Transportation network with multiple lines

notes GG

April 25, 2022

1 Formalism

1.1 The transportation network with multiple lines

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a simple, oriented, and connected graph representing a transportation network between $|\mathcal{V}| = n$ nodes, having $|\mathcal{E}| = m$ edges, and possessing p different transportation lines. Each node belongs to only one line, i.e. $\mathcal{V} = \bigcup_{k=1}^{p} \mathcal{V}_k$ and $\mathcal{V}_k \cap \mathcal{V}_l = \emptyset$, $\forall k \neq l$, where $\mathcal{V}_k$ represents the set of nodes in line k. The edge set $\mathcal{E}$, can also be decomposed with

$$\mathcal{E} = \mathcal{E}_\mathrm{W} \cup \mathcal{E}_\mathrm{B}, \qquad \mathcal{E}_\mathrm{W} := \bigcup_{k=1}^{p} \mathcal{E}_k, \qquad \mathcal{E}_\mathrm{W} \cap \mathcal{E}_\mathrm{B} = \emptyset, \quad \mathcal{E}_k \cap \mathcal{E}_l = \emptyset, \quad \forall k \neq l. \tag{1}$$

where $\mathcal{E}_k$ is the set of edges composing line k, $\mathcal{E}_\mathrm{W}$ the set containing all edges inside lines, and $\mathcal{E}_\mathrm{B}$ the set of transfer edges, connecting the different lines.
The graph $\mathcal{G}$ can be represented by its adjacency matrix $\mathbf{A} = (a_{ij})$, which can also be decomposed with

$$\mathbf{A} = \mathbf{A}_\mathrm{W} + \mathbf{A}_\mathrm{B}, \qquad \mathbf{A}_\mathrm{W} = \sum_{k=1}^{p} \mathbf{A}_k, \tag{2}$$

with $\mathbf{A}_k = (a_{ij}^k)$ are edges of line k, $\mathbf{A}_\mathrm{W} = (a_{ij}^\mathrm{W})$ edges of inside all lines, and $\mathbf{A}_\mathrm{B} = (a_{ij}^\mathrm{B})$ transfer edges. We suppose that there is an uniquely define route inside lines, i.e.

$$a_{i\bullet}^k \leq 1 \text{ and } a_{\bullet i}^k \leq 1, \qquad \forall i, k. \tag{3}$$

where $\bullet$ designates a summation over the replaced index.

1.2 The origin-destination matrix

The $(n \times n)$ *origin-destination matrix*, denoted by $\mathbf{N} = (n_{st})$, $n_{st} \geq 0$, $\forall s, t$, contains the flow (e.g. the number of passengers) entering the network in source node s and leaving it in target node t. We can denote its margins with

$$\boldsymbol{\sigma}_\mathrm{in} := \mathbf{N}\mathbf{e}_n \tag{4}$$

$$\boldsymbol{\sigma}_\mathrm{out} := \mathbf{N}^\top \mathbf{e}_n \tag{5}$$

where $\mathbf{e}_n$ is the vector of ones of size n. The vector $\boldsymbol{\sigma}_{\text{in}} = (\sigma_i^{\text{in}})$ is the *vector of flow entering the network* and $\boldsymbol{\sigma}_{\text{in}} = (\sigma_i^{\text{in}})$ is the *vector of flow leaving the network*. We have

$$\sigma_\bullet^{\text{in}} = \sigma_\bullet^{\text{out}}. \tag{6}$$

Note that if only $\boldsymbol{\sigma}_{\text{in}}$ and $\boldsymbol{\sigma}_{\text{out}}$ are given, a flow matrix $\mathbf{N}$ can be computed relatively to an *origin-destination affinity matrix* $\mathbf{S} = (s_{st})$, $0 \leq s_{st} \leq 1$, where $s_{st} = 1$ denote a perfect affinity and $s_{st} = 0$ no affinity, through

