Travel time uncertainty and network models

CE 392C
TRAVEL TIME
UNCERTAINTY
One major assumption throughout the semester is that travel times can be predicted exactly and are the same every day.

\[
C = 25.87321
\]

Thursday 11/19

Friday 11/20

Monday 11/23

Tuesday 11/22

... 

W can relax this assumption.
Outline

- What causes travel time uncertainty?
- Why do we care?
- How can we incorporate uncertainty into network models?
There are actually many reasons why travel times vary from day to day.

What proportion of total delay comes from “nonrecurring” causes?
Exact numbers vary, but some estimates suggest over half of delay can be attributed to nonrecurring causes.

If we are planning only for recurring congestion, we’re missing half of the story.

What if we replaced each link’s capacity with its average or expected capacity, rather than its nominal capacity?
Congestion is a nonlinear effect, and delay typically decreases *slower than linearly* with respect to capacity.

As a result, planning for average conditions will *systematically underestimate* the actual average congestion level.
Reliability plays a major role in travel choices, especially in mode and route choice.

Research shows that for many travelers, reliability is just as important as average travel time.
In the last two decades, agencies have placed an increasing focus on ITS rather than capacity expansion.

How can we evaluate strategies such as VMSs or active traffic management which are aimed at travel time reliability?

Variable Message Sign  Highway Advisory Radio  Parking Information

How can we evaluate strategies such as VMSs or active traffic management which are aimed at travel time reliability?
STOCHASTIC NETWORKS
We want to model uncertainty in link travel times, which we assume can be described by a known probability distribution.

<table>
<thead>
<tr>
<th>Travel time perception</th>
<th>Actual link travel time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deterministic</td>
<td>TAP</td>
</tr>
<tr>
<td>Stochastic</td>
<td>SUE</td>
</tr>
<tr>
<td>Deterministic</td>
<td>This week</td>
</tr>
<tr>
<td>Stochastic</td>
<td></td>
</tr>
</tbody>
</table>

In contrast to stochastic user equilibrium, here the uncertainty is in *network conditions*, not *user perception*.
First, we’ll address the “shortest path” version of this problem, where the probability distribution is fixed.

This is the perspective of an individual driver. Tomorrow we’ll deal with the equilibrium case, where travel times are flow-dependent.
Assume that each link \((i,j)\) can exist in one or more states \(s\) with known probabilities \(p^s\). The travel time in state \(s\) is \(t_{ij}^s\).

For instance...

<table>
<thead>
<tr>
<th>State</th>
<th>Probability</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal conditions</td>
<td>0.9</td>
<td>10</td>
</tr>
<tr>
<td>Accident</td>
<td>0.1</td>
<td>25</td>
</tr>
</tbody>
</table>
Assume that each link \((i, j)\) can exist in one or more states \(s\) with known probabilities \(p^s\). The travel time in state \(s\) is \(t_{ij}^s\).

For instance...

<table>
<thead>
<tr>
<th>State</th>
<th>Probability</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal conditions</td>
<td>0.7</td>
<td>10</td>
</tr>
<tr>
<td>Poor weather</td>
<td>0.18</td>
<td>15</td>
</tr>
<tr>
<td>Accident</td>
<td>0.07</td>
<td>25</td>
</tr>
<tr>
<td>Accident and poor weather</td>
<td>0.05</td>
<td>35</td>
</tr>
</tbody>
</table>
This means that the travel time on any path $\pi$ in the network is also a discrete random variable.

Can we find the shortest path without having to enumerate all of these possibilities?
First, what does it even mean to find the “shortest path” when $C^\pi$ is a random variable?

The simplest behavior assumption is that drivers want to find the path with least *expected* (average) travel time.

