Path-based algorithms for solving traffic assignment

CE 392C

OUTLINE

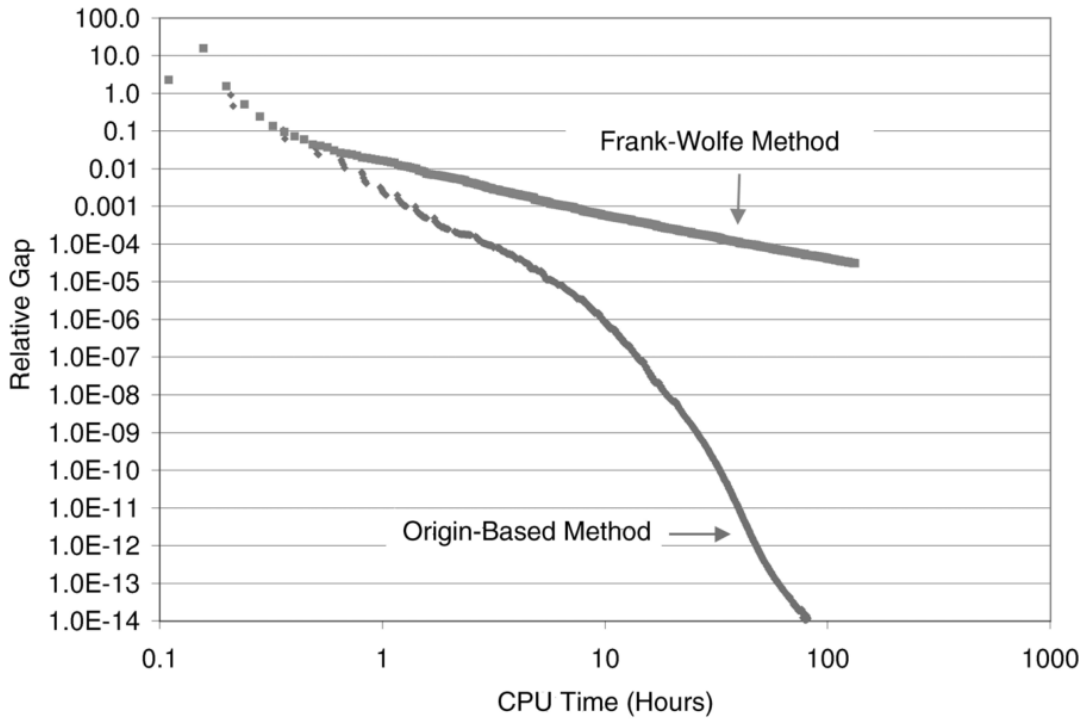
Faster methods for finding equilibrium

1. What's wrong with Frank-Wolfe on large networks
2. Gradient projection: a smarter version of "trial-and-error"
3. Newton's method for traffic assignment

For the next few weeks, we'll return to the basic traffic assignment problem. Think about how these ideas might be applied to the extensions as well...

WHAT'S WRONG WITH FRANK-WOLFE?

Frank-Wolfe starts off well, and usually makes great progress in the first few iterations.



(Bar-Gera, 2002)

However, its convergence slows down very quickly.

