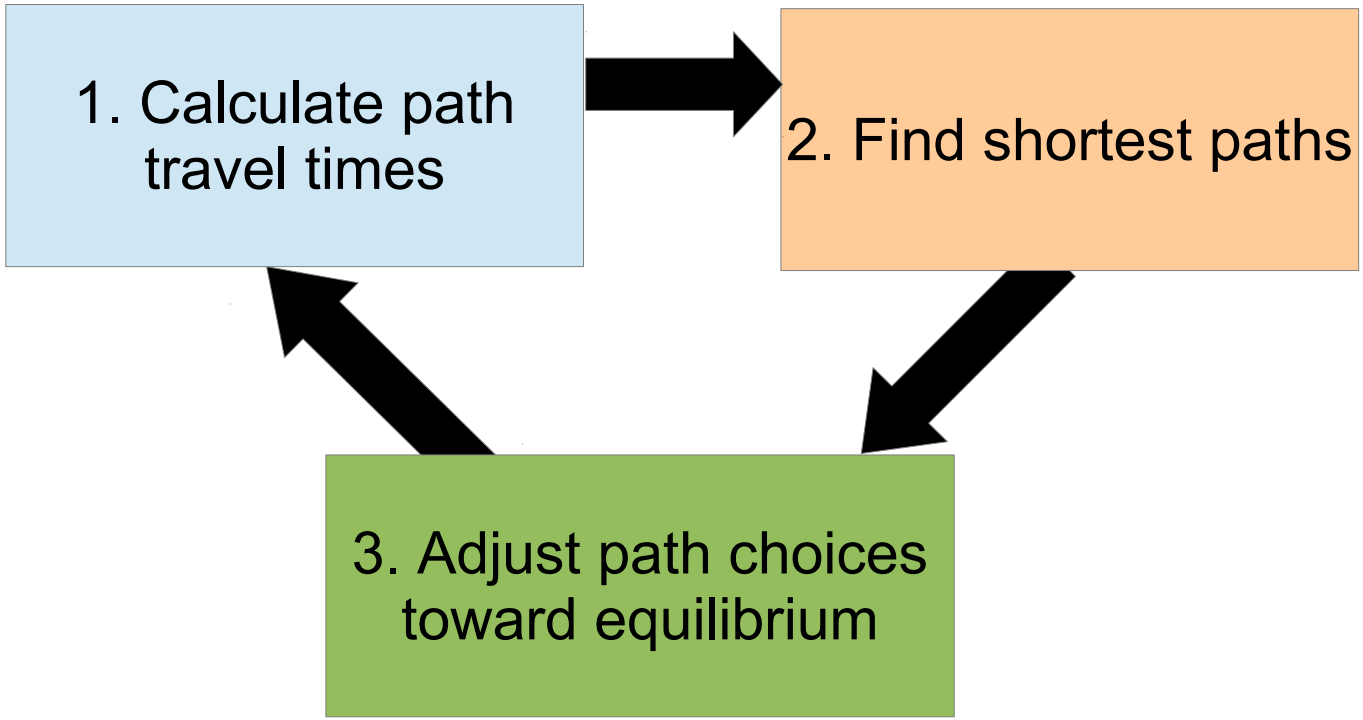
Beckmann's formulation, MSA, and Frank-Wolfe

CE 392C



We're now done with steps 1 and 2; how can we shift flows to shortest paths (step 3)?

OUTLINE

1. System optimal as optimization
2. User equilibrium as optimization
3. Method of successive averages
4. Frank-Wolfe algorithm

SYSTEM OPTIMAL ASSIGNMENT

Can we formulate the system optimum problem as an optimization problem?

What are the objective function, decision variables, and constraints?

There are a few ways to do this; one is include two types of decision variables: the link flows x_{ij}, and the path flows h^{π}.

