

Algorithms and Data Structures

Modelling with Linear Programs

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

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

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

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

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

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 $1, \dots, n$.

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

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_j

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 $1, \dots, n$.

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

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_j

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_j

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 ? $c_j = \sigma_j - \sum_{i=1}^m \alpha_{ij} \cdot \rho_i$

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_j

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 ? $c_j = \sigma_j - \sum_{i=1}^m \alpha_{ij} \cdot \rho_i$

What is the revenue from all the units 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_j

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 ? $c_j = \sigma_j - \sum_{i=1}^m \alpha_{ij} \cdot \rho_i$

What is the revenue from all the units 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_j

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 ? $c_j = \sigma_j - \sum_{i=1}^m \alpha_{ij} \cdot \rho_i$

What is the revenue from all the units 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_j

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 ? $c_j = \sigma_j - \sum_{i=1}^m \alpha_{ij} \cdot \rho_i$

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

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

Our LP formulation

$$\begin{array}{ll} \text{Maximise} & \sum_{i=1}^n c_j x_j \\ \text{subject to} & ? \end{array}$$

Step 3: Writing the constraints

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

1. We cannot produce negative amounts:

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

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 $1, \dots, n$.

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

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:

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

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

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

Our LP formulation

$$\begin{array}{ll}\text{Maximise} & \sum_{j=1}^n c_j x_j \\ \text{subject to} & \sum_{j=1}^n \alpha_{ij} \cdot x_j \leq b_i \quad \text{for all } i = 1, \dots, m \\ & x_j \geq 0 \quad \text{for all } j = 1, \dots, n\end{array}$$

Integer Linear programming

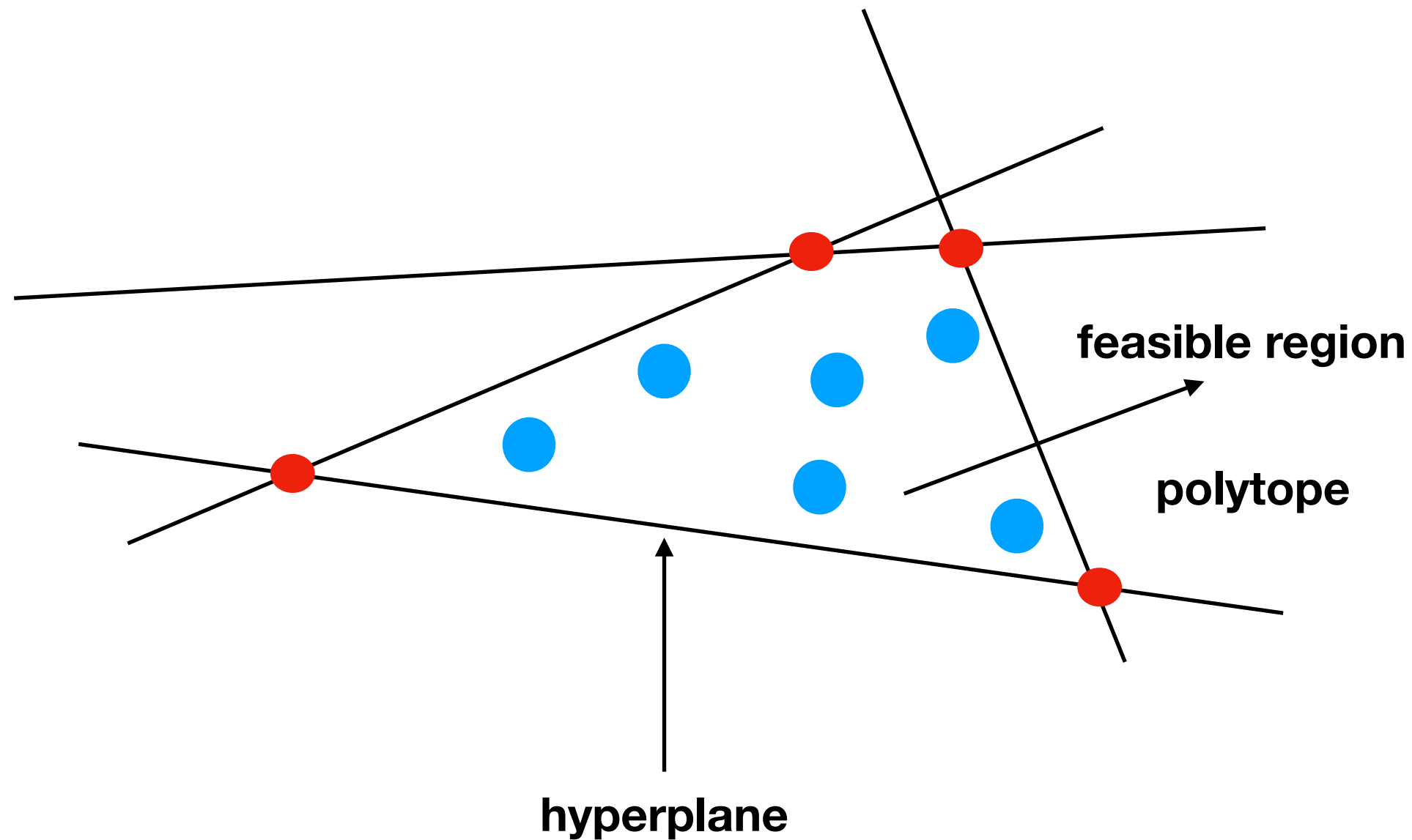
$$\text{maximise} \quad \sum_{j=1}^n c_j x_j$$

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

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

x_j is integer

Feasible region



● candidate optimal solution

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

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 $1, \dots, n$ 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

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

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 ILP formulation

Minimise $\sum_{j=1}^n c_j x_j$
subject to ?

