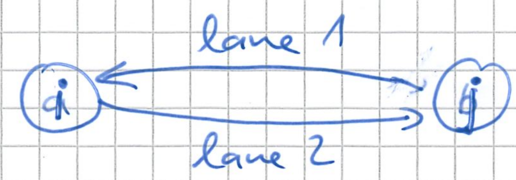
Chapter 1: The constraints

Let $G = (V, E)$ be a directed graph. We must make it directed to keep track of one way streets and to enforce constraints. As most vertices are connected by two-way streets we typically have two edges



For every ordered pair of nodes $w = (f, t) \in W \subseteq V^2$ we have a demand for transportation $d_w \geq 0$. We will use both the notation with index w and with index (f, t) in the following.

The TAP is usually formulated in the following variables:

$h_{r,w}$: flow of vehicles from f to t via path r (Note that $w = (f,t)$).

x_e : Total flow along an edge e

These quantities are related by several topological quantities

$$\Delta_{er} = \begin{cases} +1 & \text{if path } r \text{ uses edge } e \\ 0 & \text{otherwise} \end{cases}$$

$$\lambda_{rw} = \begin{cases} +1 & \text{if path } r \text{ originates at node } f \text{ and terminates at node } t \text{ where } w = (f, t) \\ 0 & \text{otherwise} \end{cases}$$

Then we have the fundamental constraints.

$$x_e = \sum_w \sum_r \Delta_{er} \, h_{r,w} \qquad (1)$$

$$d_w = \sum_r \lambda_{r,w} \, h_{r,w} \qquad (2)$$

Here we propose a different formulation inspired by Kirchhoff's laws from circuit theory. First we introduce the node-edge incidence matrix

$$E_{n,e} = \begin{cases} +1 & \text{if edge } e \text{ starts at node } n \\ -1 & \text{" terminates "} \\ 0 & \end{cases}$$

Then we have

$$\sum_e E_{ne} \cdot \Delta_{er} = \begin{cases} -1 & \text{if } n \text{ is the terminal node of path } r \\ +1 & \text{if } n \text{ is the starting node of path } r \\ 0 & \text{otherwise} \end{cases}$$

$$= \sum'_{u} \lambda_{r,(n\to u)} - \sum'_{v} \lambda_{r,(u\to n)}$$

Furthermore we need the relations

kronecker δ !!!

$$\lambda_{r,u\to n}\,\lambda_{r,f\to v} = \lambda_{r,f\to n}\,\delta_{v,n}\,\delta_{u,f}$$

$$\lambda_{r,f\to u}\,\lambda_{r,f\to v} = \lambda_{r,n\to u}\,\delta_{u,v}$$

We define the flow over edge e that originates from node f and leads to an arbitrary other node $v \neq f$:

$$x_e^{(f)} = \sum'_{v\neq f} \underbrace{\sum'_{r} \lambda_{r,f\to v}\,\Delta_{e,r}\,h_{r,f\to v}}_{\text{Flow over edge } e \text{ from } f \text{ to } v}$$

Then we obtain (for $n \neq f$):

$$\sum'_{e} E_{ne}\, x_e^{(f)} = \sum'_{e}\sum'_{v}\sum'_{r} E_{ne}\,\lambda_{r,f\to v}\,\Delta_{e,r}\,h_{r,f\to v}$$

$$= \sum'_{v}\sum'_{r} \underbrace{\Big(\sum'_{e} E_{ne}\,\Delta_{e,r}\Big)}_{\sum'_{u}\lambda_{r,(n\to u)} - \sum'_{u}\lambda_{r,(u\to n)}} \lambda_{r,f\to v}\,h_{r,f\to v}$$

$$= \sum'_{r}\Big(\overbrace{\sum'_{v,u}\lambda_{r,(n\to u)}\,\lambda_{r,f\to v}}^{=0 \text{ weil } n\neq f} - \underbrace{\sum'_{v,u}\lambda_{r,(u\to n)}\,\lambda_{r,f\to v}}_{=\text{ see above}}\Big)\, h_{r,f\to v}$$

$$= \sum_r{}' - \sum_{u,v}{}' \lambda_{r,f\to u}\, \delta_{v,n}\, \delta_{u,f}\, h_{r,f\to v}$$

$$= - \sum{}' \lambda_{r,f\to n}\, h_{r,f\to n}$$

$$= - d_{f\to n}$$

Similarly we find for $n = f$:

$$\sum_e{}' \hat{E}_{ne}\, x_e^{(f)} = \sum_r{}' \sum_{u,v}{}' \Big(\lambda_{r,f\to u} \lambda_{r,f\to v} - \overbrace{\lambda_{r,u\to f} \lambda_{r,f\to v}}^{=0,\text{ as } v\neq f} \Big) \cdot h_{r,f\to v}$$