The objective function is to minimize the total system travel time $\sum_{ij} x_{ij} t_{ij}(x_{ij})$

There are two types of constraints: (1) the path flows must be a feasible assignment; (2) the link flows must be consistent with the path flows:

1. $h^{\pi} \geq 0 \qquad \forall \pi \in \Pi$ (obviously can't have a negative path flow)
2. $\sum_{\pi \in \Pi^{rs}} h^{\pi} = d^{rs} \qquad \forall (r,s) \in Z^2$ (no vehicle left behind)
3. $x_{ij} = \sum_{\pi \in \Pi} \delta_{ij}^{\pi} h^{\pi}$ (link flows consistent with path flows)

Then, the SO optimization problem is

$$\min_{\mathbf{x},\mathbf{h}} \quad \sum_{(i,j)\in A} t_{ij}(x_{ij})x_{ij}$$

$$\text{s.t.} \quad x_{ij} = \sum_{\pi\in\Pi} \delta_{ij}^{\pi} h^{\pi} \qquad \forall (i,j) \in A$$

$$d^{rs} = \sum_{\pi\in\Pi^{rs}} h^{\pi} \qquad \forall (r,s) \in Z^2$$

$$h^{\pi} \geq 0 \qquad \forall \pi \in \Pi$$

USER EQUILIBRIUM ASSIGNMENT

What should the objective be for user equilibrium assignment?

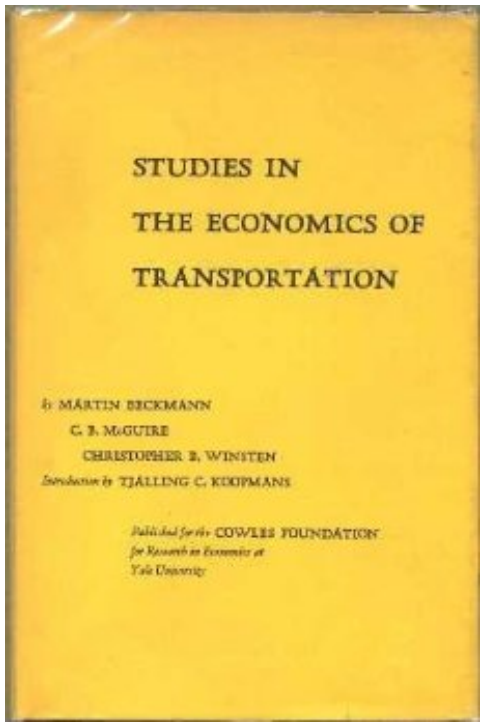
The answer isn't obvious; it turns out that the *Beckmann function*

$$\sum_{(i,j)\in A} \int_0^{x_{ij}} t_{ij}(x)dx$$

is the appropriate one.

To see why, we need to look at the optimality conditions using this objective function.

This formulation comes from the seminal book *Studies in the Economics of Transportation* by Martin Beckmann, C. B. McGuire, and Christopher Winsten.



This book was published in 1956 and was the starting point for transportation network analysis as used today.

Consider the optimization problem

$$\min_{\mathbf{x},\mathbf{h}} \quad \sum_{(i,j)\in A} \int_0^{x_{ij}} t_{ij}(x)dx$$

$$\text{s.t.} \quad x_{ij} = \sum_{\pi\in\Pi} \delta_{ij}^{\pi} h^{\pi} \qquad \forall (i,j) \in A$$

$$d^{rs} = \sum_{\pi\in\Pi^{rs}} h^{\pi} \qquad \forall (r,s) \in Z^2$$

$$h^{\pi} \geq 0 \qquad \forall \pi \in \Pi$$

What are the optimality conditions?

After some simplification, these reduce to

$$h^{\pi} \geq 0 \qquad \forall \pi \in \Pi$$
$$c^{\pi} \geq \kappa_{rs} \qquad \forall (r,s) \in Z^2$$
$$h^{\pi}(c^{\pi} - \kappa_{rs}) = 0 \qquad \forall \pi \in \Pi$$

where the κ_{rs} in the third constraint corresponds to the OD pair connected by π.

The first condition is nonnegative path flows.

