

# 'MATH+ECON+CODE' MASTERCLASS ON MATCHING MODELS, OPTIMAL TRANSPORT AND APPLICATIONS

Alfred Galichon (New York University)

Spring 2018

Day 1, January 15 2018: Linear programming  
Block 2. The min-cost flow problem

- ▶ Basic concepts of directed networks
- ▶ The min-cost flow problem
- ▶ Duality, optimality and equilibrium
- ▶ The shortest path problem

- ▶ [OTME], Ch.8
- ▶ Tolstoi (1930). Methods of finding the minimal total kilometrage in cargo transportation planning in space. *Transportation Planning* [in Russian]
- ▶ Koopmans (1949). Optimum utilization of the transportation system. *Econometrica*.
- ▶ Schrijver (2002). On the History of the Transportation and Maximum Flow Problems. *Mathematical programming*.

# Section 1

## THEORY

- ▶ In 1930 Tolstoř, a Russian engineer, has to optimize the shipping of cement from factories to cities in the Soviet union through railway.
  - ▶ each factory produces a fixed number of tons
  - ▶ each city needs a fixed number of tons – for now, we'll assume total production = total consumption
  - ▶ each factory is connected by rail with a few cities, and the corresponding distance is given
  - ▶ how to ship in order to minimize the total distance travelled?
- ▶ This problem belongs to the class of *min-cost flow problems*, an important class of linear programming problems, which are the focus of today's lecture. A decade before the invention of linear programming and the work of Kantorovich, Koopmans and Dantzig, Tolstoř described a heuristic method for solving the problem, which led to the optimal solution. We'll solve the problem using modern tools (Gurobi), and will see that his solution was right.
- ▶ The *shortest path problem*, or how to find the path of minimal distance from a point to another in a network, also belongs in this class. Later in this lecture, you will determine the shortest path between NYU and your place of residence through the NYC subway.

- ▶ We will look at three types of data,
  - ▶ Tolstoï's data, collected by Schrijver (2002), which is in the directory '02-appli-networks/sovietplanning'. There are 10 factories and 68 Soviet cities, and 155 links connecting a factory to a city. This yields a sparse  $68 \times 10$  matrix. Two vectors listing the demand of each city and the supply of each factory are also specified. This is stored in a  $69 \times 11$  matrix, where we have appended the demand/supply vectors to the right and to the bottom of the distance matrix.
  - ▶ The data on the NYC subway, available from the mta website, which is in the directory '02-appli-networks/subway'. This data is made of two files. The file 'nodes.csv' lists the stations: each station is indexed by the line number; each line has the name of the station, and its spatial coordinates. This is a  $472 \times 3$  array. The file 'arcs.csv' lists the links between stations: each link specifies the index of the origin, the index of the destination, and the length of the segment. This is a  $1230 \times 3$  array.
  - ▶ The data of the NYC roads ('02-appli-networks/NYCrads'), which is available as a 'shapefile' but can be essentially described as the NYC subway above – except that the network is much larger, as there are 67,316 nodes and 169,399 arcs.

- We start by defining the directed graph on which transportation takes place.

## DEFINITION

A (directed) *graph*  $(\mathcal{X}, \mathcal{A})$  is a set of *nodes* (cities)  $\mathcal{X}$ , along with a set of *arcs*  $\mathcal{A} \subseteq \mathcal{X}^2$  which are pairs  $(x, y)$  where  $x, y \in \mathcal{X}$ .

Our definition allows an arc to have the same origin and destination. Note that for a dense network,  $\mathcal{A} = \mathcal{X}^2$ . For a line,  $|\mathcal{A}| = |\mathcal{X}| - 1$ .

- Next, we define the gradient matrix.

## DEFINITION

We define the *gradient matrix* (also called an ‘edge-node matrix’) as the matrix with general term  $\nabla_{ax}$ ,  $a \in \mathcal{A}$ ,  $x \in \mathcal{X}$ , such that

$$\begin{aligned}\nabla_{ax} &= -1 \text{ if } a \text{ is out of } x, \\ &= +1 \text{ if } a \text{ is into } x, \\ &= 0 \text{ else.}\end{aligned}$$

Hence, if  $f \in \mathbb{R}^{\mathcal{X}}$ , then  $\nabla f \in \mathbb{R}^{\mathcal{V}}$ , and  $(\nabla f)_{xy} = f_y - f_x$ .

We shall denote  $\nabla^T$  the transpose of the gradient matrix. It is the network analog of the  $-\operatorname{div}$  differential operator.



