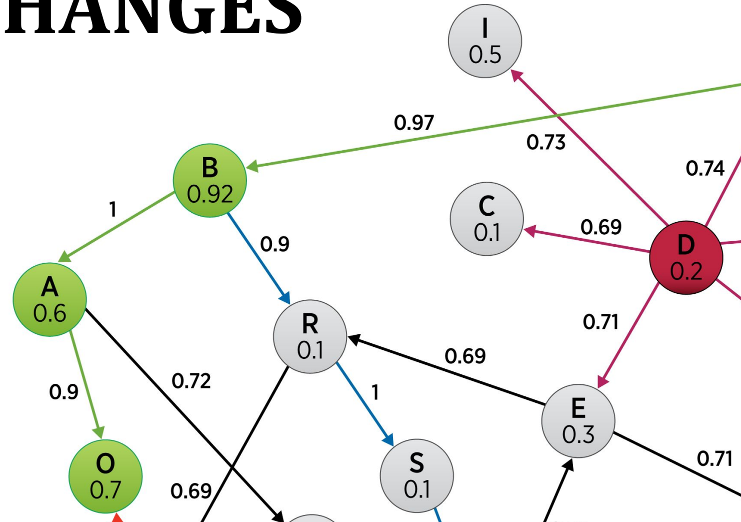
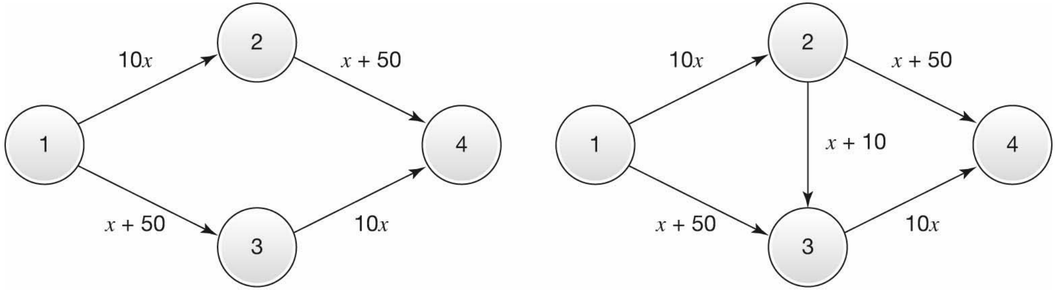
DEFEATING BRAESS' PARADOX BY USING DIJKSTRA ALGORITHM TO EVALUATE ROAD INFRASTRUCTURE CHANGES

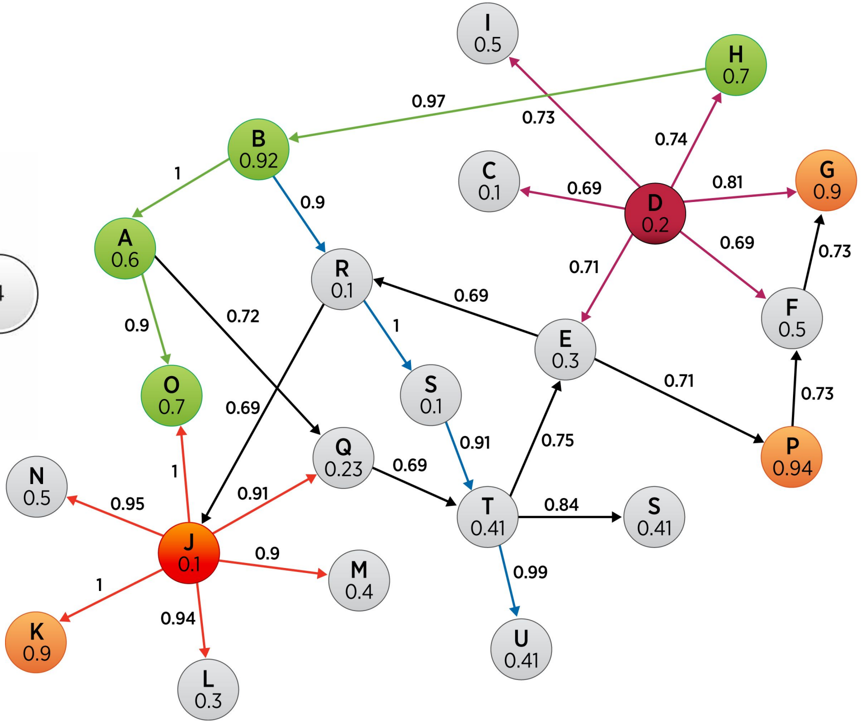
Hayat AlHassan
Peter Chen
Theodore Tenev



The Problem: **Braess' Paradox**

Building a new road between two destinations with the intention to alleviate traffic can in fact increasing overall journey time!