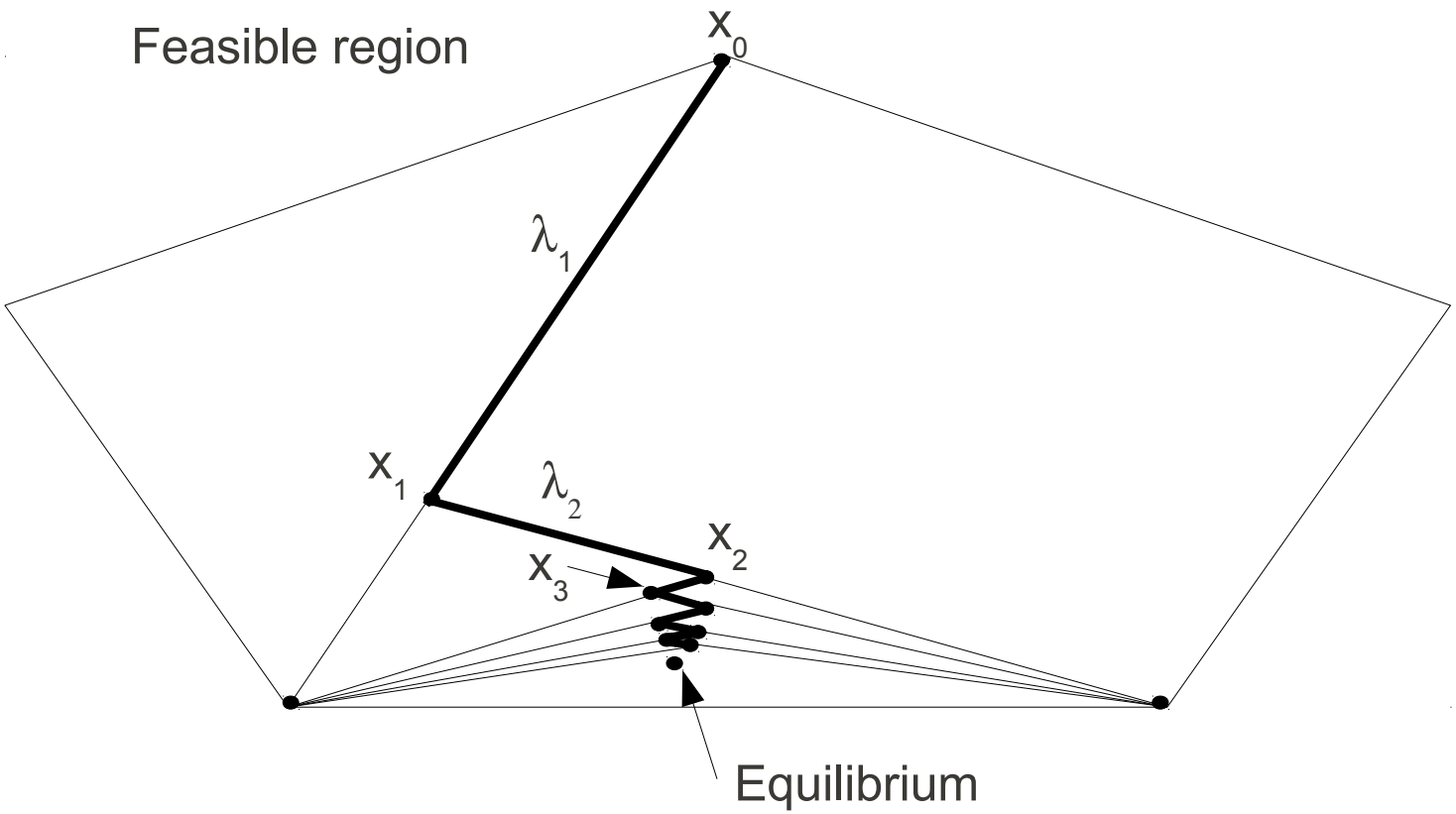
Why is this the case? Three major reasons:

1. FW adjusts all OD pairs by the same amount λ, even if some are closer to equilibrium than others.
2. Geometrically, the FW target is always a corner of the feasible region.
3. It is unable to erase cyclic flows.