- Next, we define paths and loops

### DEFINITION

Given two nodes  $x$  and  $y$ , a *path* from  $x$  to  $y$  is a sequence  $x_1, x_2, \dots, x_K$  in  $\mathcal{X}$  where  $x_1 = x$ ,  $x_K = y$ , and for every  $1 \leq k \leq K - 1$ ,  $(x_k, x_{k+1}) \in \mathcal{A}$ .  
A *loop* (also called 'cycle') is a path from a node  $x$  to itself.

- ▶ A vector  $c \in \mathbb{R}^{\mathcal{V}}$  defined transportation costs. That is, for  $xy \in \mathcal{V}$ ,  $c_{xy}$  is the transportation cost associated to arc  $xy$ .  $c$  can also be thought of as the length of arc  $xy$ . The cost of moving the good from node  $x$  to node  $y$  along path  $x_1, x_2, \dots, x_K$  is

$$\sum_{k=1}^{K-1} c_{x_k x_{k+1}}.$$

- ▶ No arbitrage conditions: there is no loop of negative cost:

## ASSUMPTION

*There is no profitable loop, which means that there is no sequence  $x_1, \dots, x_K$  in  $\mathcal{X}$  such that  $x_K = x_1$ ,  $(x_k, x_{k+1}) \in \mathcal{A}$ , and  $\sum_{k=1}^{K-1} c_{x_k x_{k+1}} < 0$ .*

In particular, there is no profitable loop if  $c \geq 0$ .

- Let  $n_x$  be the *net demand*, which is the flow of goods disappearing from the graph. The set of nodes defined by

$$\mathcal{X}_0 = \{x \in \mathcal{X} : n_x < 0\}, \text{ and } \mathcal{X}_1 = \{x \in \mathcal{X} : n_x > 0\}$$

are called the *supply nodes* and *demand nodes* respectively. A node which is neither a supply node, neither a demand node is called a *transit node*.

- Total supply is  $-\sum_{x \in \mathcal{X}_0} n_x$ , total demand is  $\sum_{y \in \mathcal{X}_1} n_y$ .

### ASSUMPTION (BALANCEDNESS)

*Assume that total supply equals total demand on the network, that is*

$$\sum_{x \in \mathcal{X}_0} n_x + \sum_{y \in \mathcal{X}_1} n_y = 0.$$

### ASSUMPTION (CONNECTEDNESS)

*Assume the set of supply nodes  $\mathcal{X}_0$  is strongly connected to the set of demand nodes  $\mathcal{X}_1$ , i.e. for every  $x \in \mathcal{X}_0$  and  $y \in \mathcal{X}_1$ , there is a path from  $x$  to  $y$ .*

The specification of the graph, the net demand vector, and the surplus vector defines a network.

### DEFINITION

A directed graph  $(\mathcal{X}, \mathcal{A})$  endowed with a net demand vector  $(n_z)_{z \in \mathcal{X}}$  and a cost vector  $(c_a)_{a \in \mathcal{A}}$  is called a *network*  $(\mathcal{X}, \mathcal{A}, n, c)$ . If Assumptions 1 (No profitable loop), 2 (total supply equals total demand) and 3 (supply is strongly connected to demand) all hold, the network is called *regular*.

Without mention of the contrary we shall assume that the network under consideration is regular.

The flow of mass disappearing at  $x$  equals the flow arriving from other nodes minus the flow shipping to other nodes

$$n_x = \sum_{z:(z,x) \in \mathcal{A}} \pi_{zx} - \sum_{z:(x,z) \in \mathcal{A}} \pi_{xz}$$

and this equation can be rewritten as  $\nabla^T \pi = n$ . This motivates the following definition:

## DEFINITION

The set of feasible flows, denoted  $\mathcal{M}(n)$ , or  $\mathcal{M}$  when there is no ambiguity, is defined as the set of flows  $\pi \geq 0$  that verify conservation equation

$$\nabla^T \pi = n. \tag{1}$$

Let  $\phi_x$  be the price of the commodity at  $x$ . Consider the strategy which consists in purchasing the good at  $x$ , shipping to  $y$ , and selling at  $y$ . The profit of this strategy is

$$\phi_y - \phi_x - c_{xy} = (\nabla\phi - c)_{xy}$$

and hence there is no arbitrage opportunity if  $\phi_y - \phi_x - c_{xy} \leq 0$  for every arc  $xy$ , that is

$$\nabla\phi \leq c.$$

Consider the *minimum cost flow problem*

$$\begin{aligned} \min_{\pi \geq 0} \quad & \sum_{(x,y) \in \mathcal{A}} \pi_{xy} c_{xy} \\ \text{s.t.} \quad & \nabla^T \pi = n \end{aligned} \tag{2}$$

which is a Linear Programming problem whose dual is

$$\begin{aligned} \max_{\phi \in \mathbb{R}^{\mathcal{X}}} \quad & \sum_{x \in \mathcal{X}} n_x \phi_x \\ \text{s.t.} \quad & \nabla \phi \leq c. \end{aligned} \tag{3}$$