$$\mathbf{N} = \mathbf{Diag}(\mathbf{a})(\mathbf{S} + \epsilon)\mathbf{Diag}(\mathbf{b}), \tag{7}$$

where $\mathbf{Diag}(.)$ denote the diagonal matrix obtained from a vector, ϵ a very small quantity, and vectors $\mathbf{a}$ and $\mathbf{b}$ are found through *proportional iterative fitting* algorithm in order to have margin constraints (4) and (5) respected for $\mathbf{N}$ (a small ϵ has to be added to $\boldsymbol{\sigma}_{\text{in}}$ and $\boldsymbol{\sigma}_{\text{out}}$ if they possess null components).

1.3 The flow matrix

A flow on edges is represented by the $(n \times n)$ *flow matrix* $\mathbf{X} = (x_{ij})$, verifying

$$x_{ij} \geq 0, \qquad \forall i, j, \tag{8}$$

$$a_{ij} = 0 \Rightarrow x_{ij} = 0, \qquad \forall i, j, \tag{9}$$

$$x_{i\bullet} + \sigma_i^{\text{out}} = x_{\bullet i} + \sigma_i^{\text{in}}, \qquad \forall i. \tag{10}$$

Again, we can decompose the flow matrix with

$$\mathbf{X} = \mathbf{X}_{\text{W}} + \mathbf{X}_{\text{B}} \qquad \mathbf{X}_{\text{W}} := \sum_{k=1}^{p} \mathbf{X}_k \tag{11}$$

where $\mathbf{X}_k$ represent the flow inside line k, $\mathbf{X}_{\text{W}}$ is the flow inside all lines, and $\mathbf{X}_{\text{B}}$ the flow between lines. This decomposition allows us to define the *vector of flow entering lines* $\boldsymbol{\rho}_{\text{in}} = (\rho_i^{\text{in}})$ and the *vector of flow leaving lines* $\boldsymbol{\rho}_{\text{out}} = (\rho_i^{\text{out}})$, with

$$\boldsymbol{\rho}_{\text{in}} := \boldsymbol{\sigma}_{\text{in}} + \mathbf{X}_{\text{B}}^\top \mathbf{e}_n, \tag{12}$$

$$\boldsymbol{\rho}_{\text{out}} := \boldsymbol{\sigma}_{\text{out}} + \mathbf{X}_{\text{B}} \mathbf{e}_n, \tag{13}$$

where $\mathbf{e}_n$ is the vector of ones of size n. It is easy to see that we still have $\rho_\bullet^{\text{in}} = \rho_\bullet^{\text{out}}$.

1.4 Shortest-paths flow

Let $\mathcal{P}_{st}$ be the set of *admissible* shortest-paths between s and t on $\mathcal{G}$. We can denote by $\mathrm{P}_{st}(i,j)$ the probability of having edge $(i,j) \in \wp$ when drawing a path $\wp$ from $\mathcal{P}_{st}$. We have

$$\mathrm{P}_{st}(i,j) := \frac{1}{|\mathcal{P}_{st}|} \sum_{\wp \in \mathcal{P}_{st}} \delta((i,j) \in \wp), \tag{14}$$

where $\delta(.)$ designate the indicator function. Note that if there is an unique shortest-path between node s and t, noted $\wp_{st}$, we have $\mathrm{P}_{st}(i,j) = 1$ if $(i,j) \in \wp_{st}$, $\mathrm{P}_{st}(i,j) = 0$ otherwise.

If we are given an origin-destination matrix $\mathbf{N} = (n_{st})$, we can compute the *shortest-path flow matrix*, noted $\mathbf{X}_{\mathrm{sp}} = (x_{ij}^{\mathrm{sp}})$, with

$$x_{ij}^{\mathrm{sp}} = \sum_{st} \mathrm{P}_{st}(i,j) n_{st}. \tag{15}$$

This matrix contains the flow on each edge if we suppose that the flow follows shortest-paths from origin to destination.

