1.204 Lecture 18

Continuous constrained nonlinear optimization:

Convex combinations 1:

Network equilibrium

Transportation network flows

- Amount of travel on any road or transit line is result of many individuals' decisions
  - These depend on price and quality of service
  - Congestion in urban areas is a significant factor
- Analyzing passenger flows on networks relies on:
  - Graph data structures
  - Shortest path algorithms
  - Network assignment algorithms that assign travelers to a particular set of streets or transit lines, based on travel time, cost and other service measures
  - Demand models are also used
    - Based on discrete choice theory (take 1.202!)

Transportation network equilibrium

- Users make their own, 'selfish' decisions on the best path through a network
  - When congestion exists, traveler choices affect travel times, which in turn affect traveler choices, which…
  - Users switch routes (and modes and time of day and trip frequency and location) in response to changes in service quality
  - We model this as a market that reaches supply-demand equilibrium on every arc in a network

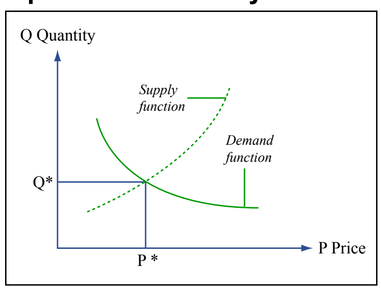

Figure by MIT OpenCourseWare.

Figures from Sheffi

Definition of equilibrium

- Links (including intersections) have a supply function:

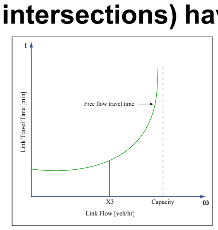

Figure by MIT OpenCourseWare.

- Definition of equilibrium:
  - For each origin-destination pair:
    - Travel time for all used paths is equal, and is
    - Less than (or equal to) the travel time on any unused path

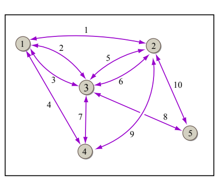
Figure by MIT OpenCourseWare.

(Transit is messier, because it has a route structure as well as a network structure, but the same principles apply)

Network equilibrium problem formulation

$$\min z(x) = \sum_{arcs\ a} \int_0^{x_a} t_a(\omega)\, d\omega$$

subject to

$$\sum_{paths\ k} f_k^{rs} = q_{rs} \qquad \forall OD\ pairs\ r,s$$

$$f_k^{rs} \geq 0 \qquad \forall k,r,s$$

$$x_a = \sum_i \sum_j \sum_k f_k^{rs} \qquad \forall r,s,k$$

if a on path from r to s

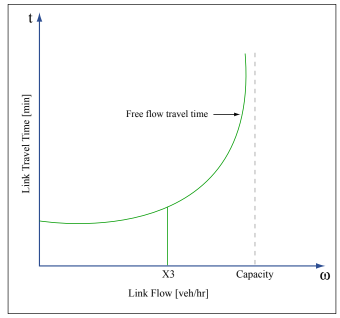

Figure by MIT OpenCourseWare.

Network equilibrium problem example

$$t_1 = 2 + x_1$$
$$t_2 = 1 + 2x_2$$
$$x_1 + x_2 = 5$$

Equilibrium conditions:

$$t_1 = t_2 \qquad (both\ routes\ used)$$

Solution, by inspection:

$$x_1 = 3$$
$$x_2 = 2$$
$$t_1 = t_2 = 5$$

$2+x_1= 1+2(5-x_1)$
$3x_1= 11-2$
$x_1= 3$

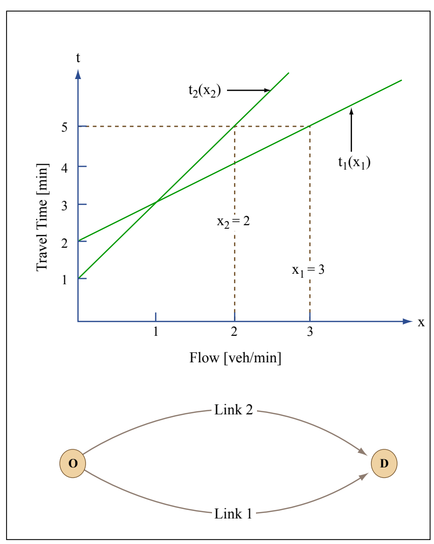

Figure by MIT OpenCourseWare.