German mathematician Dietrich Braess identified the paradox for the 1st time in 1968.

The Solution: **Network Traffic Simulation**

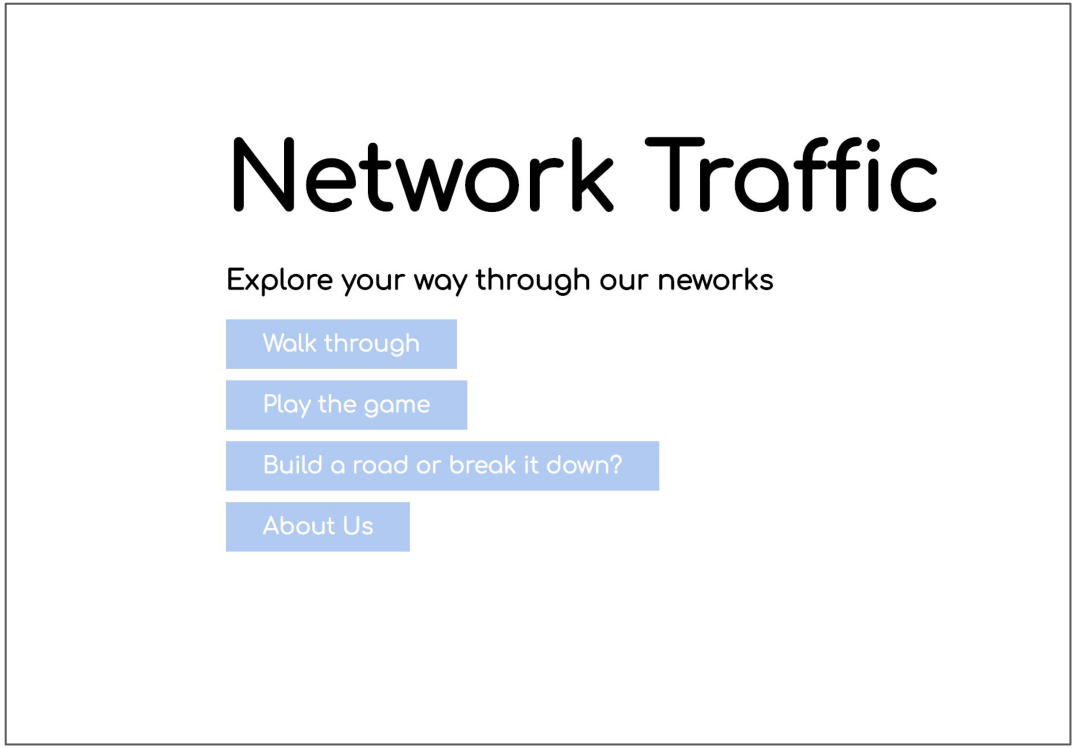
Simple

Interactive

Graphical

Efficient

Road network simulation

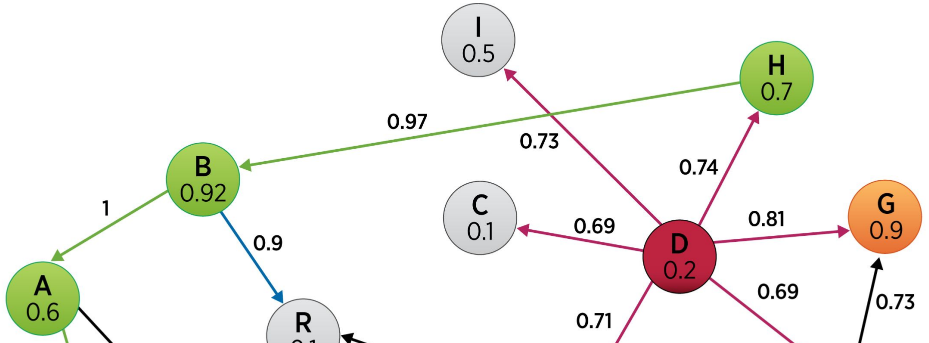

Network Traffic
Explore your way through our neworks
Walk through
Play the game
Build a road or break it down?
About Us

The Approach:

The simulation provides the user with a visual representation of the road network.

