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

Definition of equilibrium

- Links (including intersections) have a supply function:

- 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

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

\[
\min z(x) = \sum_{\omega \in \omega} t_{\omega}(x) \, d\omega
\]

subject to

\[
\sum_{\text{paths } k} f_{rs}^k = q_{rs} \quad \forall \text{OD pairs } r, s
\]

\[
f_{rs}^k \geq 0 \quad \forall k, r, s
\]

\[
x_a = \sum_i \sum_j \sum_k f_{rs}^k \quad \forall r, s, k
\]

if \( a \) on path from \( r \) to \( s \)

Network equilibrium problem example

\[
t_1 = 2 + x_1 \\
t_2 = 1 + 2x_2 \\
x_1 + x_2 = 5
\]

Equilibrium conditions:
\( t_1 = t_2 \) (both routes used)

Solution, by inspection:
\( x_1 = 3 \)
\( x_2 = 2 \)
\( t_1 = t_2 = 5 \)

\[
2x_1 = 1 + 2(5-x_1) \\
3x_1 = 11 - 2 \\
x_1 = 3
\]
**Formulation example**

\[
\begin{align*}
\min z(x) &= \int_0^y (2 + \omega)d\omega + \int_0^z (1 + 2\omega)d\omega \\
\text{s.t.} & \\
x_1 + x_2 &= 5 \\
x_i &\geq 0, x_2 \geq 0 \\
\text{Convert to } 1 - D \text{ by setting } x_2 = 5 - x_i \\
\min z(x) &= \int_0^y (2 + \omega)d\omega + \int_0^{5-x_i} (1 + 2\omega)d\omega \\
\text{s.t.} & \\
x_i &\geq 0, 5 - x_i \geq 0 \\
\text{Integrate analytically:} & \\
z(x) &= 1.5x_i^2 - 9x_i + 30 \\
\frac{dz(x_i)}{dx_i} &= 0 \Rightarrow x_i = 3
\end{align*}
\]

**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

\[
\begin{align*}
\min z(x) \\
s.t. \\
\sum_i h_j x_i & \geq b_j \quad \forall j \\
\text{Assume current solution is } x^n = (x_1^n, x_2^n, ..., x_i^n) \\
\text{To find descent direction, we wish to find auxiliary feasible solution } y^n = (y_1^n, y_2^n, ..., y_i^n) \text{ so direction from } x^n \text{ to } y^n \text{ gives maximum decrease.} \\
\text{Direction from } x^n \text{ to } y \text{ is unit vector } (y - x^n) / \| y - x^n \| \\
(\| v \| \text{ means } \sqrt{v \cdot v}) \\
\text{Slope of } z(x^n) \text{ in direction of } (y - x^n) = \\
- \nabla z(x^n) \cdot \frac{(y - x^n)^T}{\| y - x^n \|} \\
\text{where } \nabla z(x^n) = \left( \frac{\partial z(x)}{\partial x_1}, \frac{\partial z(x)}{\partial x_2}, ..., \frac{\partial z(x)}{\partial x_j} \right)
\end{align*}
\]

Solution method: convex combinations 2

\[
\text{Rewrite original problem as linear approximation:} \\
\min z^*_I(y) = z(x^n) + \nabla z(x^n) \cdot (y - x^n)^T \\
s.t. \\
\sum_i h_j y_i & \geq b_j \\
\text{At } x = x^n \text{ value of objective function is constant: we can drop } z(x^n) \\
\text{Also } \nabla z(x^n) \text{ is constant at } x = x^n, \text{ so we can drop it, leaving:} \\
\min z^*_I(y) = \nabla z(x^n) \cdot y^T = \sum_i \frac{\partial z(x^n)}{\partial x_i} \cdot y_i \\
s.t. \\
\sum_i h_j y_i & \geq b_j \\
\text{This is a linear program whose solution is } y. \\
\text{It gives a descent direction } (y^n - x^n)
\]
Solution method: convex combinations 3

To determine how far to go in this direction:
\[
\min z[x^n + \alpha (y^n - x^n)]
\]
\[
s.t.
\]
\[
0 \leq \alpha \leq 1
\]
This is a 1-D minimization problem in \(\alpha\), solved with a line search using bisection.

Once \(\alpha\) is found, the 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.
  \[
  \min z^n_i(y) = \sum_i \frac{\partial z(x^n)}{\partial x_i} y_i \quad s.t. \quad \sum_j h_j y_j \geq b_j
  \]

- **Step 2:** Step size determination, or line search. Find \(\alpha_n\) 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