Estimation of flow trajectories in a multiple lines network

Experiments with *transports publics de la région lausannoise* (tl) data

---

Guillaume Guex
Romain Loup
François Bavaud

University of Lausanne

Table of contents

Introduction

Context

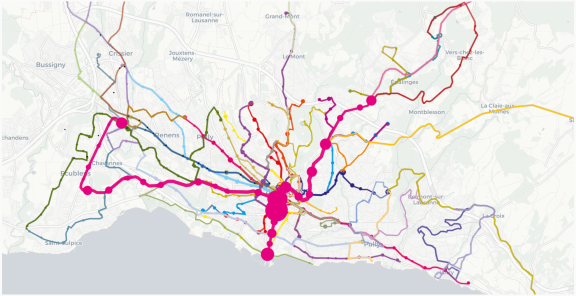
The tl dataset, exploited by Romain Loup for his PhD:

- 1 year of data (2019).
- 115 millions of passengers.
- 42 bus and subway lines.
- 1361 stops and 497 “superstops”.
- Every journey data: traveling time, waiting time, embarking and disembarking passengers at each stops, etc.

Context

```
##   stop_id  stop_name line_id direction order embarkment disembarkment
## 1 MALAD_N  Maladière       1         A     1     164558             0
## 2 MTOIE_E    Montoie       1         A     2     136236         12705
## 3 BATEL_E  Batelière       1         A     3     203045         13409
## 4 RTCOU_E Riant-Cour       1         A     4     156015         24909

##    stop_id  stop_name line_id direction order embarkment disembarkment
## 42 RTCOU_O Riant-Cour       1         R    19      23634        132201
## 43 BATEL_O  Batelière       1         R    20      13707        168884
## 44 MTOIE_O    Montoie       1         R    21       4259        128255
## 45 MALAD_N  Maladière       1         R    22          0        146798
```



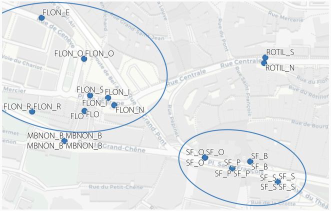
The multiple lines network

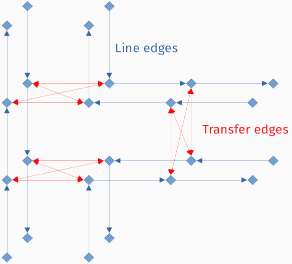
Having only lines data, the structure is a disconnected oriented graph.

In addition to line edges, it is possible to construct transfer edges to make the graph connected, by using, e.g.,

- Superstops names,
- Pedestrian time,
- Euclidean distance.

With transfer edges, we have a unilaterally connected graph.





Problematic

This dataset offers multiple axes of research. In this presentation, we focus on one question:

> *Knowing (1) the network structure and (2) the number of passengers embarking and disembarking at each stop, can we deduce trajectories of the passengers in the network ?*

Short answer: **No**.

Thank you for your attention !
Questions ?

(just kidding)

Problematic

Exact trajectories are impossible to know, but, with additional assumptions, we can produce estimations of them.

We divide this problematic into two parts:

- Estimation of trajectories on a single line.
- Estimation of trajectories on the multiple lines network.

The single line problem

Formal problem definition

Let a line (in one direction), which have n stops, indexed regarding line order. Let $\boldsymbol{\rho}_{\text{in}} = (\rho_s^{\text{in}})$ and $\boldsymbol{\rho}_{\text{out}} = (\rho_t^{\text{out}})$ be two vectors representing, respectively, the **passengers entering and leaving the line at each stop**.

We search a $(n \times n)$ **origin-destination matrix** $\mathsf{N} = (n_{st})$ where components represents

n_{st} = "the number of passengers entering line at s and leaving at t".

These components must verify

1. $n_{st} \geq 0$,
2. $n_{s\bullet} = \rho_s^{\text{in}}$,
3. $n_{\bullet t} = \rho_t^{\text{out}}$.