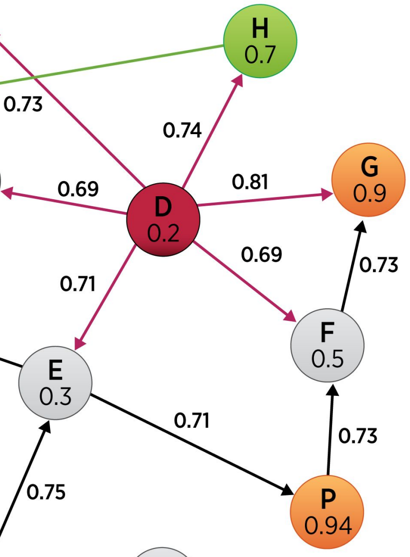
Given the amount of traffic on each road on the network, upon constructing a new road, our simulation will be able to confirm or deny whether this new road will reduce overall journey time.

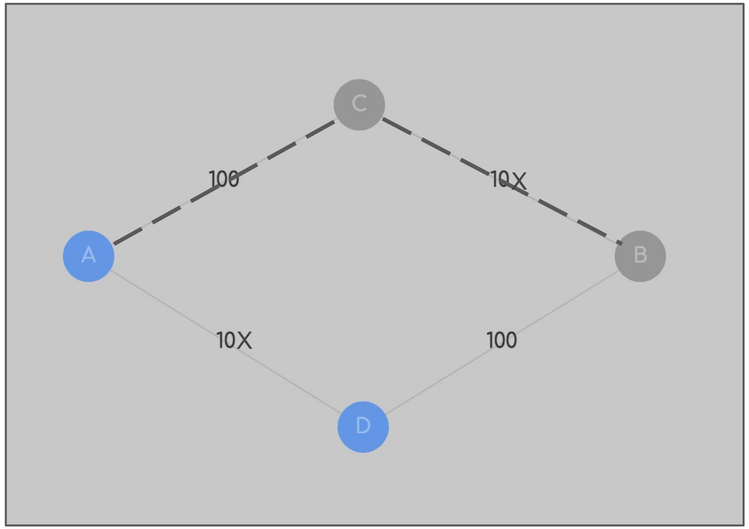
Assumption: All players take the most optimal path for them, ie. the shortest path!



The Approach:

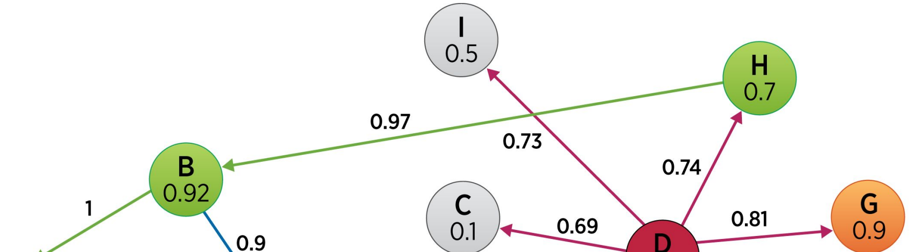
1. Calculate the shortest path given a beginning point and an ending point for the network (using **Dijkstra Algorithm**).

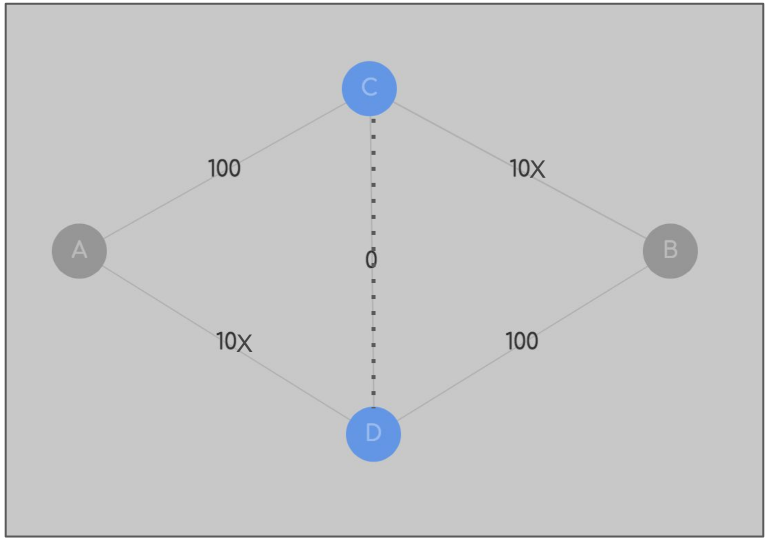




The Approach:

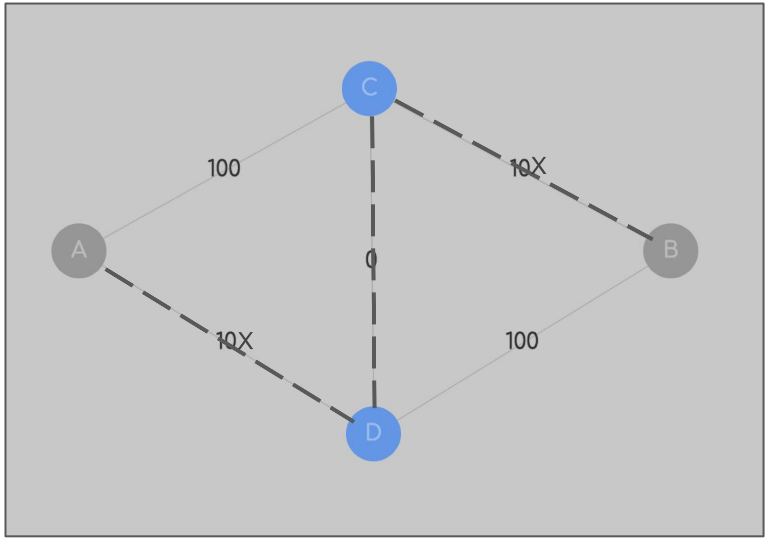
2. Make a change in the road network (create a new connected node, or a new edge between existing nodes).

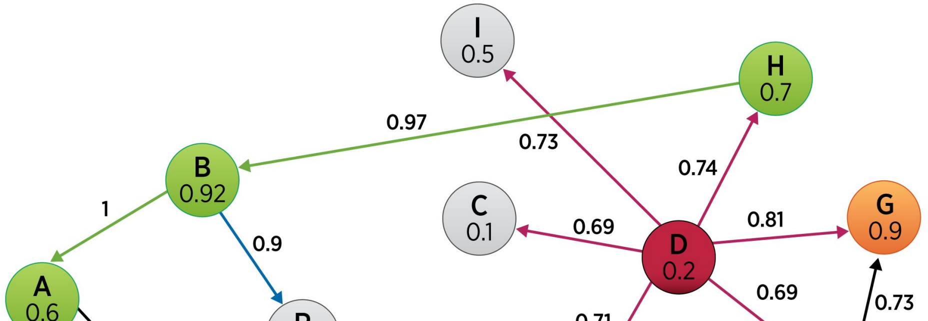




The Approach:

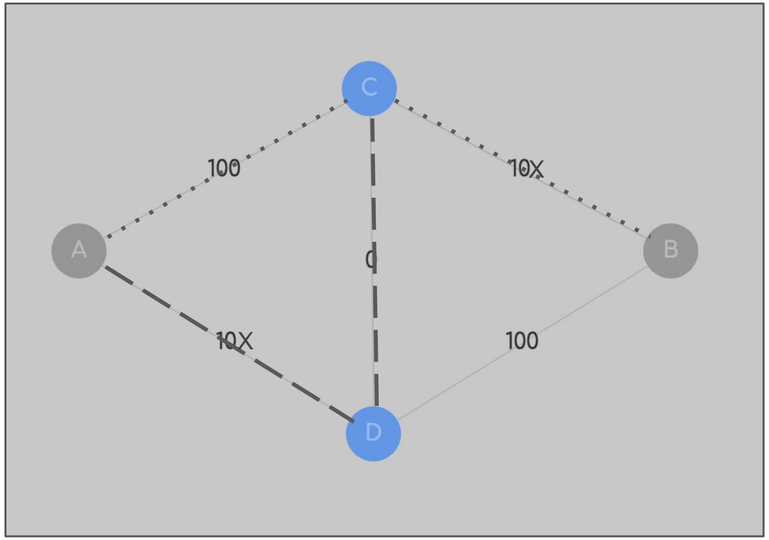
3. Calculate the *new* shortest path given a beginning point and an ending point.