The second condition shows that κ_{rs} is *the shortest path travel time between r and s*

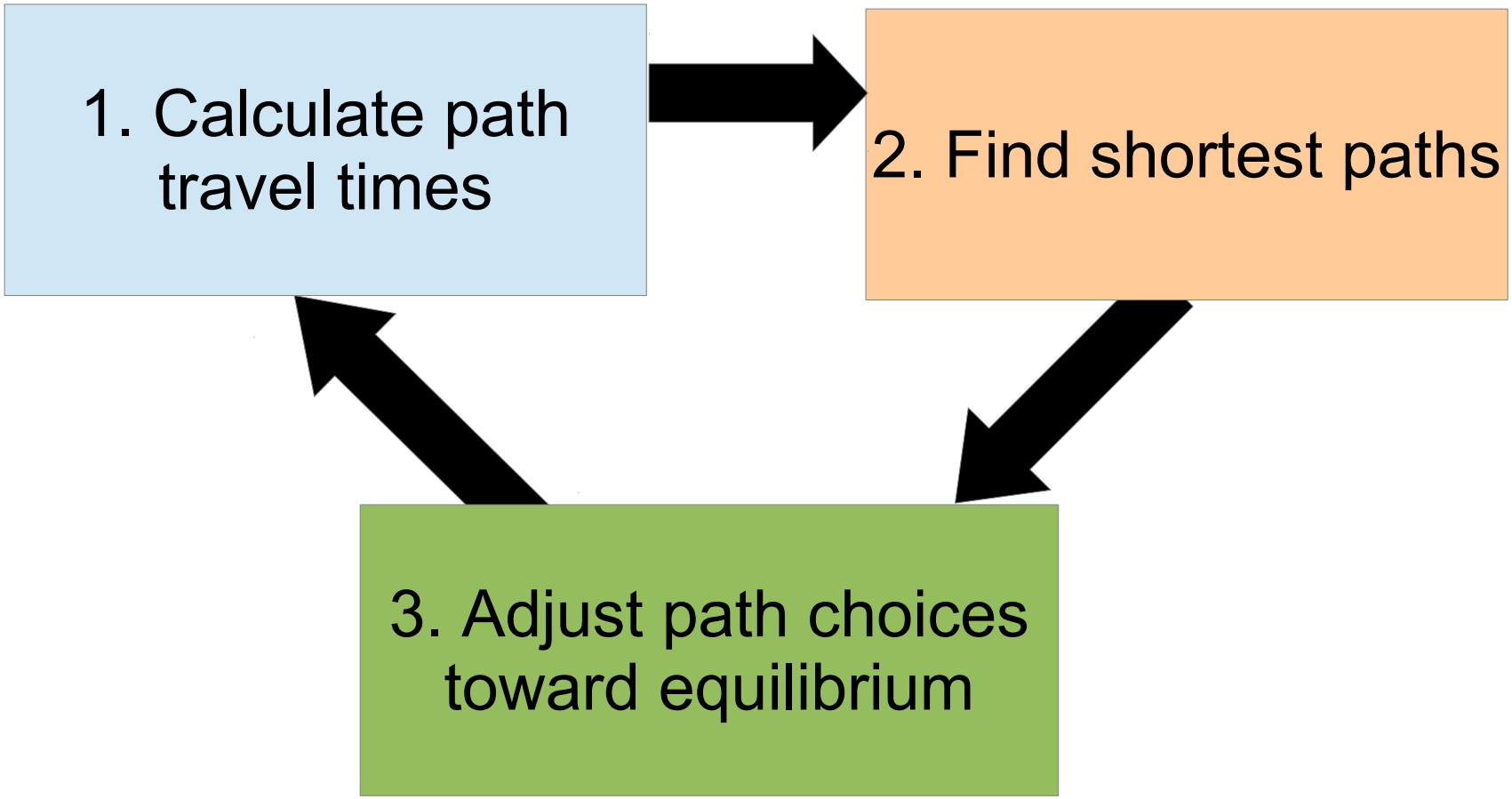
The third condition shows that if a path is used ($h^{\pi} > 0$) its travel time must be equal to κ_{rs}.

This is a case where the objective function was determined by **working backward from the optimality conditions.**

Furthermore, the Beckmann function is strictly convex in the link flows $\mathbf{x}$ if the link performance functions are increasing.