($\bullet$ indicates a sum on the replaced index)

Formal problem definition

It reads

$$
\mathbf{N} = \begin{array}{cc}
 & \begin{array}{ccccc} \rho_1^{\text{out}} & \rho_2^{\text{out}} & \cdots & \rho_{n-1}^{\text{out}} & \rho_n^{\text{out}} \end{array} \\
\begin{array}{c} \rho_1^{\text{in}} \\ \rho_2^{\text{in}} \\ \vdots \\ \rho_{n-1}^{\text{in}} \\ \rho_n^{\text{in}} \end{array} &
\begin{pmatrix}
n_{11} & n_{12} & \cdots & n_{1,n-1} & n_{1n} \\
n_{21} & n_{22} & \cdots & n_{2,n-1} & n_{2n} \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
n_{n-1,1} & n_{n-1,2} & \cdots & n_{n-1,n-1} & n_{n-1,n} \\
n_{n,1} & n_{n,2} & \cdots & n_{n,n-1} & n_{n,n}
\end{pmatrix}
\end{array}
$$

In fact, we already know that some components are null

$$
\mathbf{N} = \begin{array}{cc}
 & \begin{array}{ccccc} 0 & \rho_2^{\text{out}} & \cdots & \rho_{n-1}^{\text{out}} & \rho_n^{\text{out}} \end{array} \\
\begin{array}{c} \rho_1^{\text{in}} \\ \rho_2^{\text{in}} \\ \vdots \\ \rho_{n-1}^{\text{in}} \\ 0 \end{array} &
\begin{pmatrix}
0 & n_{12} & \cdots & n_{1,n-1} & n_{1n} \\
0 & 0 & \ddots & \cdot & n_{2n} \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
0 & 0 & \cdots & 0 & n_{n-1,n} \\
0 & 0 & \cdots & 0 & 0
\end{pmatrix}
\end{array}
$$

Formal problem definition

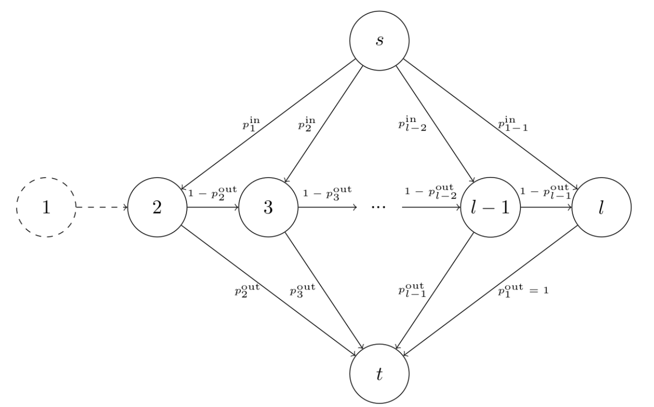
In this form, the problem is ill posed, because it has multiple acceptable solutions. An exemple of solution is to make passengers follow a first in, first out (FIFO) scheme.

A principle of mathematical modeling is to find the solution which makes the least assumptions about passenger behavior, in other words the maximum entropy solution.

In this case, it translates by supposing that there is the same probability of leaving the line for every passenger which have traveled at least one stop.

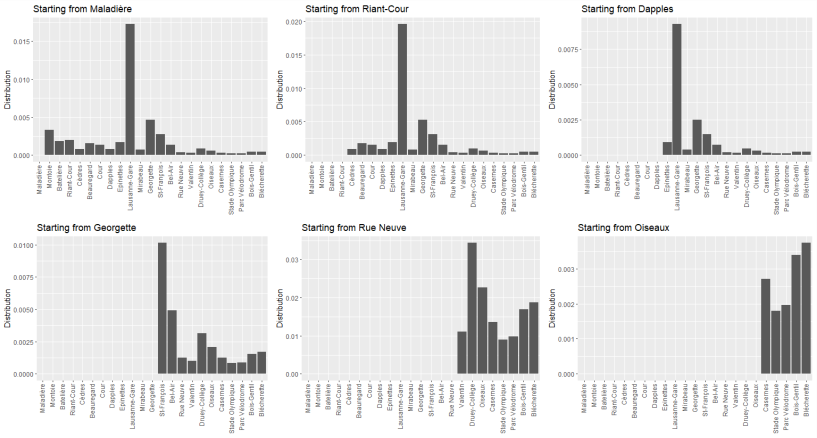
Solution with Markov chain modeling

We can then model passenger flow with a Markov chain:

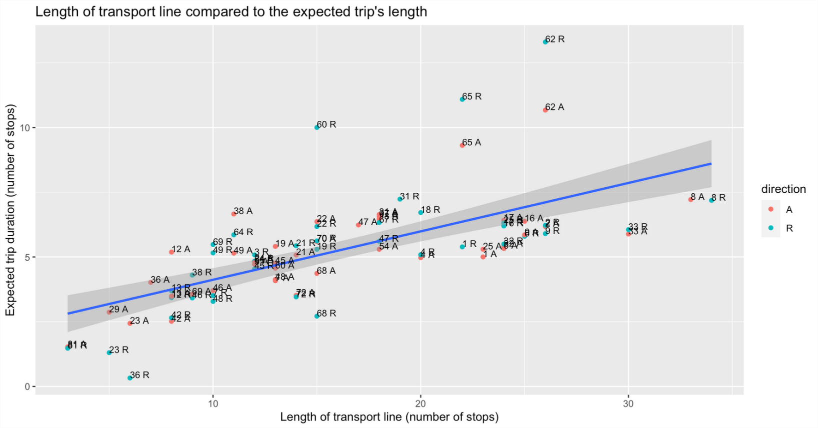


with $p_i^{\text{in}} = \frac{\rho_i^{\text{in}}}{\rho_\bullet^{\text{in}}}$ and $p_i^{\text{out}} = \frac{\rho_i^{\text{out}}}{\sum_{1 \leq k \leq (i-1)} (\rho_k^{\text{in}} - \rho_k^{\text{out}})}$.

Solution with Markov chain modeling



Solution with Markov chain modeling



Iterative proportial fitting

Iterative proportional fitting

The same solution can be obtained with the iterative proportional fitting (IPF) algorithm [Bishop et al., 1975]. Let

1. $\mathrm{P} = (p_{ij})$ a $(n \times m)$ matrix,
2. $\mathbf{u} = (u_i)$ a n-length vector, and
3. $\mathbf{v} = (v_i)$ a m-length vector,

all of them with strictly positive components. We can find two vectors $\mathbf{a} = (a_i)$ and $\mathbf{b} = (b_i)$ such that the matrix $\mathrm{Q} = (q_{ij})$, defined with

$$q_{ij} = a_i b_j p_{ij},$$

verifies

- $q_{i\bullet} = u_i$,
- $q_{\bullet j} = v_j$,
- $K(\mathrm{Q}|\mathrm{P}) := \sum_{ij} \frac{q_{ij}}{q_{\bullet\bullet}} \log\left(\frac{q_{ij}/q_{\bullet\bullet}}{p_{ij}/p_{\bullet\bullet}}\right)$ is minimum.

Solution with iterative proportional fitting

In our context, it means that if we define an origin-destination affinity matrix $\mathsf{S} = (s_{st})$ with

$$\mathsf{S} = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 0 & 0 & 1 & \cdots & 1 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 0 \end{pmatrix}$$

we can find the same maximum entropy solution with iterative proportional fitting, i.e., we can find $\mathbf{a} = (a_s)$ and $\mathbf{b} = (b_t)$, such that $n_{ij} = a_s b_t s_{st}$ verifies

1. $n_{s\bullet} = \rho_s^{\text{in}}$,
2. $n_{\bullet t} = \rho_t^{\text{out}}$,
3. $K(\mathsf{N}|\mathsf{S})$ is minimum.