Uniform treatment of OD pairs

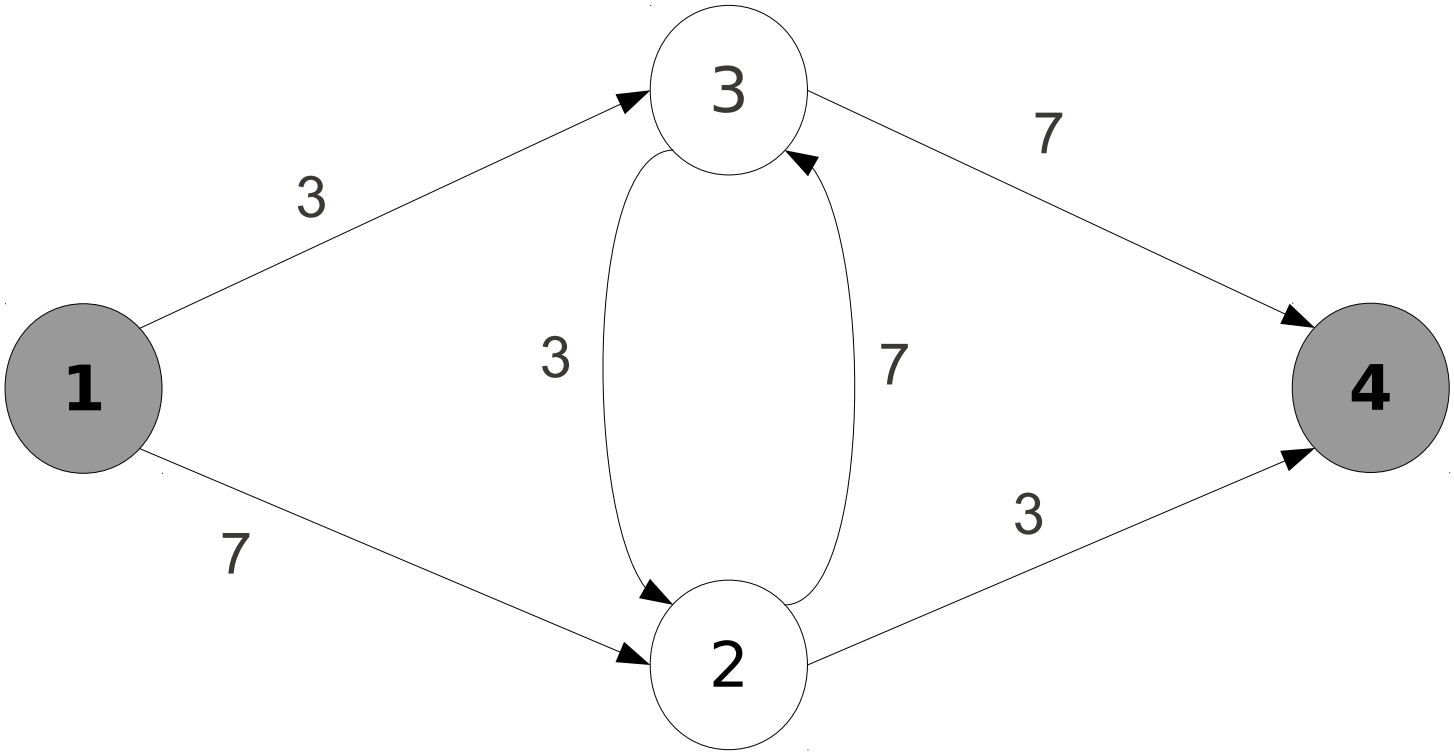
OD pair 1 is using two paths with travel times of 10 and 100 minutes; OD pair 2 is using two paths with travel times 10 and 10.1 minutes.

Limited target selection



FW can only move towards corner points.

Erasing cyclic flows



These link flows cannot represent an equilibrium solution, but FW will not erase them.

The first and third of these drawbacks are shared by all *link-based algorithms*, that is, algorithms which only need to keep track of link flows $\mathbf{x}$.

MSA falls into the same category, and is usually even worse than FW.

The major advantage of link-based methods is that they require *little computer memory*. This was a big deal back in the 1970s when FW was first used for traffic assignment. It's not as important today.

The major disadvantage is that you *lose a lot of information* when you aggregate all OD pairs together. (For instance, you can't adjust one OD pair's flows apart from the others.)

The algorithms you will see next store more information about the solution. They use more memory, but leads to a much faster solution.

GRADIENT PROJECTION

Gradient projection is an example of a *path-based algorithm*, which tracks the path flows $\mathbf{h}$ in addition to the link flows $\mathbf{x}$.

Since there can be billions and billions of paths in a network, we don't want to keep track of literally every path's flow.

Instead, we define a set of working paths $\hat{\Pi}^{rs}$ for each OD pair, and only track the flows on these paths. We'll let this set grow and shrink over successive iterations.

A general scheme for path-based algorithms:

1. Initialize $\hat{\Pi}^{rs} \leftarrow \emptyset$ for all OD pairs.
2. Find the shortest path π^*_{rs} for each OD pair. Add it to $\hat{\Pi}^{rs}$ if it's not already used.
3. Within each OD pair, shift travelers among paths to get closer to equilibrium.
4. Update travel times; drop paths from $\hat{\Pi}^{rs}$ if they are no longer used; return to step 2.

This should remind you of the trial-and-error method, with a "relaxed" step 3 and the "try again" steps spelled out more clearly.

Step 3 is where path-based algorithms usually differ.

The *gradient projection* algorithm uses Newton's method to try to move closer to an equilibrium.

NEWTON'S METHOD FOR GRADIENT PROJECTION

There are several ways to perform step 3. Here is one method.

For each OD pair, define the *basic path* to be the shortest path in $\hat{\Pi}^{rs}$. All other paths are *nonbasic*.

For each nonbasic path, perform one step of Newton's method with the basic path to try to equalize their travel times.