$$= \sum_r{}' \sum_{v\neq n}{}' \lambda_{r,f\to v}\, h_{r,f\to v}$$

$$= \sum_{v\neq n}{}' \underbrace{\sum_r{}' \lambda_{r,f\to v}\, h_{r,f\to v}}_{d_{f\to v}}$$

$$= \sum_{v\neq f}{}' d_{f\to v}$$

Thus: Defining the source vector $\vec{s}^{(f)}$ with components

$$s_n^{(f)} = \begin{cases} - d_{f\to n} & n\neq f \\ + \sum_{v\neq f}{}' d_{f\to v} & n = f \end{cases}$$

the demand law (2) can equivalently formulated as

$$\sum_e{}' E_{ne}\, x_e^{(f)} = s_n^{(f)} \qquad \forall\, n, f$$

or in vectorial form

$$E\,\vec{x}^{(f)} = \vec{s}^{(f)}$$

The total flow over an edge e (Eq. (1)) then reads

$$\vec{x} = \sum_f{}' \vec{x}^{(f)}$$

$$x_e = \sum_f{}' x_e^{(f)}$$

Final note: We should formulate this as a proper lemma. In particular, I have shown that

$$\begin{matrix} (1) \\ (2) \end{matrix} \Longrightarrow \begin{matrix} E\,\vec{x}^{(f)} = \vec{s}^{(f)} \\ \vec{x} = \sum_f{}' \vec{x}^{(f)} \end{matrix}.$$

We must make sure that the reverse $\Leftarrow$ also holds to establish full equivalence.

Chapter 16: The inequality constraint.

Intuitively we should have

$$h_{r,f\to t} \geq 0 \qquad ③$$

in the original path based formulation.

Now we have defined

$$x_e^{(f)} = \sum_{v\neq f}' \sum_r' \underbrace{\lambda_{r,f\to v}\,\Delta_{e,r}}_{\geq 0}\, h_{r,f\to v}$$

Hence we see directly that ③ implies directly

$$x_e^{(f)} \geq 0 .$$

Can we also show the reverse statement?

Hypothesis

Given $\vec{x}^{(f)}$ with $x_e^{(f)} \geq 0 \quad \forall\, e$ then we can find variables

$$h_{r,f\to t} \qquad \begin{array}{l} r: \text{path from } f\to t \\ t\neq f \quad \text{vertex} \end{array}$$

such that:

$$h_{r,f\to t} \geq 0$$

$$\sum_t' \sum_{r\in \text{path}(f\to t)}' h_{r,f\to t}\,\Delta_{er} = x_e^{(f)}$$

Note: The more i think about this, the more trivial it appears...

Chapter 2: The optimization problem.

The original TAP reads

$$\min \sum_e \int_0^{x_e} t_e(x')\, dx'$$

s.t. (1) and (2)

$$x_e \geq 0$$ ← Do we need this for all O-D pairs separately?

Let's be definite and use $n \in \mathbb{N}$

$$t_e(x') = \tau_e + \beta_e (x')^n$$

$$\Rightarrow \int_0^{x_e} t_e(x')\, dx' = \tau_e x_e + \frac{\beta_e}{n+1} x_e^{n+1}$$

We have shown that (1) and (2) can be reformulated as

$$E \vec{x}^{(f)} = \vec{s}^{(f)}$$

$$\vec{x} = \sum_f \vec{x}^{(f)}$$

Hence we have the optimization problem

$$\min \underbrace{\sum_e \tau_e x_e + \frac{\beta_e}{n+1} x_e^{n+1}}_{=F}$$

s.t. $E\vec{x}^{(f)} = \vec{s}^{(f)}$ $\qquad \vec{x} = \sum_f \vec{x}^{(f)}$

and $x_e^{(f)} \geq 0$

Now evaluate the KKT conditions for this optimization problem:

$$\frac{\partial F}{\partial x_e^{(f)}} = \tau_e + \beta x_e^n$$

$$\Rightarrow \tau_e + \beta_e \left[x_e^n + \sum_n \lambda_n^{(f)} E_{ne} + \mu_e^{(f)} = 0 \right.$$

(stationarity)

Primal feasibility:

$$E_{ne} x_e^{(f)} = s_n^{(f)} \quad \forall f, n \in V$$

$$x_e^{(f)} \geq 0$$

Dual feasibility:

$$\mu_e^{(f)} \geq 0$$

$$\mu_e^{(f)} x_e^{(f)} = 0 \ldots$$

Now lets denote the edges by their endpoints, i.e replace e by (i,j). Then for every edge there are two possibilities:

(a) $x_{ij}^{(f)} = 0$

$$\Rightarrow \mu_{ij}^{(f)} = -\tau_{ij} + \left(\lambda_i^{(f)} - \lambda_j^{(f)}\right)$$

(b) $\mu_{ij}^{(f)} = 0$

$$\Rightarrow x_{ij}^{(n)} = \beta_{ij}^{-1} \left[-\tau_{ij} + \left(\lambda_i^{(f)} - \lambda_j^{(f)}\right)\right]$$

Now the funny bit is that for case (b):

$$x_{ij}^{u} = \beta_{ij}^{-1}\left[-\tau_{ij} + \left(\lambda_i^{(f)} - \lambda_i^{(j)}\right)\right]$$

$\boxed{\text{Independent of } (f)\ !}$

Hence, for if $\mu_{ij}^{(f)} = 0$ and $\mu_{ij}^{(f')} = 0$ for two vertices $f \neq f'$, then

$$\lambda_i^{(f)} = \lambda_i^{(f')}$$

$$\lambda_j^{(f)} = \lambda_j^{(f')}$$

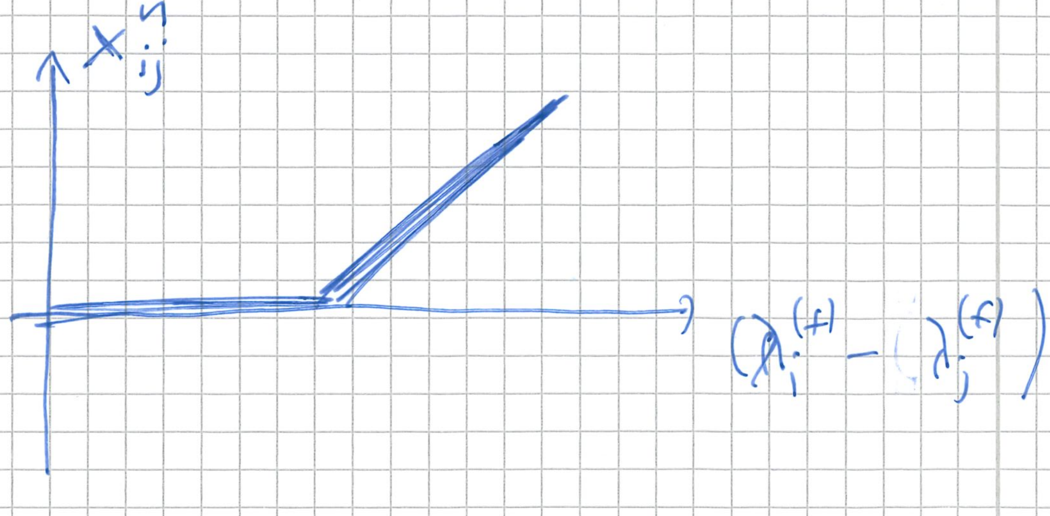
I wonder if we can establish that

$\lambda_i^{(f)} \stackrel{?}{=} \lambda_i^{(f')}$ for <u>all</u> vertices i

<u>all</u> source vertices f, f'.

That would indeed simplify things a lot.

Furthermore we get a similar picture as in M. Dahlmanns' thesis.



Chapter III Approximations

We note that the objective function $\mathcal{F}$ is strictly convex.

We have recently established a method to <u>rigorously</u> bound the error of any approximate solution of such* an optimization problem. This could be very interesting here as we could bound how much traffic flow in the congested regime differs from the free traffic flow.

* There is one tiny caveat here. Up to now we only worked with equality constraints. Now we would have to see if we can generalize this approach to include <u>inequality</u> constraints.

** There may be a way out of this, though. MD has established that in the case of <u>two-lane</u> traffic we may get rid of the inequality constraint... but only for single source traffic up to now.

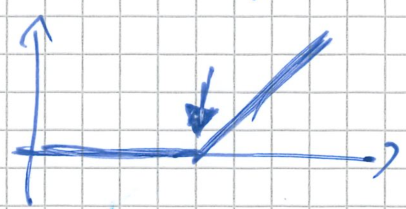
Chapter IV numerics

Working with the Kirchhoff approach we only have $|V| \times |E|$ free variables

$$x_e^{(f)} .$$

This might facilitate numerical calculations quite a bit.

Notably, it is not advisable to work with the Lagrangian multipliers λ_i. The non-differentiability in the relation



makes it extremely hard to achieve convergence in any iterative scheme.

At least this is MD's experience, but he might have missed better algorithms.