(a small number ϵ has to be added on null components).

Solution with iterative proportional fitting

By decreasing (resp. increasing) s_{st}, we reduce (resp. expand) the resulting number of passengers going from s to t obtained with IPF.

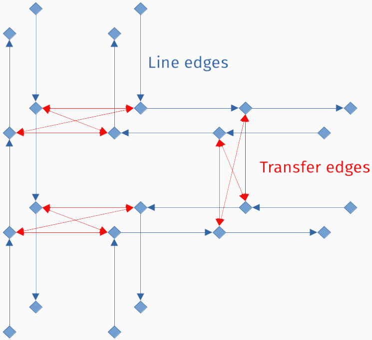
Thus, this approach is more flexible, because we could give a specific affinity matrix $\mathrm{S} = (s_{st})$, based on other data (additional assumptions).

The relationship $n_{st} = a_s b_t s_{st}$ can be seen as a gravity model (still to investigate).

The multiple lines problem

The multiple lines problem

In this problem, the whole multiple lines network is used. Let us recall that it is composed of line edges and transfer edges.



It is not possible to use a Markov chain modeling in this case, but the iterative proportional fitting approach is still valid. However, there are additional difficulties.

Flow behavior

The first difficulty is that there are generally multiple routes to go from node s to t. This can be solved by making a new assumption.

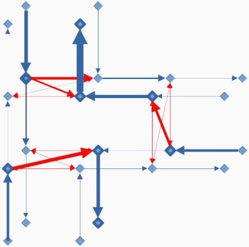
In the multiple lines network, we suppose that passengers take shortest-paths in order to reach node t from node s.

If multiple shortest-paths exists between s and t, the passenger flow is divided equally among them.

Flow behavior

This assumption unlocks a very useful property. If we have an origin-destination matrix $\mathsf{N} = (n_{st})$, we can compute the $(n \times n)$ flow matrix on edges $\mathsf{X} = (x_{ij})$.

$$\mathsf{N} = \begin{pmatrix} 0 & 13055 & 243 & \cdots & 144 \\ 3498 & 0 & 24429 & \cdots & 7523 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1508 & & \cdots & 0 & 5093 \\ 8903 & 6343 & \cdots & 53 & 0 \end{pmatrix} \rightarrow$$



The flow in/out lines/network

A second problem is that we have to distinguish between

- The passengers **entering and leaving lines** at each stop, represented by vectors $\boldsymbol{\rho}_{\text{in}}$ and $\boldsymbol{\rho}_{\text{out}}$, which are **measured**, and
- The passengers **entering and leaving the network** at each stop, represented by vectors $\boldsymbol{\sigma}_{\text{in}}$ and $\boldsymbol{\sigma}_{\text{out}}$, which are **unknown**.

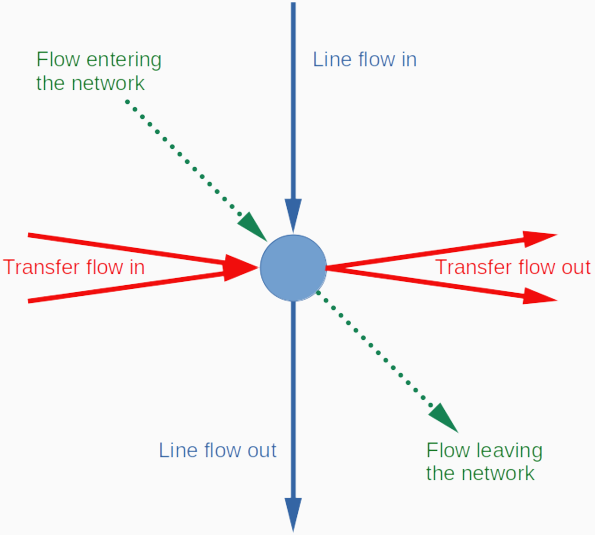
At each stop i, we have

$$\begin{aligned} \rho_i^{\text{in}} &= \sigma_i^{\text{in}} + x_{\bullet i}^{\text{B}}, \\ \rho_i^{\text{out}} &= \sigma_i^{\text{out}} + x_{i \bullet}^{\text{B}}, \end{aligned}$$