Consider any two paths π and π^*. Let Δh be the amount of flow to shift *away* from π and *onto* π^*.

Let $c_\pi(\Delta h)$ and $c_{\pi^*}(\Delta h)$ be the travel times on these paths after shifting Δh flow.

Defining $g(\Delta h) = c_\pi(\Delta h) - c_{\pi^*}(\Delta h)$, a zero of g corresponds to equal travel times on these two paths.

What is $g'(\Delta h)$?

$$g(\Delta h) = \sum_{(i,j)\in\mathcal{A}} (\delta_{ij}^{\pi} - \delta_{ij}^{\pi^*}) t_{ij}(x_{ij}(\Delta h))$$

so

$$g'(\Delta h) = \sum_{(i,j)\in\mathcal{A}} (\delta_{ij}^{\pi} - \delta_{ij}^{\pi^*}) \frac{dt_{ij}}{dx_{ij}} \frac{dx_{ij}}{d\Delta h}$$

For each link in the sum, there are four cases.

Case I : $\delta_{ij}^{\pi} = \delta_{ij}^{\pi^*} = 0$. Then $(\delta_{ij}^{\pi} - \delta_{ij}^{\pi^*}) \frac{dt_{ij}}{dx_{ij}} \frac{dx_{ij}}{d\Delta h} = 0$.

Case II : $\delta_{ij}^{\pi} = \delta_{ij}^{\pi^*} = 1$. Then $(\delta_{ij}^{\pi} - \delta_{ij}^{\pi^*}) \frac{dt_{ij}}{dx_{ij}} \frac{dx_{ij}}{d\Delta h} = 0$

Case III : $\delta_{ij}^{\pi} = 1$ and $\delta_{ij}^{\pi^*} = 0$. Then $(\delta_{ij}^{\pi} - \delta_{ij}^{\pi^*}) \frac{dt_{ij}}{dx_{ij}} \frac{dx_{ij}}{\Delta h} = -\frac{dt_{ij}}{dx_{ij}}$.

Case IV : $\delta_{ij}^{\pi} = 0$ and $\delta_{ij}^{\pi^*} = 1$. Then $(\delta_{ij}^{\pi} - \delta_{ij}^{\pi^*}) \frac{dt_{ij}}{dx_{ij}} \frac{dx_{ij}}{\Delta h} = -\frac{dt_{ij}}{dx_{ij}}$.

In short, the only links contributing to the derivative g' are those which appear in either π or π^*, *but not both.* These are the only links whose flow values will change when we shift travelers from π to π^*.

The derivative can be written

$$g'(\Delta h) = -\sum_{(i,j)\in A_3\cup A_4} \frac{dt_{ij}}{dx_{ij}}$$

where A_3 and A_4 are the sets of links falling into Case III and Case IV on the previous slide. (This is the "gradient" part.)
Then, using the Newton's method formula with an initial guess $\Delta h_0 = 0$, the amount of flow to shift is

$$\Delta h = -g(0)/g'(0) = \frac{c_\pi - c_{\pi^*}}{\sum_{a\in A_3\cup A_4} \frac{dt_{ij}}{dx_{ij}}}$$

We do need to make sure that the flow on π remains nonnegative:

$$\Delta h = \min\left\{ h_\pi, \frac{c_\pi - c_{\pi^*}}{\sum_{a\in A_3\cup A_4} \frac{dt_{ij}}{dx_{ij}}} \right\}$$

(This is the "projection.")

Complete algorithm

A general scheme for path-based algorithms is:

1. Initialize $\hat{\Pi}^{rs} \leftarrow \emptyset$ for all OD pairs.
2. For each OD pair (r, s):
   1. Find the shortest path π^*_{rs}. Add it to $\hat{\Pi}^{rs}$.
   2. If there is only one path π^{rs} in $\hat{\Pi}^{rs}$, set $h^{rs}_{\pi} \leftarrow d^{rs}$. Otherwise, for each non-basic path $\pi^{rs} \neq \pi^*_{rs}$, adjust path flows with