Formulation example

$$\min z(x) = \int_0^{x_1} (2+\omega)d\omega + \int_0^{x_2} (1+2\omega)d\omega$$

$s.t.$

$x_1 + x_2 = 5$

$x_1 \geq 0, x_2 \geq 0$

Convert to $1-D$ *by setting* $x_2 = 5 - x_1$

$$\min z(x) = \int_0^{x_1} (2+\omega)d\omega + \int_0^{5-x_1} (1+2\omega)d\omega$$

$s.t.$

$x_1 \geq 0, 5 - x_1 \geq 0$

Integrate analytically :

$z(x) = 1.5x_1^2 - 9x_1 + 30$

$$\frac{dz(x_1)}{dx_1} = 0 \Rightarrow x_1 = 3$$

$z(x) = 2x_1 + x_1^2/2 + (5-x_1) + (5-x_1)^2$

$= 2x_1 + 0.5x_1^2 + 5 - x_1 + 25 - 10x_1 + x_1^2$

Formulation

- **The formulation has no economic or physical significance**
  - **It happens to produce the desired first-order conditions for an optimum**
  - **They require that the time on all routes used between an origin and destination be equal**
  - **And the time on routes not used must be greater**
  - **We can view the objective function as a convergence criterion for the equilibrium solution**
- **Nonetheless, equilibrium is a key concept**
  - **And it's a nonlinear optimization problem, techniques for which we want to cover in this course**
  - **This is a constrained continuous nonlinear optimization problem**

Solution method: convex combinations

$$\min z(x)$$
$$s.t.$$
$$\sum_i h_{ij} x_i \ge b_j \quad \forall j$$

Assume current solution is $x^n = (x_1^n, x_2^n, ..., x_I^n)$

To find descent direction, we wish to find auxiliary feasible solution $y^n = (y_1^n, y_2^n, ..., y_I^n)$ *so direction from* x^n *to* y *gives maximum decrease.*

Direction from x^n *to* y *is unit vector* $(y - x^n) / \| y - x^n \|$

$(\| v \|$ *means* $\sqrt{v \cdot v})$

Slope of $z(x^n)$ *in direction of* $(y - x^n) =$

$$-\nabla z(x^n) \cdot \frac{(y - x^n)^T}{\| y - x^n \|} \qquad where\ \nabla z(x^n) = \left(\frac{\partial z(x)}{\partial x_1}, \frac{\partial z(x)}{\partial x_2}, ..., \frac{\partial z(x)}{\partial x_j}\right)$$

Solution method: convex combinations 2

Rewrite original problem as linear approximation:

$$\min z_L^n(y) = z(x^n) + \nabla z(x^n) \cdot (y - x^n)^T$$
$$s.t.$$
$$\sum_i h_{ij} y_i \ge b_j$$

At $x = x^n$ *value of objective function is constant: we can drop* $z(x^n)$

Also $\nabla z(x^n)$ *is constant at* $x = x^n$*, so we can drop it, leaving:*

$$\min z_L^n(y) = \nabla z(x^n) \cdot y^T = \sum_i \frac{\partial z(x^n)}{\partial x_i} \cdot y_i$$
$$s.t.$$
$$\sum_i h_{ij} y_i \ge b_j$$

This is a linear program whose solution is y.

It gives a descent direction $(y^n - x^n)$

Solution method: convex combinations 3

To determine how far to go in this direction:

$$\min z[x^n + \alpha(y^n - x^n)]$$

s.t.

$$0 \le \alpha \le 1$$

This is a 1-D minimization problem in α, solved with a line search using bisection

Once α is found, next point generated by:

$$x^{n+1} = x^n + \alpha_n(y^n - x^n)$$

The new solution is a weighted average, or convex combination of x^n and y^n

Continue until convergence, which is slow but guaranteed

Convex combinations algorithm

- **Step 1: Direction finding. Find y^n that solves linear program.**

$$\min z_L^n(y) = \sum_i \frac{\partial z(x^n)}{\partial x_i} \cdot y_i \quad s.t. \quad \sum_i h_{ij} y_i \ge b_j$$

- **Step 2: Step size determination, or line search. Find α_n that solves**

$$\min z[x^n + \alpha(y^n - x^n)]$$

- **Step 3: Move. Set**

$$x^{n+1} = x^n + \alpha_n(y^n - x^n)$$

- **Step 4: Convergence test. If $z(x^n) - z(x^{n-1}) < K$, stop**

Next time

- We'll apply the convex combinations method to the network equilibrium problem
  - Formulation
  - Algorithms
    - Direction finding
      - Shortest path algorithm solves the linear program
      - Compute y flow vector (auxiliary solution)
    - Line search
      - Bisection solves the line search problem
      - Must compute derivative of objective function
    - Move
      - Update x flows on network as linear combination of x and y flows
      - Update arc travel times; both of these steps are just algebra
    - Convergence test
      - Compute change in flows as simplest measure
  - Java implementation