Congestion Games

Nick Andrews

March 13, 2021

Congestion Model

- Congestion model: $C(N, M, (\Sigma^i)_{i \in N}, (c_j)_{j \in M}$
- N = number of players (drivers, users, etc.)
- M = set of facilities (road segments, network paths, etc.)
- Σ^i = set of possible strategies for player i, where each $A^i \in \Sigma^i$ is a nonempty set of facilities (e.g. a route)
- $c_j(k)$ = cost (travel time, etc.) to each user of facility j, if there are exactly k users
- Define payoff function $v_i : \Sigma \to R$ by

$$v^i(A) = \sum_{j \in A^i} c_j(\sigma_j(A))$$

- $\sigma_j(A)$ = number of users of facility j

Congestion Game

Summary

Each player's cost for a given facility is a function of the number of other players that choose the same facility. A player's *total cost* for a chosen strategy is the sum of costs over all chosen facilities in the player's strategy.

Applications

Can be used to model network flow and routing problems in which players have to compete over the same set of shared resources.

Motivation: Braess' Paradox

- 4000 players can travel from start to end with 2 routes: start-A-end or start-B-end
- Players split routes 50-50 and leads to equilibrium with travel time of 65min
- New route built from A to B with travel time of 0min
- More players are inclined to take shortcut route and come to new equilibrium of 80min
- Hence the paradox: adding more routes can increase equilibrium cost

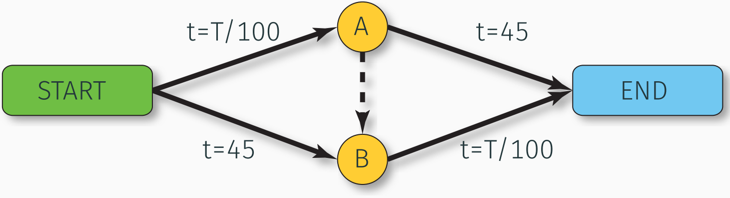

Figure 1: Braess' Network (click for source)

Code Development

Goal

Develop a modular and scalable tool for simulating and finding equilibrium of congestion games

- Built in Python
- User specifies nodes, connections, edge cost functions, and number of players
- Results were verified and tested against examples found online
- Equilibrium is calculated by fixing strategies of all players and letting each player play their best response one at a time
- Best response is minimum cost path from start to end
- How to calculate minimum cost path? Not too computationally expensive to look at all possible paths in Braess' paradox example, but what if network looks like this...?

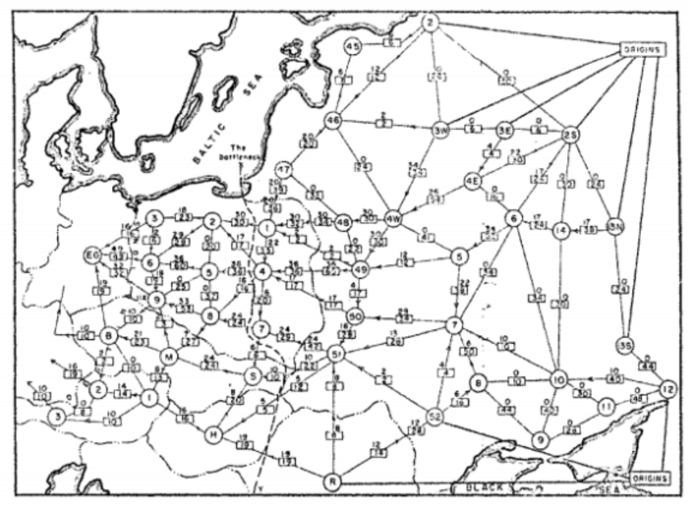

Figure 2: Railway network studied by Russian mathematician A.N. Tolstoi in the 1930's (click for source)

Dijkstra’s Algorithm

- Algorithm for determining shortest path between nodes in a network without calculating all possible combinations
- Pseudocode
    - Calculate total distance of neighboring nodes to start node
    - Pick node with shortest total distance
    - Stop if selected node is end node, else repeat
- Click for link to Wikipedia Dijkstra animation
- Click for link to PathFinding.js interactive demo

Demo

Lessons Learned

- Possible for game to reach ”cost plateau” where adding additional players doesn’t change equilibrium cost
- Can be more complicated than it initially seems to determine best response
- A lot of work goes in to building game framework and representative cost functions
- Could possibly build cost functions by fitting real world data
- Thought it was interesting how routes change with number of players; less traveled edges become relevant

Future Work

- Look at real world network with cost functions that also vary temporally (SpaceX Starlink constellation)
- Look at game where players have different start and end destinations
- Make code cleaner and faster
- Code is available on GitHub at *https://github.com/nian6018/Congestion-Game.git*