More sophisticated behavior assumptions are possible, to account for risk aversion, desired arrival times, etc. See my thesis and dissertation for more details.
The expected shortest path problem is surprisingly easy to solve, since the expected value is a linear operator:

\[
E \left[ C^\pi \right] = E \left[ \sum_{ij} \delta_{ij}^\pi t_{ij} \right] = \sum_{ij} \delta_{ij}^\pi E \left[ t_{ij} \right]
\]

Simply replace each link’s cost with its average value and find the shortest path.
A more interesting variant involves *en route* travel information: depending on the information you receive, you can change your path while driving.

This is called the *online shortest path* or *shortest hyperpath* problem.
This is fundamentally different than simply re-solving the expected shortest path problem whenever you receive information.
To keep things simple and tangible, let’s make a few assumptions:

- The state of a link is independent of the state of any other links you have previously encountered on your trip.
- In particular, repeated visits to the same link are *independent trials* and may result in different states.
- Upon arriving at a node, you learn the state (travel time) of each adjacent link.

These assumptions can be relaxed with varying degrees of ease. Independence and the type of information is “easy” to relax, at the cost of more complicated notation and higher memory requirements. The fully “no reset” version of the problem is much harder.
At each node, you receive a message $\theta$ containing the state of each adjacent link. Let $\Theta_i$ be the set of all possible messages received at node $i$, and denote $(i, \theta)$ as a node-state.

Drawbridge is open with probability $1/2$.

Dangerous road has incident with probability $1/10$.

No particular technology is assumed; the information can be from a VMS, phone app, simple observation, etc.
Instead of a single path, we want to find a routing policy which maps each node-state to the link you will choose to travel on next, if you are currently at node \( i \) and receive message \( \theta \).

Note that a simple path is what would result if you made the same choice for every message \( \theta \).
The online shortest path problem can be solved with a labeling algorithm similar to what we’ve seen for deterministic shortest paths, with a few changes:

- We solve the problem as an “all-to-one” rather than “one-to-all” shortest path, due to causality.
- We store a label for each node $L_i$ reflecting the expected travel time to the destination on the best-known policy.
- We store a label $\pi(i, \theta)$ for each node-state reflecting the best-known policy so far.
Let $q$ be the common destination.

1. Initialize each label $L_i$ to infinity (but $L_q = 0$)
2. Initialize each policy label $\pi(i, \theta)$ to $-1$.
3. Initialize $SEL$ to all nodes immediately upstream of $q$.
4. Remove an node $i$ from $SEL$ and set $temp$ to 0.
5. For each message $\theta \in \Theta_i$:
   1. Identify the travel times $t_{ij}^\theta$ associated with this message.
   2. Pick the arc $(i, j^*)$ which minimizes $t_{ij^*}^\theta + L_{j^*}$
   3. Set $\pi(i, \theta)$ to $(i, j^*)$.
   4. Add $p(\theta) \left( t_{ij^*}^\theta + L_{j^*} \right)$ to $temp$.
6. If $temp < L_i$ then update $L_i = temp$.
7. If the previous step changed the value of $L_i$, then add all nodes immediately upstream of $i$ to $SEL$.
8. If $SEL$ is empty, terminate; otherwise, return to step 4.
Example

Stochastic shortest paths

Stochastic networks
What happens in this network?
What is the optimal hyperpath?

What happens if you apply the algorithm from last class?

What is the expected travel time?
A few implications:

- It is possible for the shortest hyperpath to have a higher travel time than any simple path.
- Furthermore, the actual travel time on the shortest hyperpath may be arbitrarily large.
- However, the shortest hyperpath can never have a greater expected travel time.

Stochastic shortest paths

Stochastic networks
USER EQUILIBRIUM WITH RECOURSE
What does an equilibrium look like when people receive information?

What kind of information do people get?

Variable Message Sign  Highway Advisory Radio  Parking Information
Rather than having a fixed travel time for each state \( t_{ij}^s \), there is a different link performance function for each state \( t_{ij}^s(x_{ij}^s) \) as a function of \( x_{ij}^s \), the number of drivers who use link \((i, j)\) in state \( s \).

When drivers arrive at a node, they learn the state of each adjacent link.

Rather than choosing a path before leaving, drivers choose a hyperpath. (That is, they choose how they will respond to information received en route).