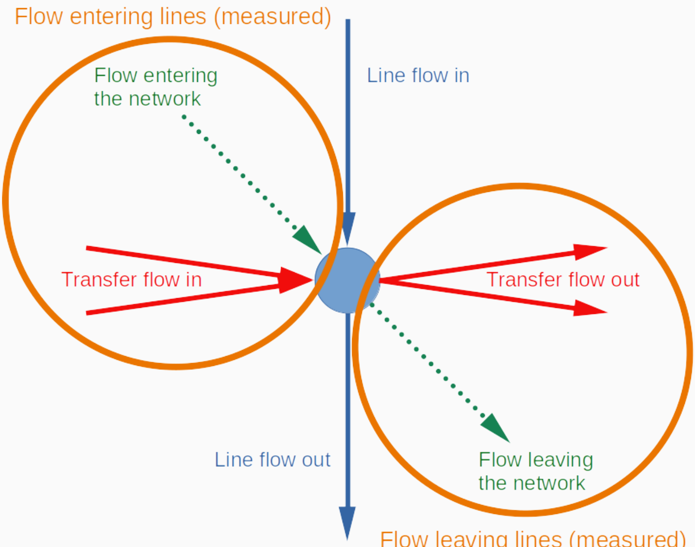
where

- $x_{\bullet i}^{\text{B}}$ is the **transfer flow entering the node** i and
- $x_{i \bullet}^{\text{B}}$ is the **transfer flow leaving the node** i.

The flow in/out lines/network



The flow in/out lines/network



The flow in/out lines/network

If we know **transfer flow on edges**, i.e. $\mathsf{X}_{\mathrm{B}} = (x_{ij}^{\mathrm{B}})$, we can **compute the flow entering/leaving the network**, $\boldsymbol{\sigma}_{\mathrm{in}}$ and $\boldsymbol{\sigma}_{\mathrm{in}}$.

The flow entering/leaving the lines, ρ_i^{in} and ρ_i^{out}, acts as **constraints on the in/out transfer flow**, $x_{\bullet i}^{\mathrm{B}}$ and $x_{i\bullet}^{\mathrm{B}}$.

When there are no transfers on i, we have $\sigma_i^{\mathrm{in}} = \rho_i^{\mathrm{in}}$ and $\sigma_i^{\mathrm{out}} = \rho_i^{\mathrm{out}}$.

Let us make an outline for the iterative Algorithm. There are 4 steps at each iteration:

$$\left.\begin{array}{r}\text{OD affinity matrix } \mathbf{S}\\ \text{Flow in the network } \boldsymbol{\sigma}_{\text{in}}\\ \text{Flow out the network } \boldsymbol{\sigma}_{\text{out}}\end{array}\right\} \xrightarrow{IPF} \text{OD matrix } \mathbf{N} \tag{1}$$

$$\text{OD matrix } \mathbf{N} \xrightarrow{SP} \text{Transfer flow } \mathsf{X}_{\mathrm{B}} \tag{2}$$

$$\left.\begin{array}{r}\text{Transfer flow } \mathsf{X}^{\mathrm{B}}\\ \text{Flow in lines } \boldsymbol{\rho}_{\text{in}}\\ \text{Flow out lines } \boldsymbol{\rho}_{\text{out}}\end{array}\right\} \xrightarrow{\text{Constraints}} \left\{\begin{array}{l}\text{Allowed transfer flow } \widetilde{\mathsf{X}}_{\mathrm{B}}\\ \text{Flow in the network } \boldsymbol{\sigma}_{\text{in}}\\ \text{Flow out the network } \boldsymbol{\sigma}_{\text{out}}\end{array}\right. \tag{3}$$

$$\left.\begin{array}{r}\text{Transfer flow } \mathsf{X}_{\mathrm{B}}\\ \text{Allowed transfer flow } \widetilde{\mathsf{X}}_{\mathrm{B}}\end{array}\right\} \xrightarrow{\text{Affinity update}} \text{OD affinity matrix } \mathbf{S} \tag{4}$$