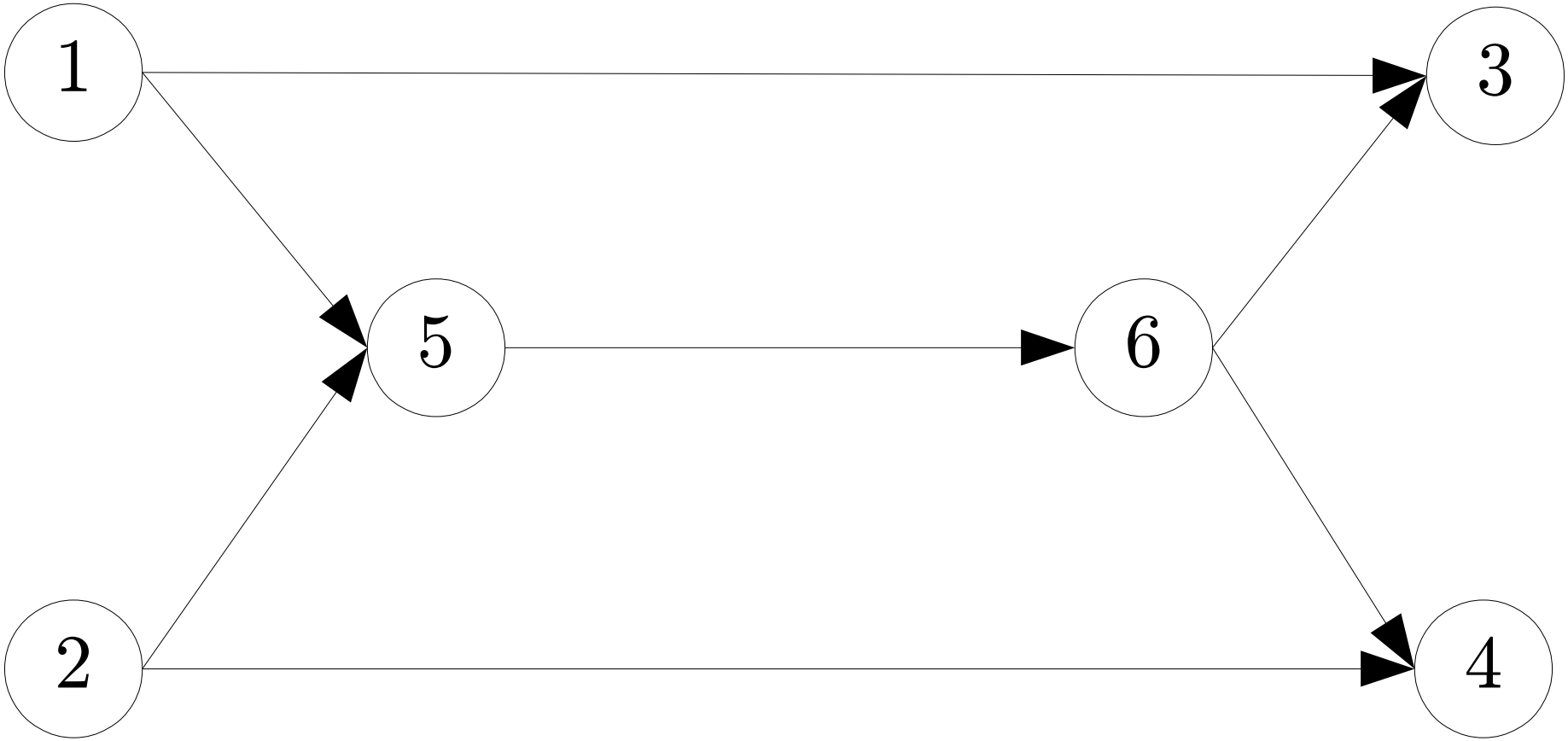
$$h_{\pi^*} \leftarrow h_{\pi^*} + \min \left\{ h_{\pi}, \frac{c_{\pi} - c_{\pi^*}}{\sum_{a \in A_3 \cup A_4} \frac{dt_{ij}}{dx_{ij}}} \right\}$$

   and

$$h_{\pi} \leftarrow h_{\pi} - \min \left\{ h_{\pi}, \frac{c_{\pi} - c_{\pi^*}}{\sum_{a \in A_3 \cup A_4} \frac{dt_{ij}}{dx_{ij}}} \right\}$$

   3. Update travel times
3. Update travel times; drop paths from $\hat{\Pi}^{rs}$ if they are no longer used; check convergence; return to step 2.

Example



Link performance functions are $10 + x/100$, demand is 5000 from 1 to 3, and 10000 from 2 to 4.

If you recall from the Frank-Wolfe section, after three iterations FW method found an AEC of 1.56 minutes. Three iterations of gradient projection produced AEC of 0.17 minutes.

Furthermore, if we were continue further, the relative advantage of gradient projection would only grow.