Therefore, **the user equilibrium solution is unique in link flows.**

ITERATIVE FRAMEWORK


1. Calculate path travel times
2. Find shortest paths
3. Adjust path choices toward equilibrium

So, this suggests one specific implementation of the iterative framework:

1. Start with some feasible link flow solution $\mathbf{x}$. (Working with $\mathbf{h}$ is too unwieldy for large networks.)
2. Calculate the link travel times using the flows $\mathbf{x}$.
3. Find the shortest paths between all origins and destinations
4. Find the all-or-nothing link flows $\mathbf{x}^*$ corresponding to these shortest paths.
5. Choose $\lambda \in [0, 1]$ and update $\mathbf{x} \leftarrow \lambda \mathbf{x}^* + (1 - \lambda)\mathbf{x}$
6. If "close enough to equilibrium" stop, otherwise return to step 2.

There are two things in things in this algorithm which should look a little bit "fuzzy" to you.

- How do we know when we are "close enough to equilibrium?"
- How do we choose λ?

We'll tackle this questions in order.

TERMINATION CRITERIA

Can we stop when the link flows aren't changing much anymore?

It's always best to measure convergence *based on consistency with the equilibrium condition*.

Most of these ideas compare two values: the total system travel time *TSTT* and the *shortest path travel time SPTT*

We've already defined *TSTT* as $\sum_{ij} t_{ij}(x_{ij})x_{ij}$. *SPTT* can be defined as $\sum_{ij} t_{ij}(x_{ij})x^*_{ij}$ where $\mathbf{x}^*$ is an assignment of all vehicles to shortest paths using the travel times $t_{ij}(x_{ij})$.

SPTT can also be written as $\sum_{(r,s)\in Z^2} d^{rs}\kappa^{rs}$.

The key point: at an equilibrium solution, $TSTT = SPTT$, and at a nonequilibrium solution $TSTT > SPTT$. (This follows from the variational inequality formulation.) So, the "gap" between these tells us how close we are

The most common gap measure in use today is the *relative gap*

$$\gamma = \frac{TSTT}{SPTT} - 1$$

(Unfortunately, the term has been given slightly different meanings by different authors, and it's hard to describe what the relative gap means in real-world terms.)

The *average excess cost* is

$$AEC = \frac{TSTT - SPTT}{\sum_{rs} d^{rs}}$$

that is, a normalized gap measure showing how much longer the average vehicle's trip is than the shortest path available.

METHOD OF SUCCESSIVE AVERAGES

How do we choose λ?

There are two ways to go wrong. If λ is "too big", then we are overcorrecting (and may oscillate endlessly).

