for all the cities j in the subtour.

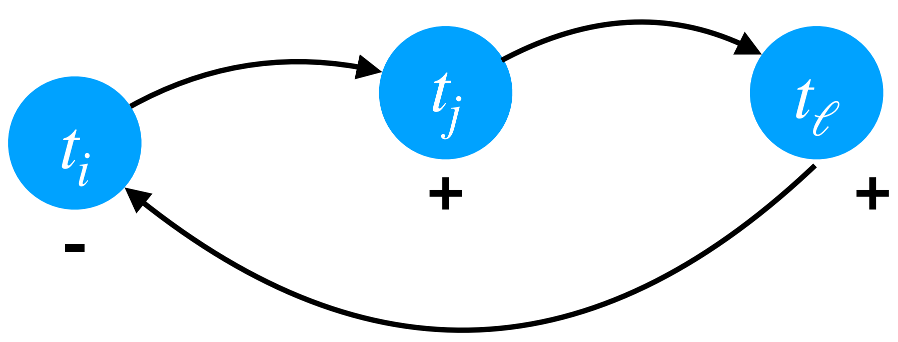


No subtours

Assume that we instead have two disjoint subtours.

Consider one of these subtours that does not include city 0, and let r be the number of cities visited by this subtour.

Consider the constraint $t_j \geq t_i + 1 - n(1 - x_{ij})$ and sum both sides for all the cities j in the subtour.

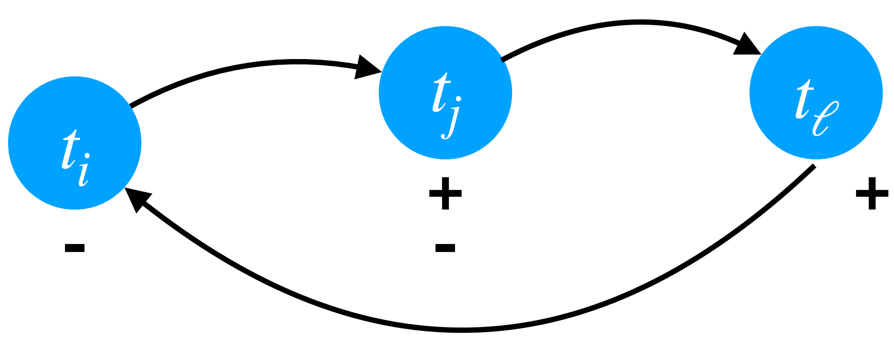


No subtours

Assume that we instead have two disjoint subtours.

Consider one of these subtours that does not include city 0, and let r be the number of cities visited by this subtour.

Consider the constraint $t_j \geq t_i + 1 - n(1 - x_{ij})$ and sum both sides for all the cities j in the subtour.

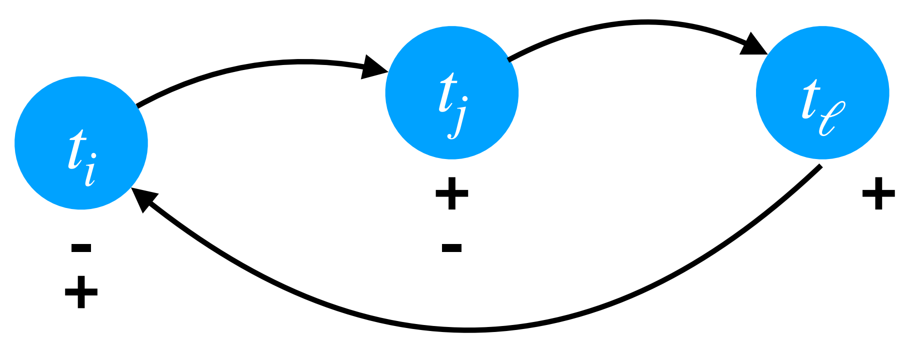


No subtours

Assume that we instead have two disjoint subtours.

Consider one of these subtours that does not include city 0, and let r be the number of cities visited by this subtour.

Consider the constraint $t_j \geq t_i + 1 - n(1 - x_{ij})$ and sum both sides for all the cities j in the subtour.

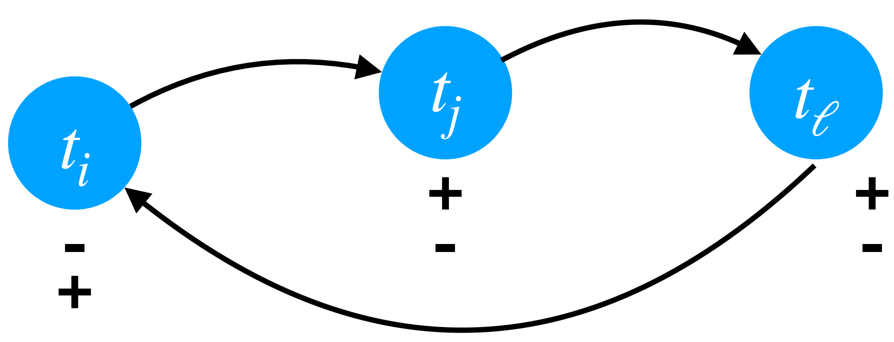


No subtours

Assume that we instead have two disjoint subtours.

Consider one of these subtours that does not include city 0, and let r be the number of cities visited by this subtour.

Consider the constraint $t_j \geq t_i + 1 - n(1 - x_{ij})$ and sum both sides for all the cities j in the subtour.

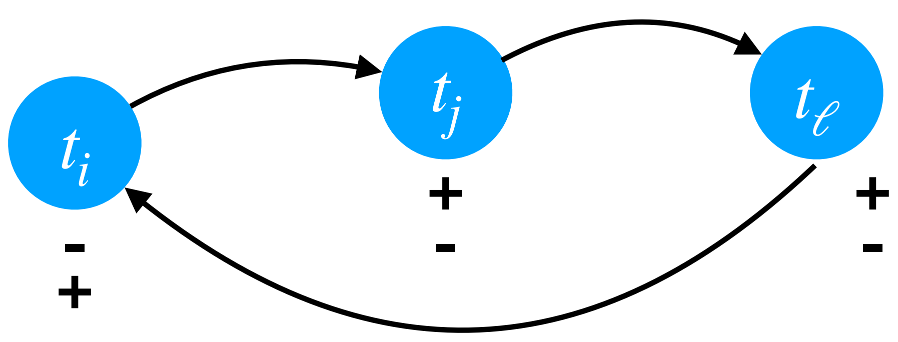


No subtours

Assume that we instead have two disjoint subtours.

Consider one of these subtours that does not include city 0, and let r be the number of cities visited by this subtour.

Consider the constraint $t_j \geq t_i + 1 - n(1 - x_{ij})$ and sum both sides for all the cities j in the subtour.



LHS = X

RHS = $X + r$

contradiction