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?

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

How many are these?

One!

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

Our ILP formulation

Anything else?

$$\begin{array}{ll} \text{Minimise} & \sum_{j=1}^n c_j x_j \\ \text{subject to} & \sum_{j=1}^n a_{ij} x_j \quad \text{for } i = 1, \dots, m \\ & x_j \in \{0, 1\} \quad \text{for } j = 1, \dots, n \end{array}$$

Unrelated Machine Scheduling

We have n jobs to be scheduled on m machines.

Each job i has a processing time t_{ij} on machine j .

Each machine processes one job after another, but different machines run in parallel.

We would like to find a way to assign the jobs to the machines such that we minimise the *makespan*, i.e., the completion time of the last machine to finish.

Step 1: Choosing the variables

We already know: t_{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 $x_{ij} = 1$ if we assign job i to machine j and $x_{ij} = 0$ otherwise.

Step 2: Writing the objective function

We want to minimise the makespan.

The processing time of a machine is the total processing time of jobs assigned to it,

i.e., $\sum_{i=1, \dots, n} t_{ij} x_{ij}$

so we want to minimise the maximum processing time,

i.e., our objective function is $\min_{j=1, \dots, m} \max \sum_{i=1, \dots, n} t_{ij} x_{ij}$ any issues?
not linear!

Our ILP formulation

Minimise $\max_{j=1,\dots,m} \sum_{i=1,\dots,n} t_{ij} x_{ij}$

subject to ?

(Let's put that aside for a second...)

Step 3: Writing the constraints

Constraint 1: Every job is assigned to exactly one machine,

i.e.,
$$\sum_{j=1, \dots, m} x_{ij} = 1 \quad \text{for every } i = 1, \dots, n$$

Constraint 2: The indicator variables correspond to an assignment of the jobs,

i.e.,
$$x_{ij} \in \{0, 1\} \quad \text{for every } i, j$$

Our ILP formulation

call this T

Minimise

$$\max_{j=1,\dots,m} \sum_{i=1,\dots,n} t_{ij} x_{ij}$$

subject to $\sum_{i=1,\dots,m} x_{ij} = 1$ for every $j = 1,\dots,m$

$$x_{ij} \in \{0,1\} \text{ for every } i,j$$

(Let's put that aside for a second...)

Our ILP formulation

Minimise T

subject to $\sum_{i=1,\dots,m} x_{ij} = 1$ for every $j = 1,\dots,m$

$x_{ij} \in \{0,1\}$ for every i,j

What is the relationship between $\sum_{i=1,\dots,n} t_{ij}x_{ij}$ and T for any j ?

Our ILP formulation

Minimise T

subject to $\sum_{i=1,\dots,m} x_{ij} = 1$ for every $j = 1,\dots,m$

$x_{ij} \in \{0,1\}$ for every i,j

$\sum_{i=1,\dots,n} t_{ij}x_{ij} \leq T$ for any j

Our ILP formulation

Minimise T

subject to $\sum_{i=1,\dots,m} x_{ij} = 1$ for every $j = 1,\dots,m$

$x_{ij} \in \{0,1\}$ for every i,j

$\sum_{i=1,\dots,n} t_{ij}x_{ij} \leq T$ for any j

The Travelling Salesman Problem

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

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

A tour can be described as a sequence of cities

$0, s_1, s_2, \dots, s_{n-1}$.

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

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

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

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?

Only one.

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

Our developing ILP

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

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

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

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

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 $t_j = t_i + 1$.

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 a 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

What we want is the following:

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

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

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

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

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

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

$$t_0 = 0$$

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

$$t_i \in \{0, 1, \dots, n-1\} \quad \text{for } i \in V$$

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.

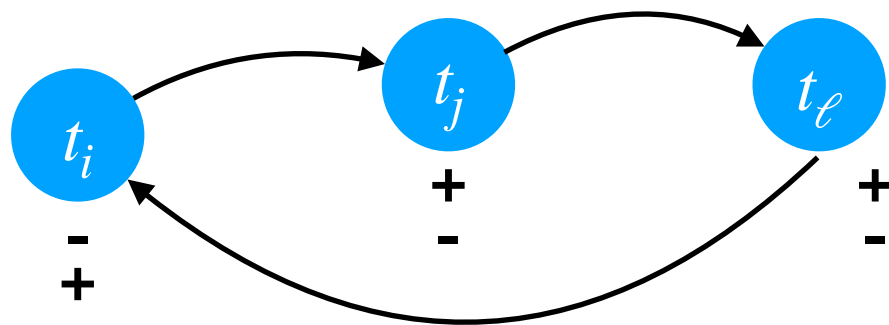
This ensures merely that the ordering of cities in the tour is correct (j is visited after i). *If all cities are visited in a single tour, we are ok.*

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.



LHS = X
RHS = $X + r$
contradiction