## PROPOSITION

- (i) Under Assumption 1 (No profitable loop), the dual problem (3) is feasible, which means that there is a vector  $\phi \in \mathbb{R}^{\mathcal{X}}$  such that  $\nabla \phi \leq c$ ; and the value of Problem (2) is strictly less than  $+\infty$ .
- (ii) Under Assumptions 2 (total supply equals total demand) and 3 (Supply is strongly connected to demand), the primal problem (2) is feasible, which means that there is a flow  $\pi \geq 0$  such that  $\nabla^T \pi = n$ ; and the value of Problem (3) is strictly greater than  $-\infty$ .

Assume that  $(\mathcal{X}, \mathcal{A}, n, c)$  is a regular network. Then the value of the primal problem (2) coincides with the value of its dual (3), and both problems have solutions. Further, if  $\pi$  is a solution to the primal and  $\phi$  is a solution to the dual, then  $\pi_{xy} > 0$  implies  $\phi_y - \phi_x = c_{xy}$ .

- Assume there is only one source node  $s \in \mathcal{X}$  and one target node  $t \in \mathcal{X}$ , each associated with unit flow. That is,  $n_x = 1 \{x = t\} - 1 \{x = s\}$ . Then the problem boils down to how to push one unit of mass from  $s$  to  $t$ . If we interpret  $c_{xy}$  as the distance along arc  $xy$ , the solution of this problem corresponds to the shortest path from  $s$  to  $t$ . This is why this problem is called shortest path problem. (More generally, this problem extends to the case when  $c$  does not have a negative loop).
- The dual problem is then

$$\begin{aligned} \max_{\phi} \quad & \phi_t - \phi_s \\ \text{s.t.} \quad & \phi_y - \phi_x \leq c_{xy} \quad \forall xy \in \mathcal{A} \end{aligned}$$

and we can impose normalization  $\phi_s = 0$ , so that along the travelled path,  $\phi_x$  interprets as the distance travelled thus far.

- ▶ Assume the problem is bipartite, that is  $\mathcal{X} = \mathcal{S} \cup \mathcal{T}$  and  $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{T}$ . That is, there are no intermediate nodes, and an arc can only go directly from a source to a target.
- ▶ Note that any min-cost flow problem can be recast in this form, by taking the shortest distance between any source node and any target node.
- ▶ The dual problem is then

$$\begin{aligned} \max_{\phi} \quad & \sum_{x \in \mathcal{X}} n_x \phi_x \\ \text{s.t.} \quad & \phi_t - \phi_s \leq c_{st} \quad \forall s \in \mathcal{S}, t \in \mathcal{T} \end{aligned}$$

which is our first encounter with optimal transport (more on this tomorrow).

## Section 2

## CODING

- Consider a network with  $\text{nbNodes}$  nodes and  $\text{nbArcs}$  arcs. The typical structure of the data needed is:
  - (i)  $\text{arcs}$ , an array of integers of size  $\text{nbArcs} \times 2$  whose entries are between 1 and  $\text{nbNodes}$ , so that each line describes an arc, represented by its origin node (first column) and its destination node (second column);
  - (ii)  $\mathbf{n}$ , a vector of length  $\text{nbNodes}$  such that  $n_x$  is the net demand at node  $x$ ;
  - (iii)  $\mathbf{c}$ , a vector of length  $\text{nbArcs}$  such that  $c_a$  is the transportation cost at arc  $a$ .

- ▶ The node-incidence matrix is coded in a sparse way. In R, this is done using the package 'Matrix', and the relevant instruction is:  
Nabla =  
`sparseMatrix(i=1:nbArcs,j=arcs[,1],dims=c(nbArcs,nbNodes),x=-1)  
+sparseMatrix(i=1:nbArcs,j=arcs[,2],dims=c(nbArcs,nbNodes),x=1).`
- ▶ We will use Gurobi for LP. The solver is called by:  
sol =  
`gurobi(list(A=t(Nabla),obj=c,modelsense='min',rhs=n,sense=''))`

- ▶ For plotting a graph, one needs the package 'igraph'.
- ▶ Create the graph using:  
`thegraph=graph_from_edgelist(arcs[,1:2])`
- ▶ Plot the graph using:  
`plot.igraph(thegraph, layout = geoCoordinates)`