Shortest Path in Transportation Networks
Consider two lanes of traffic that are perfectly controlled so that there are only two streams of traffic:

- Fast vehicles all travel at 100 km/h in the inner lane.
- Slow vehicles all move at 50 km/h in the outer lane.

Traffic flow in each lane is 1000 vehicles per hour, and lane change is forbidden.

**Time mean speed** is the average speed of all vehicles passing a reference point over a duration of time. It is the simple average of spot speed. Time mean speed $v_t$ is given by

$$v_t = \frac{1}{n} \sum_{i=1}^{n} v_i$$

$v_i$ is the spot speed of $i$th vehicle, and $n$ is the number of vehicles passing the fixed point.
Consider two lanes of traffic that are perfectly controlled so that there are only two streams of traffic:

- Fast vehicles all travel at 100 km/h in the inner lane.
- Slow vehicles all move at 50 km/h in the outer lane.

Traffic flow in each lane is 1000 vehicles per hour, and lane change is forbidden.

**Time mean speed** is the average speed of all vehicles passing a reference point over a duration of time. It is the simple average of spot speed. Time mean speed \( v_t \) is given by

\[
v_t = \frac{1}{n} \sum_{i=1}^{n} v_i
\]

\( v_i \) is the spot speed of \( i \)th vehicle, and \( n \) is the number of vehicles passing the fixed point.

\[
v_t = \frac{1}{2000}(1000 \times 100 + 1000 \times 50) = 75
\]
Consider two lanes of traffic that are perfectly controlled so that there are only two streams of traffic:

- Fast vehicles all travel at 100 km/h in the inner lane.
- Slow vehicles all move at 50 km/h in the outer lane.

Traffic flow in each lane is 1000 vehicles per hour, and lane change is forbidden.

**Space mean speed** is measured over the whole roadway segment and \( n \) represents the number of vehicles passing the roadway segment.

\[
v_s = \frac{n}{\sum_{i=1}^{n} \left( \frac{1}{v_i} \right)}
\]

\( V_s = \frac{\text{total no. of vehicles}}{[(\text{no. of vehicles/speed}) + (\text{no. of vehicles/speed})]} \)

\( V_s = \frac{(1000+1000)}{[(1000/100)+(1000/50)]} \)

\( = \frac{2000}{(3000/100)} \)

\( = 66.667 \text{ Km/h} \)
Consider two lanes of traffic that are perfectly controlled so that there are only two streams of traffic:

- Fast vehicles all travel at 100 km/h in the inner lane.
- Slow vehicles all move at 50 km/h in the outer lane.

Traffic flow in each lane is 1000 vehicles per hour, and lane change is forbidden.

**Space mean speed** is measured over the whole roadway segment and $n$ represents the number of vehicles passing the roadway segment.

$$v_s = \frac{n}{\sum_{i=1}^{n} (1/v_i)}$$

**Vs** = total no. of vehicles / [(no. of vehicles/speed) + (no. of vehicles/speed)]

$$Vs = \frac{(1000+1000)}{[(1000/100)+(1000/50)]=2000 / (3000/100)} = 66.667 \text{ Km/h}$$

Flow: 1000 veh/h
What is veh/km?

Incorrect :(
Consider two lanes of traffic that are perfectly controlled so that there are only two streams of traffic:

- Fast vehicles all travel at 100 km/h in the inner lane.
- Slow vehicles all move at 50 km/h in the outer lane.

Traffic flow in each lane is 1000 vehicles per hour, and lane change is forbidden.

Space mean speed is measured over the whole roadway segment and \( n \) represents the number of vehicles passing the roadway segment.

\[
v_s = \frac{n}{\sum_{i=1}^{n} \left( \frac{1}{v_i} \right)}
\]

Flow: 1000 veh/h

What is \#veh/km?

Flow/Speed

In the inner lane: \#veh/km=1000/100=10

In the outer lane: \#veh/km=1000/50=20
Consider two lanes of traffic that are perfectly controlled so that there are only two streams of traffic:

- Fast vehicles all travel at 100 km/h in the inner lane.
- Slow vehicles all move at 50 km/h in the outer lane.

Traffic flow in each lane is 1000 vehicles per hour, and lane change is forbidden.