Step 1: iterative proportional fitting

At the beginning of the algorithm, we have to set

- An initial flow in the network . We can set it to $\boldsymbol{\sigma}_{\text{in}}^{\text{init}} = \boldsymbol{\rho}_{\text{in}}$,
- An initial flow out the network . We can set it to $\boldsymbol{\sigma}_{\text{out}}^{\text{init}} = \boldsymbol{\rho}_{\text{out}}$,
- An initial affinity matrix between origin-destination, S^{init}.

The initial affinity matrix $\mathsf{S}^{\text{init}} = (s_{st}^{\text{init}})$ is crafted in order to have

- $s_{st}^{\text{init}} = 1$, if going from s to t is a valid trajectory for using the network,
- $s_{st}^{\text{init}} = 0$, otherwise.

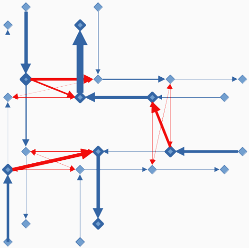
It now possible to obtain a first origin-destination matrix N with iterative proportional fitting.

Step 2: shortest-paths flow

In this step, we use the assumption that passengers use **shortest-paths** in the network to obtain flow on edges, and in particular, **flow on transfer edges**:

$$\mathsf{N} \longrightarrow \mathsf{X}_{\mathsf{B}}$$

$$\mathsf{N} = \begin{pmatrix} 0 & 13055 & 243 & \cdots & 144 \\ 3498 & 0 & 24429 & \cdots & 7523 \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ 1508 & & \cdots & 0 & 5093 \\ 8903 & 6343 & \cdots & 53 & 0 \end{pmatrix} \rightarrow$$

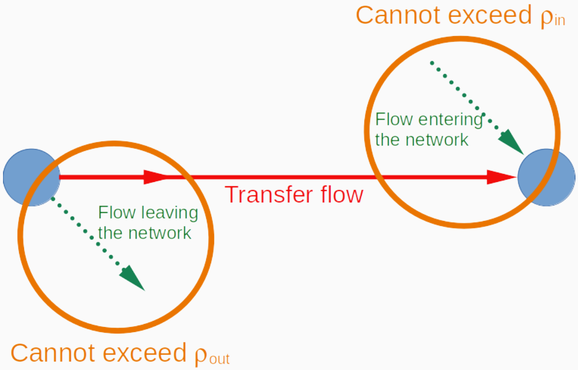


Step 3: corrected transfer flow

This transfer flow X_B could be used to update $\boldsymbol{\sigma}_{\text{in}}$ and $\boldsymbol{\sigma}_{\text{out}}$, with

$$\sigma_i^{\text{in}} = \rho_i^{\text{in}} - x_{\bullet i}^{\mathrm{B}},$$
$$\sigma_i^{\text{out}} = \rho_i^{\text{out}} - x_{i\bullet}^{\mathrm{B}}.$$

However, there is no guarantee that the transfer flow do not exceed limits given by $\boldsymbol{\rho}_{\text{in}}$ and $\boldsymbol{\rho}_{\text{in}}$.



Step 3: allowed transfer flow

For each x_{ij}^{B}, we use ρ_i^{out} and ρ_j^{in} in order to compute a allowed transfer flow $\widetilde{x}_{ij}^{\mathrm{B}}$, with $\widetilde{x}_{ij}^{\mathrm{B}} \leq x_{ij}^{\mathrm{B}}$. There are multiple choices:

1. Constraint thresholds could be reachable.
2. We can set a percentage limit for the transfer flow among the flow in/out of the lines.
3. We can set a soft limit to the transfer flow, with, e.g., an exponential law.