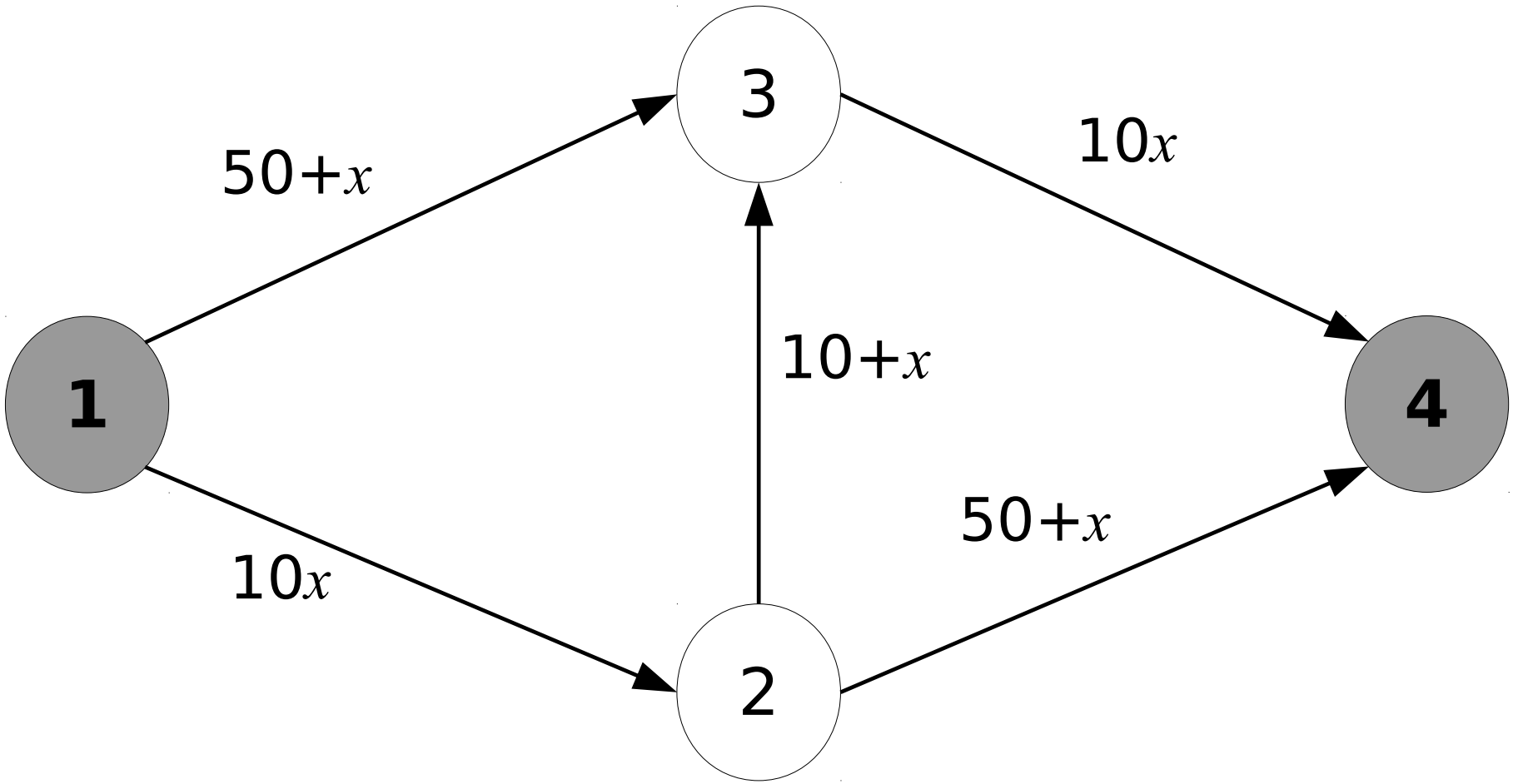
If λ is "too small", then it will take a very long time to finish (if at all).

The *method of successive averages* tries to prevent both problems by starting with large λ values and moving to smaller ones.

If λ_i is the step size for the i-th iteration, $\{\lambda_i\} = \{1/2, 1/3, 1/4, 1/5, \ldots\}$.

We can choose other patterns for the step sizes, but will require $\sum \lambda_i = \infty$ and $\sum \lambda_i^2 < \infty$. (Why?)

Example



FRANK-WOLFE

The Frank-Wolfe algorithm tries to choose λ more intelligently: at each iteration, λ is chosen to *get as close to equilibrium as possible* along the line connecting $\mathbf{x}$ to $\mathbf{x}^*$.

This is done by solving a "restricted" VI where the feasible set X is the line segment between $\mathbf{x}$ and $\mathbf{x}^*$ and the force points in the direction of $-\mathbf{t}(\mathbf{x})$.

That is, X consists of all vectors of the form $\lambda\mathbf{x} + (1-\lambda)\mathbf{x}^*$ for all $\lambda \in [0, 1]$.

The VI is: find $\hat{\mathbf{x}}$ such that $\mathbf{t}(\hat{\mathbf{x}}) \cdot (\hat{\mathbf{x}} - \mathbf{x}') \leq 0$ for all $\mathbf{x}' \in X$.

How does this VI in the "link-flows" $\mathbf{x}$ relate to the VI we previously saw in the path flows $\mathbf{h}$?

They are actually the same, and the proof is very simple using the matrix version of the link-path relationships. Remember $\mathbf{x} = \Delta\mathbf{h}$ and $\mathbf{c} = \Delta^T\mathbf{t}$, and also remember that an inner product $\mathbf{u} \cdot \mathbf{v}$ can also be written as as the matrix product $\mathbf{u}^T\mathbf{v}$.

$$\begin{aligned}
\mathbf{c}(\mathbf{h}^*) \cdot (\mathbf{h}^* - \mathbf{h}) &= \mathbf{c}(\mathbf{h}^*)^T(\mathbf{h}^* - \mathbf{h}) \\
&= (\Delta^T\mathbf{t}(\mathbf{x}^*))^T(\mathbf{h}^* - \mathbf{h}) \\
&= (\mathbf{t}(\mathbf{x}^*))^T(\Delta\mathbf{h}^* - \Delta\mathbf{h}) \\
&= \mathbf{t}(\mathbf{x}^*) \cdot (\mathbf{x}^* - \mathbf{x})
\end{aligned}$$

What does the solution to this VI look like? The set X has two endpoints; for now assume that the solution to the VI is not at one of these points. (Is this a safe assumption?)