**Space mean speed** is measured over the whole roadway segment and \( n \) represents the number of vehicles passing the roadway segment.

\[
v_s = \frac{n}{\sum_{i=1}^{n}(1/v_i)}
\]

Flow: 1000 veh/h

What is #veh/km? Flow/Speed

In the inner lane: #veh/km = 1000/100 = 10
In the outer lane: #veh/km = 1000/50 = 20

\[
v_s = \frac{30}{\frac{10}{100} + \frac{20}{50}} = 60
\]
Each plot exhibits a trend which suggests a pairwise relationship among flow, speed, and density.

For example, the top-left plot reveals a decreasing relationship between speed and density with two intercepts intuitively known.

One intercept represents a scenario where there are very few vehicles on the road (i.e., $k \rightarrow 0$). Hence, one may drive at the desired speed without being blocked by a slow driver ($v \rightarrow v_f$, the free-flow speed).

The other intercept corresponds to a scenario where everyone rushes home. As such, the road is jammed ($k \rightarrow k_j$, the jam density), resulting in a stop-and-go condition ($v \rightarrow 0$).

Numerous functions have been proposed to establish the relationship between speed and density.
Each plot exhibits a trend which suggests a pairwise relationship among flow, speed, and density.

For example, the top-left plot reveals a decreasing relationship between speed and density with two intercepts intuitively known.

One intercept represents a scenario where there are very few vehicles on the road (i.e., $k \to 0$). Hence, one may drive at the desired speed without being blocked by a slow driver ($v \to v_f$, the free-flow speed).

The other intercept corresponds to a scenario where everyone rushes home. As such, the road is jammed ($k \to k_j$, the jam density), resulting in a stop-and-go condition ($v \to 0$).

Numerous functions have been proposed to establish the relationship between speed and density.

All these models are one equation models, meaning that the models apply to the entire range of density. The models are typically simple because they involve few parameters and typically suffer from poor fitting quality.
Each plot exhibits a trend which suggests a pairwise relationship among flow, speed, and density.

For example, the top-left plot reveals a decreasing relationship between speed and density with two intercepts intuitively known.

One intercept represents a scenario where there are very few vehicles on the road (i.e., $k \to 0$). Hence, one may drive at the desired speed without being blocked by a slow driver ($v \to v_f$, the free-flow speed).

The other intercept corresponds to a scenario where everyone rushes home. As such, the road is jammed ($k \to k_j$, the jam density), resulting in a stop-and-go condition ($v \to 0$).

Numerous functions have been proposed to establish the relationship between speed and density.

All these models are one equation models, meaning that the models apply to the entire range of density.

The models are typically simple because they involve few parameters and typically suffer from poor fitting quality.
Lecture 9 (3 Nov 2021)
Shortest Path in Transportation Networks
Ottawa
Ontario, Canada

Get on Trans-Canada Hwy/CN-401 W
Exit 8

Take ON-401 W/OH-401 W, Ontario 401 Express, OH-401 W, ... and Pan-American Hwy/CA-1 to Carte de Bassasau/CA-9 in Choluteca, Honduras

Continue on CA-3. Take NN-252, NCO-22 and CA-1, Panama/CN-CA-1/CA-SA/SA-2 1.8 km in Maragol, Nicaragua
The speed is 93 km/h (58 mph)

Get on Pan-American/CA-1/CA-SA/SA-2 2
Length: 135 km

Continue on Pan-American/CA-1/CA-SA/SA-2 2
Take CA-3, Pan-American/CA-1/CA-SA/SA-2 2
Take CA-3, Pan-American/CA-1/CA-SA/SA-2 2
Take CA-3, Pan-American/CA-1/CA-SA/SA-2 2

Panama City
Panama

These instructions are for planning purposes only. You may find or construction ongoing, traffic, weather or other events that cause unexpected changes to the route. Please plan accordingly. You must obey all signs or directives regarding your route.
Sorry, we could not calculate driving directions from "Ottawa, Ontario, Canada" to "Lima, Peru"
Wardrop's first principle of route choice, also known as "user equilibrium"

The journey times in all routes actually used are equal and less than those that would be experienced by a single vehicle on any unused route.

John Glen Wardrop
(1922–1989)
Shortest Path Algorithms

Let $G = (V, E)$ be a (directed) graph with a set $V$ of vertices and a set $E$ of edges. Each edge $(u, v) \in E$ has an associated nonnegative length $l(u, v)$.