We can rewrite equation (15) by defining the $(n^2 \times n^2)$ *shortest-path - edge matrix* $\mathbf{P} = (p_{\alpha\beta})$ with

$$p_{\alpha\beta} = \begin{cases} \mathrm{P}_{st}(i,j) & \text{if } \alpha = s + n(t-1) \text{ and } \beta = i + n(j-1), \\ 0 & \text{otherwise.} \end{cases} \tag{16}$$

Then (15) writes

$$\mathbf{vec}(\mathbf{X}_{\mathrm{sp}}) = \mathbf{P}^{\top} \mathbf{vec}(\mathbf{N}), \tag{17}$$

where $\mathbf{vec}(.)$ denotes the vectorization function of a matrix, obtained by stacking matrix columns on top of one another.

From equation (15), we see that

$$\frac{\partial x_{ij}^{\mathrm{sp}}}{\partial n_{st}} = \mathrm{P}_{st}(i,j), \tag{18}$$

which equal to 1 if there is a unique shortest-path between s and t and (i,j) belongs to this path. This equation means that if we multiply n_{st} by a factor $\alpha \geq 0$ on each s,t which contains (i,j) on their shortest-paths, the resulting flow on (i,j) will also be multiplied by α.

1.5 Problem definition

The problem: We suppose that we know the flow entering and leaving each line, i.e. $\boldsymbol{\rho}_{\mathrm{in}}$ and $\boldsymbol{\rho}_{\mathrm{out}}$, and we want to find origin-destination trajectories n_{st}.

By setting the problem like that, we easily see that it is ill posed. Several solutions exists, with some of them trival (e.g. units remain on the same line and follow a first-in/first-out scheme), and we need to add some hypotheses to restrain it.

Hypothesis 1: Trajectories in the network follow shortest-paths from origin s to destination t.

Hypothesis 2: The number of trajectories $\mathbf{N} = (n_{st})$ should be as close as possible to s_{st}, where $\mathbf{S} = (s_{st})$ is a given affinity matrix between origin and destination nodes, in the sens that

$$K(\mathbf{N}|\mathbf{S}) := \sum_{st} \frac{n_{st}}{n_{\bullet\bullet}} \log\left(\frac{n_{st}/n_{\bullet\bullet}}{s_{st}/s_{\bullet\bullet}}\right), \tag{19}$$

i.e. the *Kullback-Leibler divergence* between the probability of selecting an origin-destination path according to $\mathbf{N}$ relatively to the probabilty of selecting an origin-destination path according to $\mathbf{S}$, is minimum. Note that the divergence (19) is well defined only if $s_{st} > 0$, $\forall s, t$.

With these two additional hypotheses, we can find a solution with the following algorithm.

1.6 Algorithm

Set $\boldsymbol{\sigma}_{\text{in}}^{(1)} = \boldsymbol{\rho}_{\text{in}}$, $\boldsymbol{\sigma}_{\text{out}}^{(1)} = \boldsymbol{\rho}_{\text{out}}$, and $\mathbf{S}^{(1)} = \mathbf{S}$. Until convergence, do:

1. Compute
$$\mathbf{N}^{(\tau)} = \mathbf{Diag}(\mathbf{a}^{(\tau)})(\mathbf{S}^{(\tau)} + \epsilon)\mathbf{Diag}(\mathbf{b}^{(\tau)}), \tag{20}$$
with proportional iterative fitting, such that $\mathbf{N}^{(\tau)}\mathbf{e}_n = \boldsymbol{\sigma}_{\text{in}}^{(\tau)} + \epsilon$ and $\left(\mathbf{N}^{(\tau)}\right)^\top \mathbf{e}_n = \boldsymbol{\sigma}_{\text{out}}^{(\tau)} + \epsilon$.