The principle of user equilibrium with recourse states that drivers will choose hyperpaths so that, for each OD pair, the expected travel time on each used hyperpath is equal and minimal.
Overall algorithmic approach:

1. Assign all drivers to shortest hyperpaths at free-flow.
2. Calculate $x_{ij}^s$ and update $t_{ij}^s$
3. Find new shortest hyperpaths, get target vector $\mathbf{x}^*$
4. Move from $\mathbf{x}$ in the direction of $\mathbf{x}^*$
5. Check convergence and iterate if necessary.
One can show that the user equilibrium with recourse solution minimizes the convex function

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

over the space of feasible hyperpath assignments

1. \( h^\pi \geq 0 \quad \forall \pi \in \Pi \)
2. \( \sum_{\pi \in \Pi^{rs}} h^\pi = d^{rs} \quad \forall (r, s) \in \mathbb{Z}^2 \)

where \( \Pi^{rs} \) represents the set of hyperpaths which start at \( r \) and terminate at \( s \) with probability 1.

What about the mapping \( x(h) \)?
Let’s address a simpler question: after solving online shortest path for a destination, can we find the link-state flows if everybody headed to that destination chose the shortest hyperpath?

If so, we can obtain $x^*$ by adding the flows for each destination.

If so, we can use the above algorithmic approach, which only involves all-or-nothing assignments $x^*$ and averaging.
Here’s what we want to do:

Stochastic shortest paths

User equilibrium with recourse

This works because the optimal policy depends only on the destination, not the origin.
Here’s what we want to do:

This works because the optimal policy depends only on the destination, not the origin.
Here’s what we want to do:

This works because the optimal policy depends only on the destination, not the origin.
Here’s what we want to do:

- **Stochastic shortest paths**
- **User equilibrium with recourse**

![Graph diagram]

**Demand** | **Messages** | **Policy**
---|---|---
A-D: 10 | A1 (0.8) | (A,A1): B
B-D: 50 | A2 (0.2) | (A,A2): C
C-D: 35 | B1 (0.5) | (B,B1): C
| B2 (0.5) | (B,B2): D
| C1 (1) | (C,C1): D

This works because the optimal policy depends only on the destination, not the origin.
Here’s what we want to do:

Stochastic shortest paths
User equilibrium with recourse
Algorithm steps: Let $\eta_i$ represent the number of vehicles at node $i$, and $SEL$ the set of nodes which still need to be processed.

1. Initialize $\eta_i$ to the demand between $i$ and destination.
2. For any node with $\eta_i > 0$, add $i$ to $SEL$.
3. While $SEL$ is nonempty:
   1. Choose a node $i$ from $SEL$ and remove it from the list.
   2. For each message $\theta$:
      1. The number of drivers who see that message is $\eta_i \cdot p(\theta)$
      2. For each message, identify the link $(i, j)$ in the shortest hyperpath.
      3. Load $\eta_i \cdot p(\theta)$ onto $x_{ij}^s$ where $s$ is the state corresponding to $\theta$.
      4. Increase $\eta_j$ by $\eta_i \cdot p(\theta)$ unless $j$ is the destination.
      5. Add $j$ to $SEL$ unless $j$ is the destination.
3. Set $\eta_i$ to 0.
How should we pick a node from SEL?

What about cycles?

In practice, define a small threshold $\eta_{min}$; if $\eta_i < \eta_{min}$, shift all flow to the adjacent node $j$ with minimum $L_j$ label.

There is an alternative approach involving solution of a sparse linear system of equations.
Overall algorithmic approach:

1. Assign all drivers to shortest hyperpaths at free-flow.
2. Calculate $x_{ij}^s$ and update $t_{ij}^s$
3. Find new shortest hyperpaths, get target vector $\mathbf{x}^*$
4. Move from $\mathbf{x}$ in the direction of $\mathbf{x}^*$ (MSA or Frank-Wolfe)
5. Check convergence and iterate if necessary.