Different variations of shortest path algorithms:

- point-to-point
- one-to-all
- many-to-many
- all-pairs
Standard one-to-all algorithm: Dijkstra

Edsger Wybe Dijkstra
1930-2002
Dijkstra Algorithm

\[
\text{dist}[s] \leftarrow 0 \\
\text{for all } v \in V - \{s\} \text{ do } \text{dist}[v] \leftarrow \infty \\
S \leftarrow \emptyset \\
Q \leftarrow V \\
\text{while } Q \neq \emptyset \text{ do } u \leftarrow \text{mindistance}(Q, \text{dist}) \\
S \leftarrow S \cup \{u\} \\
\text{for all } v \in \text{neighbors}[u] \text{ do if } \text{dist}[v] > \text{dist}[u] + w(u, v) \text{ then } \text{dist}[v] \leftarrow \text{dist}[u] + w(u, v) \\
\text{return dist}
\]

(distance to source vertex is zero)
(set all other distances to infinity)
(S, the set of visited vertices is initially empty)
(Q, the queue initially contains all vertices)
(while the queue is not empty)
(select the element of Q with the min. distance)
(add u to list of visited vertices)
(if new shortest path found)
(set new value of shortest path)
(if desired, add traceback code)

Complexity: \(O(|V| + |E| \log |V|)\)
Dijkstra Algorithm - Example
Dijkstra Algorithm - Example
Dijkstra Algorithm - Example
Dijkstra Algorithm - Example
Dijkstra Algorithm - Example
Dijkstra Algorithm - Example

![Dijkstra Algorithm Diagram](image_url)

![Dijkstra Algorithm Table](image_url)
Dijkstra Algorithm - Example
Goal-Directed Techniques: A* search

- uses a potential function $h : V \rightarrow R$ on the vertices, which is a lower bound on the distance $\text{dist}(u, t)$ from $u$ to $t$.
- then runs a modified version of Dijkstra’s algorithm in which the priority of a vertex $u$ is set to $\text{dist}(s, u) + h(u)$. 
A* search - Example

A* Search Algorithm

What is the shortest path to travel from A to Z?

Numbers in orange are the heuristic values, distances in a straight line (as the crow flies) from a node to node Z.
## A* search - Example

Start by setting the starting node (A) as the current node.

**Table:**

<table>
<thead>
<tr>
<th>Node</th>
<th>Status</th>
<th>Shortest Distance From A</th>
<th>Heuristic Distance to Z</th>
<th>Total Distance*</th>
<th>Previous Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Current</td>
<td>0</td>
<td>14</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>∞</td>
<td>12</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>∞</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>∞</td>
<td>6</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>∞</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>∞</td>
<td>11</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Z</td>
<td>∞</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Total Distance = Shortest Distance from A + Heuristic Distance to Z*
A* search - Example

Check all the nodes connected to A and update their "Shortest Distance from A" and set their "previous node" to "A".
Update their total distance by adding the shortest distance from A and the heuristic distance to Z.
Set the current node (A) to “visited” and use the unvisited node with the smallest total distance as the current node (e.g. in this case: Node C).

Check all unvisited nodes connected to the current node and add the distance from A to C to all distances from the connected nodes. Replace their values only if the new distance is lower than the previous one.

C → D: 3 + 7 = 10 < ∞ – Change Node D
C → E: 3 + 10 = 13 < ∞ – Change Node E

The next current node (unvisited node with the shortest total distance) could be either node B or node D. Let’s use node B.
Check all unvisited nodes connected to the current node (B) and add the distance from A to B to all distances from the connected nodes. Replace their values only if the new distance is lower than the previous one.

B -> E: $4 + 12 = 16 > 13$ – Do not change Node E
B -> F: $4 + 5 = 9 < \infty$ – Change Node F

The next current node (unvisited node with the shortest total distance) is D.
A* search - Example

Check all unvisited nodes connected to the current node (D) and add the distance from A to D to all distances from the connected nodes. Replace their values only if the new distance is lower than the previous one.

D -> E: 10 + 2 = 12 < 13 – Change Node E

The next current node (unvisited node with the shortest total distance) is E.
A* search - Example

Check all unvisited nodes connected to the current node (E) and add the distance from A to E to all distances from the connected nodes. Replace their values only if the new distance is lower than the previous one.

E → Z: 12 + 5 = 17 < ∞ – Change Node Z
A* search - Example

We found a path from A to Z, but is it the shortest one?

Check all unvisited nodes. In this example, there is only one unvisited node (F). However its total distance (20) is already greater than the distance we have from A to Z (17) so there is no need to visit node F as it will not lead to a shorter path.

We found the shortest path from A to Z.