Then at the solution to the VI $\hat{\mathbf{x}}$, the force vector $-\mathbf{t}(\hat{\mathbf{x}})$ is perpendicular to the direction $\mathbf{x}^* - \mathbf{x}$.

That is, $-\mathbf{t}(\hat{\mathbf{x}}) \cdot (\mathbf{x}^* - \mathbf{x}) = 0$.

Writing this out in terms of individual components, the solution happens if $\sum_{ij} t_{ij}(\hat{x}_{ij}) \left(x^*_{ij} - x_{ij} \right) = 0$ or equivalently

$$\sum_{ij} t_{ij}(\hat{x}_{ij}) x^*_{ij} = \sum_{ij} t_{ij}(\hat{x}_{ij}) x_{ij}$$

You can think of $\hat{\mathbf{x}}$ as a "balance point" between the current solution $\mathbf{x}$ and the target solution $\mathbf{x}^*$: if the travel times were based on the flows $\hat{\mathbf{x}}$, the points $\mathbf{x}$ and $\mathbf{x}^*$ look equally attractive.

How can we find this balance point?

It is also possible to show that the value of λ solving the restricted variational inequality also minimizes the Beckmann function along the line segment between $\mathbf{x}$ and $\mathbf{x}^*$.

Write $f(\mathbf{x}(\lambda))$ to denote the value of the Beckmann function at the new link flows corresponding to λ.

We can show that

$$\frac{df}{d\lambda} = \sum_{(i,j)\in A} t_{ij}(\hat{x}_{ij})(x^*_{ij} - x^0_{ij})$$

which equals zero if

$$\sum_{ij} t_{ij}(\hat{x}_{ij})x^*_{ij} = \sum_{ij} t_{ij}(\hat{x}_{ij})x_{ij}$$

Method 1: Bisection

1. Set the boundaries on λ: $\lambda_{lo} = 0$, $\lambda_{hi} = 1$
2. Pick $\lambda \leftarrow (\lambda_{lo} + \lambda_{hi})/2$ as the midpoint.
3. Evaluate the equation at the midpoint:
   1. Let $\hat{\mathbf{x}} \leftarrow \lambda \mathbf{x}^* + (1 - \lambda)\mathbf{x}$
   2. Then the equation is $\sum_{ij} t_{ij}(\hat{x}_{ij}) \left(x^*_{ij} - x_{ij}\right)$
4. If the equation is negative, set $\lambda_{lo} \leftarrow \lambda$.
5. If the equation is positive, set $\lambda_{hi} \leftarrow \lambda$.
6. If λ_{hi} and λ_{lo} are sufficiently close (or if the equation is sufficiently close to zero), terminate. Otherwise return to step 2.