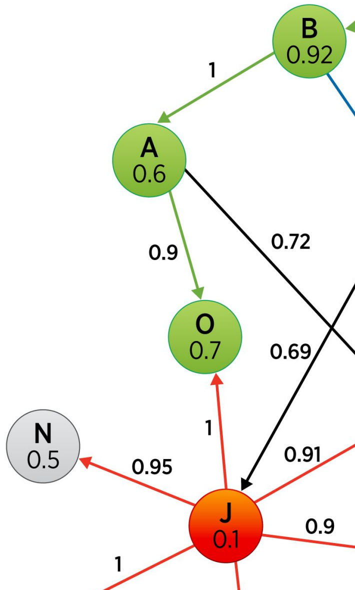




The Approach:

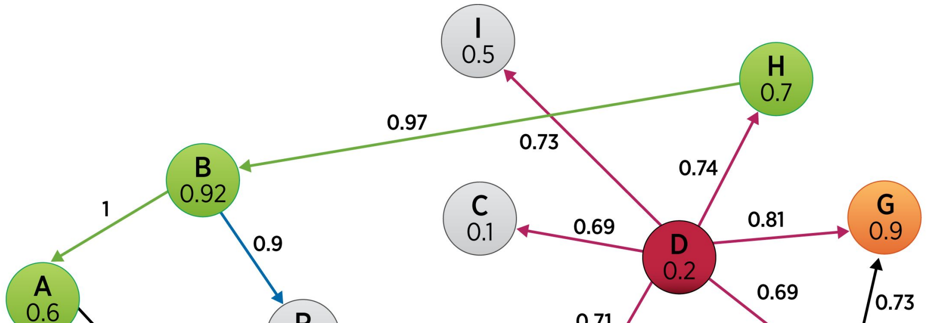
4. Compare the travel time of the first path and the new travel time of the new path





Honorable Mention: **Dijkstra Algorithm**

1. Create a set **S** to keep track of vertices included in shortest path (initially empty)
2. Assign a distance value to all vertices in the input graph. (the source vertex has a value of 0)
3. While set **S** doesn't include all vertices
   a. Pick a vertex **V** which is not in set **S** *and* has the smallest distance value from the rest
   b. Include **V** in **S**
   c. Update distance values of all adjacent vertices of **V**



The Back-End: **Technologies Used**

express

node JS™

Java™

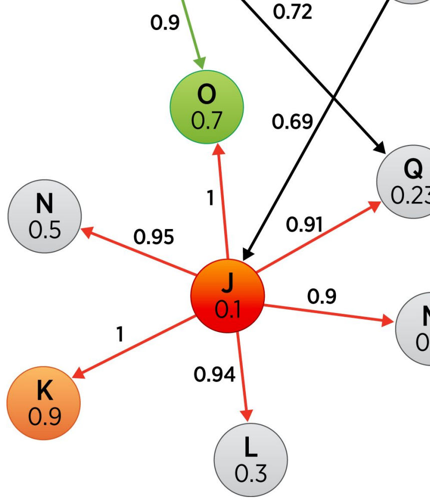
jQuery

The Back-End: **Tasks / Challenges**

- Convert Dijkstra's Algorithm from Java to JavaScript

The Back-End: **Tasks / Challenges**

- Convert Dijkstra’s Algorithm from Java to JavaScript
- Create cases
  - Ideally develop an algorithm to various cases
  - Vary numbers of nodes and edges
- Integration with frontend technologies