Read the path from Z to A using the previous node column:

Z > E > D > C > A

So the Shortest Path is:

A – C – D – E – Z with a length of 17
Dijkstra

A*
Real-world Extensions

We are often interested in more than just the length of the shortest path between two points in a static network. Most importantly, one should also be able to retrieve the shortest path itself.

Some applications require computing multiple paths at once. For example, advanced logistics applications may need to compute all distances between a source set $S$ and a target set $T$.

Transportation networks tend to be dynamic, with unpredictable delays, traffic, or closures. In real transportation networks, the best route often depends on the departure time in a predictable way.
Test on the road network of Western Europe from PTV AG, with 18.0 million vertices and 42.5 million directed edges.

### Route Planning in Road Networks

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Impl. source</th>
<th>Data structures</th>
<th>Queries</th>
<th>Time [h:m]</th>
<th>Scanned vertices</th>
<th>Time [μs]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dijkstra</td>
<td>[75]</td>
<td>0.4</td>
<td>9326696</td>
<td>2195080</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bidir. Dijkstra</td>
<td>[75]</td>
<td>0.4</td>
<td>4914804</td>
<td>1205660</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CRP</td>
<td>[78]</td>
<td>0.9</td>
<td>2766</td>
<td>1650</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arc Flags</td>
<td>[75]</td>
<td>0.6</td>
<td>2646</td>
<td>408</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CH</td>
<td>[78]</td>
<td>0.4</td>
<td>280</td>
<td>110</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CHASE</td>
<td>[75]</td>
<td>0.6</td>
<td>28</td>
<td>5.76</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HLC</td>
<td>[82]</td>
<td>1.8</td>
<td>0.50</td>
<td>2.55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TNR</td>
<td>[15]</td>
<td>2.5</td>
<td>0.22</td>
<td>2.09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TNR+AF</td>
<td>[40]</td>
<td>5.4</td>
<td>1.24</td>
<td>0.70</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HL</td>
<td>[82]</td>
<td>18.8</td>
<td>0.37</td>
<td>0.56</td>
<td></td>
<td></td>
</tr>
<tr>
<td>HL∞</td>
<td>[5]</td>
<td>17.7</td>
<td>60:00</td>
<td>0.25</td>
<td></td>
<td></td>
</tr>
<tr>
<td>table lookup</td>
<td>[75]</td>
<td>1208358.7</td>
<td>145:30</td>
<td>0.06</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Some Remarks

- The last decade has seen astonishing progress in the performance of shortest path algorithms on transportation networks.
- Careful engineering is essential to unleash the full computational power of modern computer architectures.
- Journey planning on public transportation systems, although conceptually similar, is a significantly harder problem due to its inherent time-dependent and multicriteria nature.
- Companies like Apple, Esri, Google, MapBox, Microsoft, Nokia, PTV, TeleNav, TomTom, and Yandex work on routing-related projects and tend to be secretive about the actual algorithms they use.
- The ultimate goal, a worldwide multimodal journey planner, has not yet been reached.
Some Remarks

- The last decade has seen astonishing progress in the performance of shortest path algorithms on transportation networks.
- Careful engineering is essential to unleash the computational power of modern computer architectures.
- Journey planning on public transportation systems, although conceptually similar, is a significantly harder problem due to its inherent time-dependent and multicriteria nature.
- Companies like Apple, Esri, Google, MapBox, Microsoft, Nokia, PTV, TeleNav, TomTom, and Yandex work on routing-related projects and tend to be secretive about the actual algorithms they use.
- The ultimate goal, a worldwide multimodal journey planner, has not yet been reached.
Do People Use the Shortest Path?

Wardrop's first principle:
The journey times in all routes actually used are equal and less than those that would be experienced by a single vehicle on any unused route.
Do People Use the Shortest Path?

Travelers differ in

- attributes (value of time (VOT), willingness to pay, time budgets, etc.),
- behavioral preferences (e.g. willingness to take risks, willingness to switch routes with potential savings),
- experience, and knowledge about travel,

all of which could lead to significant heterogeneity in route choice behavior.

Wardrop's first principle:
The journey times in all routes actually used are equal and less than those that would be experienced by a single vehicle on any unused route.
Case study:
3 weeks, random vehicles equipped with GPS devices, Minneapolis, St. Paul metropolitan area.

- The results show that about two-thirds of the subjects do not use the shortest travel time path.
- No subjects followed the shortest distance path unless it also coincided with the shortest travel time path.
- Travelers clearly have other preferences when making their route choices. Therefore, a better understanding of people’s route preferences could also inform the development of choice set generation algorithms.