When this allowed flow is computed, $\boldsymbol{\sigma}_{\mathrm{in}}$ and $\boldsymbol{\sigma}_{\mathrm{out}}$ can be updated with

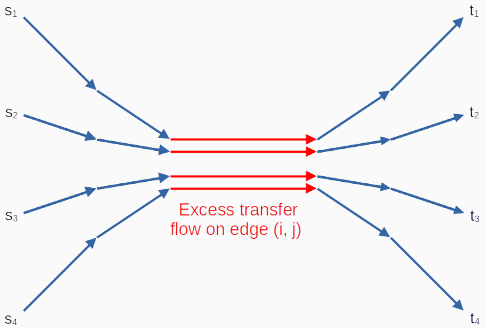
$$
\begin{aligned}
\sigma_i^{\mathrm{in}} &= \rho_i^{\mathrm{in}} - \widetilde{x}_{\bullet i}^{\mathrm{B}}, \\
\sigma_i^{\mathrm{out}} &= \rho_i^{\mathrm{out}} - \widetilde{x}_{i \bullet}^{\mathrm{B}}.
\end{aligned}
$$

Step 4: Affinity update

What about the excess flow on transfer edges, i.e., $x_{ij}^{B} - \widetilde{x}_{ij}^{B}$?

Note that we only updated margins distribution, but we also need a way to reduce dependencies between particular s and t.

Having flow following shortest-paths, we know which couples s, t are "responsible" for the excess flow on edge (i, j).



Step 4: Affinity update

On each transfer edge, we can compute the **proportion of allowed flow**:

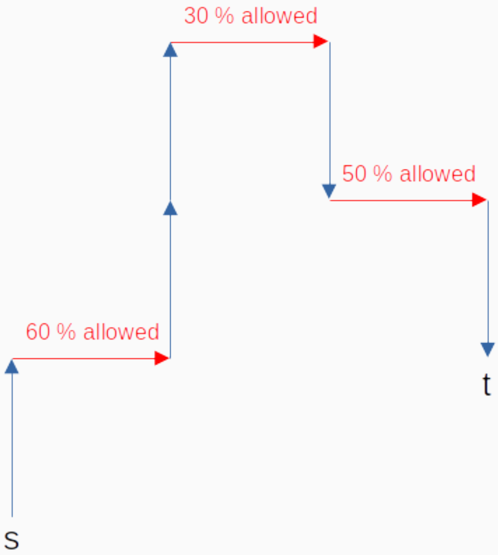
$$p_{ij}^{\text{allowed}} = \frac{\widetilde{x}_{ij}^{\text{B}}}{x_{ij}^{\text{B}}}$$

Each transfer edge (i, j) transmits this proportion p_{ij}^{allowed} to all couples s, t that use this edge. Each couple s, t receiving a reduced proportion have to **reduce its affinity s_{st}**.

Each couple s, t receives a **list of proportion of allowed flow**, from all transfer edges on its shortest-path. The **update factor for the affinity s_{st}** is constructed by using the minimum allowed flow received.

$$s_{st}^{\text{new}} = \min\{p_{i_1,j_1}^{\text{allowed}}, \ldots, p_{i_k,j_k}^{\text{allowed}}\} \cdot s_{st}^{\text{old}}$$

Step 4: Affinity update



In this example, $s_{st}^{\text{new}} = 0.3 \cdot s_{st}^{\text{old}}$

Results (demo)

Conclusion

Conclusion

As it is still a work in progress, the conclusion takes the form of a to-do list:

- Optimizing speed/memory (almost done).
- Running the algorithm on all lines.
- Testing multiple initial conditions and hyperparameter values.
- Finding applications.
- Making links with gravity model, transportation problem, ...
- Using carefully crafted OD affinities.
- Finding proof/conditions of convergence.
- Adapting the algorithm to closely related problems.

Thank you for your attention !
Questions ?

(for real this time)