The Front-End: Technologies Used

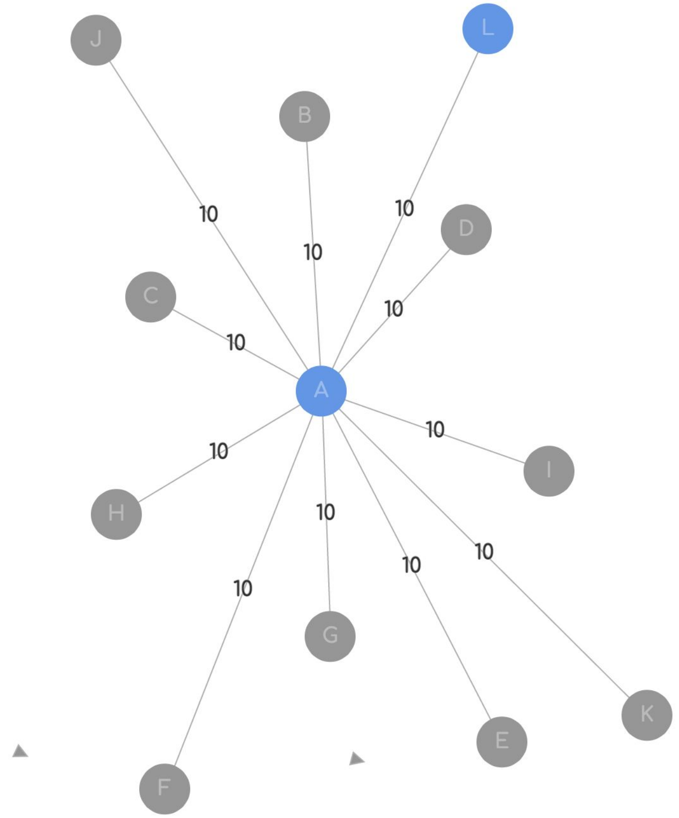
HTML CSS JS

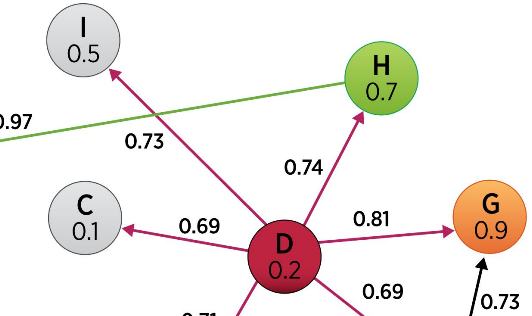
jQuery

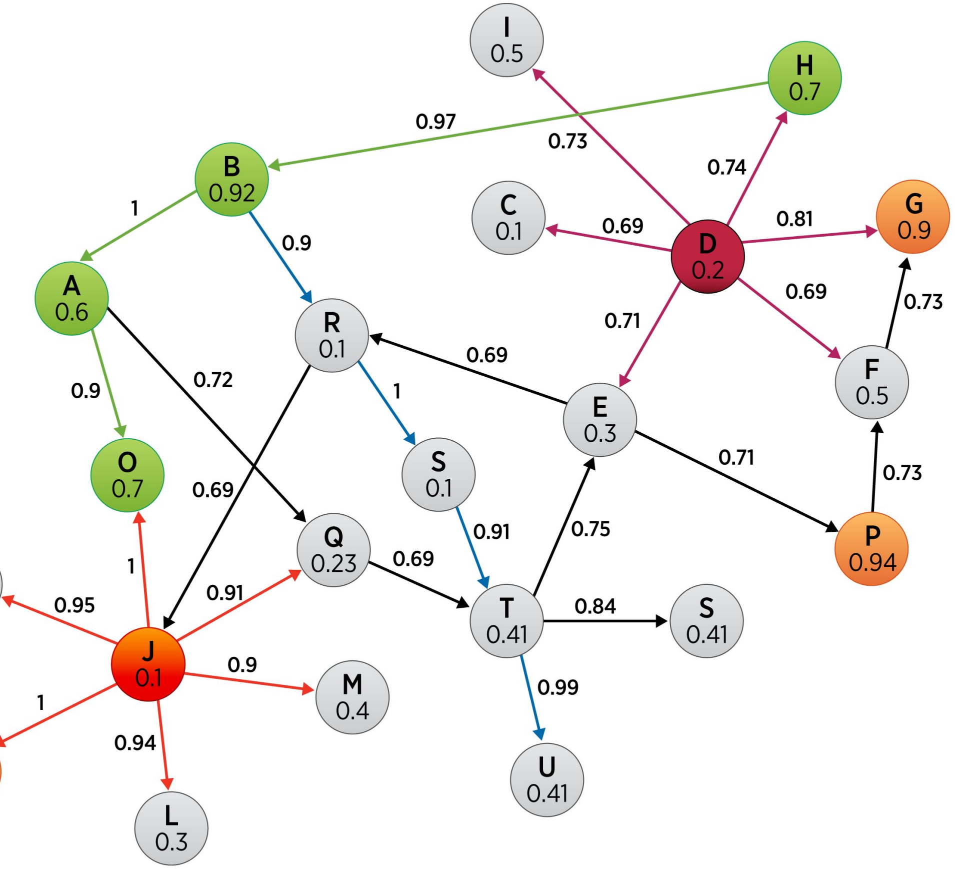
p5.js

The Front-End:

- Object oriented programming
  - Classes to allow the visualization of graphs created and simulation

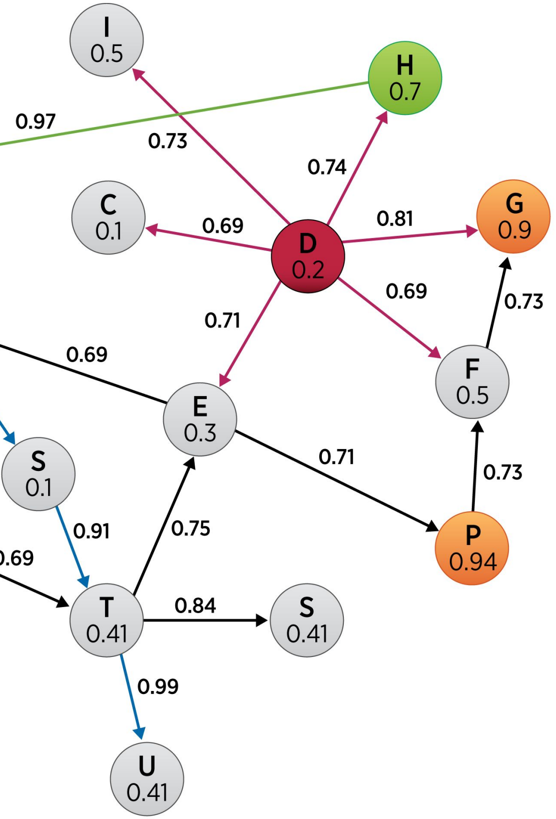





I 0.5
H 0.7
B 0.92
C 0.1
D 0.2
G 0.9
A 0.6
R 0.1
E 0.3
F 0.5
O 0.7
S 0.1
Q 0.23
P 0.94
T 0.41
S 0.41
J 0.1
M 0.4
U 0.41
L 0.3
0.97
0.73
0.74
1
0.9
0.69
0.81
0.69
0.71
0.73
0.72
0.69
0.9
1
0.71
0.73
0.69
0.91
0.75
0.69
1
0.95
0.91
0.84
0.9
0.99
1
0.94

DEMO TIME

End Goal:



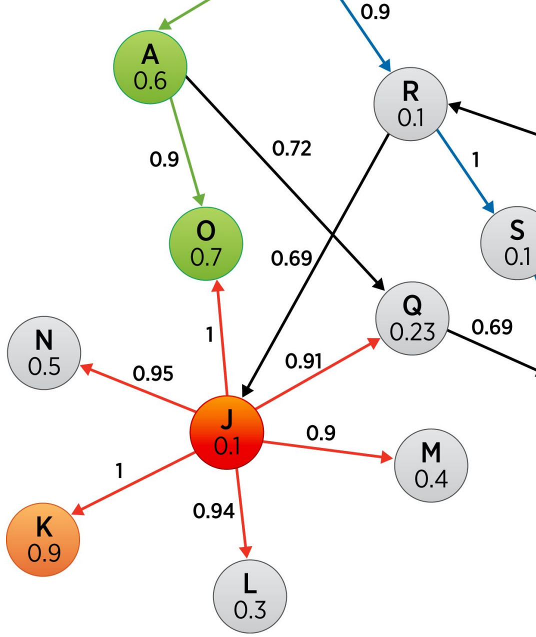
Empower the user to see how a specific decision about road infrastructure change affects the status quo by presenting the information in a visually rich yet simple manner.

Enable governments and third-parties to make informed decisions about making changes to existing road infrastructure.

Create a consistent way to test proposed changes and evaluate them according to social benefit and economic prudence.

Thank you!

Questions?


0.9
A
0.6
R
0.1
0.9
0.72
1
O
0.7
S
0.1
0.69
Q
0.23
0.69
N
0.5
1
0.95
0.91
J
0.1
0.9
M
0.4
1
0.94
K
0.9
L
0.3