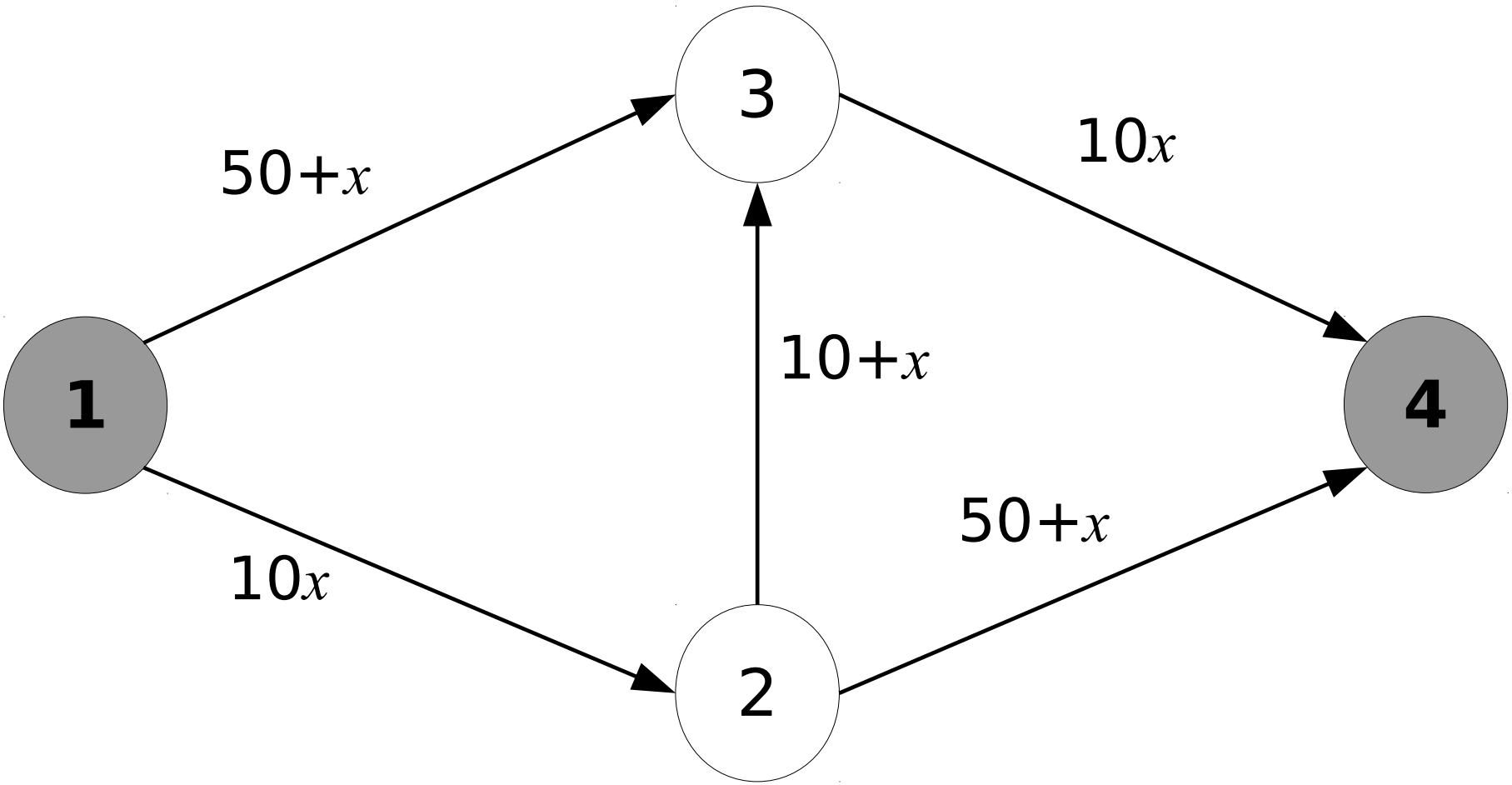
Method 2: Newton's method

1. Choose some initial value of λ
2. Evaluate the equation and its derivative with respect to λ at the midpoint:
   1. Let $\hat{\mathbf{x}} \leftarrow \lambda \mathbf{x}^* + (1-\lambda)\mathbf{x}$
   2. Let $f = \sum_{ij} t_{ij}(\hat{x_{ij}}) \left(x^*_{ij} - x_{ij}\right)$
   3. Let $f' = \sum_{ij} t'_{ij}(\hat{x_{ij}}) \left(x^*_{ij} - x_{ij}\right)^2$
3. Update $\lambda \leftarrow \lambda - f/f'$
4. If λ falls outside of $[0,1]$ "project" it onto that set.
5. If the equation is sufficiently close to zero (or we have hit the same endpoint twice in a row), terminate. Otherwise return to step 2.

Do we have to worry about dividing by zero?

Newton's method usually requires fewer iterations, but requires more calculation at each iteration. Both of them are easy to program.

Example



If you're solving by hand, the Frank-Wolfe method can be a bit tedious. However, with the help of a spreadsheet or some simple code, you can automate the tedious parts.