2. Compute the associated shortest-path flow matrix $\mathbf{X}^{(\tau)}$ with
$$\mathbf{vec}(\mathbf{X}^{(\tau)}) = \mathbf{P}^\top \mathbf{vec}(\mathbf{N}^{(\tau)}). \tag{21}$$

3. Compute the vectors of *between-lines flow entering and leaving each nodes*, i.e.
$$\mathbf{x}_{\text{B,in}}^{(\tau)} = (\mathbf{X}_{\text{B}}^{(\tau)})^\top \mathbf{e}_n \tag{22}$$
$$\mathbf{x}_{\text{B,out}}^{(\tau)} = \mathbf{X}_{\text{B}}^{(\tau)} \mathbf{e}_n \tag{23}$$

4. Compute the vectors of *between-line allowed flow entering and leaving each nodes*, written resp. $\tilde{\mathbf{x}}_{\text{B,in}}^{(\tau)} = (\tilde{x}_i^{\text{B,in},(\tau)})$ and $\tilde{\mathbf{x}}_{\text{B,out}}^{(\tau)} = (\tilde{x}_i^{\text{B,out},(\tau)})$, with
$$\tilde{x}_i^{\text{B,in},(\tau)} = \rho_i^{\text{in}} \phi\left(\frac{x_i^{\text{B,in},(\tau)}}{\rho_i^{\text{in}}}\right), \tag{24}$$
$$\tilde{x}_i^{\text{B,out},(\tau)} = \rho_i^{\text{out}} \phi\left(\frac{x_i^{\text{B,out},(\tau)}}{\rho_i^{\text{out}}}\right), \tag{25}$$
where $\phi(x)$ is a positive increasing function which should be the identity when $x \to 0$ and with $\phi(x) \leq 1$, $\forall x$. For example

(a) $\phi(x) = \min(x, 1)$,

(b) $\phi(x) = \min(x, 1 - \exp(-\lambda x))$, with $\lambda > 0$ a parameter.

5. Compute the *between-lines allowed flow on edges*, noted $\widetilde{\mathbf{X}}_{\mathrm{B}}^{(\tau)} = (\widetilde{x}_{ij}^{\mathrm{B},(\tau)})$ with

$$\widetilde{x}_{ij}^{\mathrm{B},(\tau)} = \begin{cases} \min\left(\frac{\tilde{x}_i^{\mathrm{B,out},(\tau)}}{x_i^{\mathrm{B,out},(\tau)}}, \frac{\tilde{x}_j^{\mathrm{B,in},(\tau)}}{x_j^{\mathrm{B,in},(\tau)}}\right) x_{ij}^{\mathrm{B},(\tau)} & \text{if } x_{ij}^{\mathrm{B},(\tau)} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{26}$$

6. Update the flow entering and leaving the network with

$$\boldsymbol{\sigma}_{\mathrm{in}}^{(\tau+1)} = \boldsymbol{\rho}_{\mathrm{in}} - (\widetilde{\mathbf{X}}_{\mathrm{B}}^{(\tau)})^{\top}\mathbf{e}_n, \tag{27}$$

$$\boldsymbol{\sigma}_{\mathrm{out}}^{(\tau+1)} = \boldsymbol{\rho}_{\mathrm{out}} - \widetilde{\mathbf{X}}_{\mathrm{B}}^{(\tau)}\mathbf{e}_n. \tag{28}$$

7. Compute the *reducing factor matrix* $\mathbf{R}^{(\tau)} = (r_{st}^{(\tau)})$ with

$$r_{st}^{(\tau)} = 1 - \max_{ij}\left(\mathrm{P}_{st}(i,j)\frac{x_{ij}^{\mathrm{B},(\tau)} - \widetilde{x}_{ij}^{\mathrm{B},(\tau)}}{x_{ij}^{\mathrm{B},(\tau)} + \epsilon}\right). \tag{29}$$

8. Update the origin-destination affinity matrix with

$$\mathbf{S}^{(\tau+1)} = \mathbf{S}^{(\tau)} \odot \mathbf{R}^{(\tau)}, \tag{30}$$

where $\odot$ designates the Hadamard (component-wise) product of matrices.