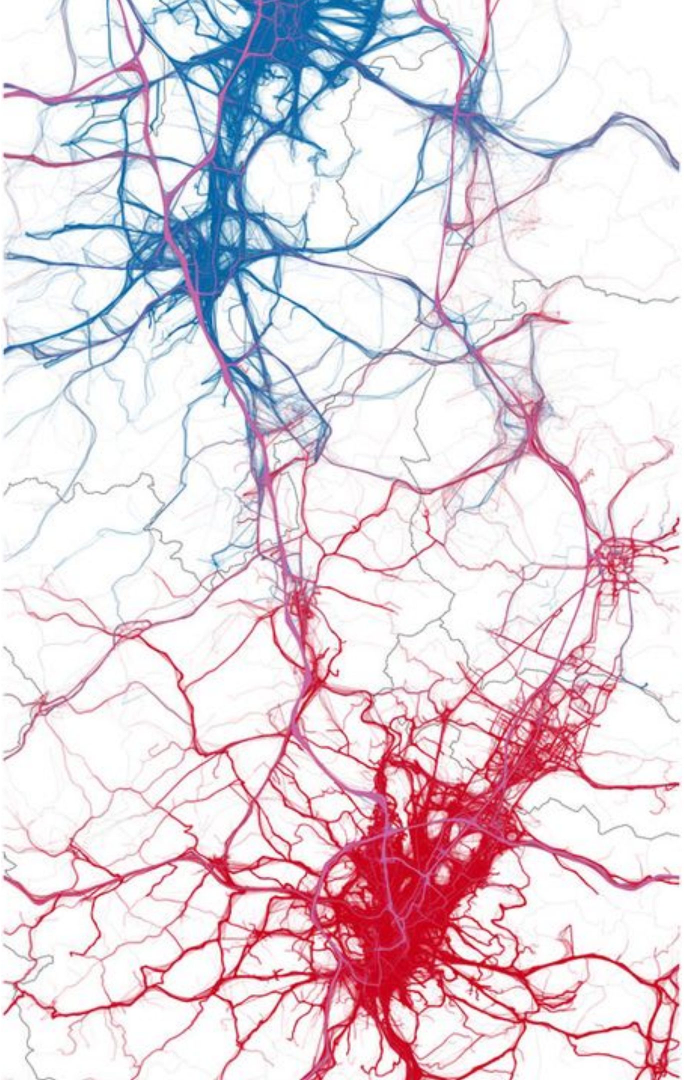


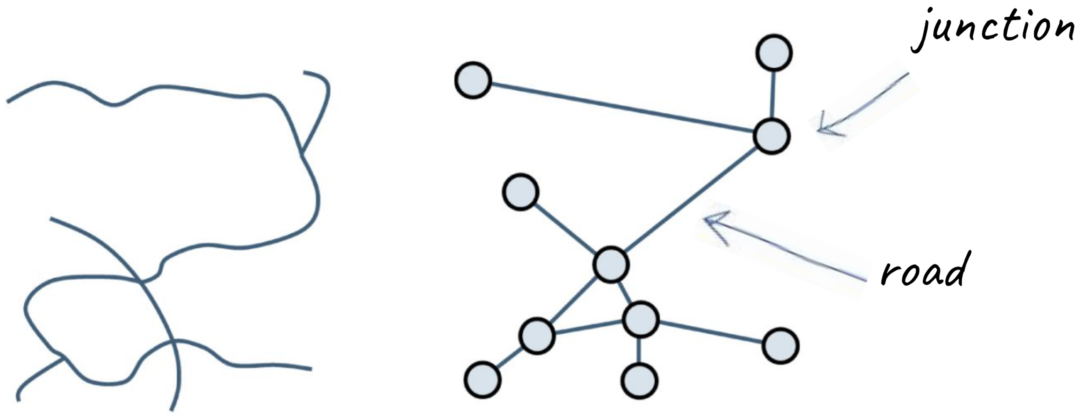
Consiglio Nazionale
delle Ricerche

Alternative Routing

The Road Network

The road network is described as a **weighted directed graph**:

- **nodes** represent intersections/junctions
- **edges** represent roads/streets
- **edge weights** represent road length or expected travel time



The Routing Problem

What is the “best” route to reach a destination from an origin?

- What does “good” mean? Its subjective (i.e., max. route dissimilarity, min. length, etc)

- The “best” route for an individual is not necessarily the best anymore when many vehicles are travelling at the same time

Shortest/Fastest route

- The default solution to routing is providing the shortest/fastest path (Dijkstra algorithm)

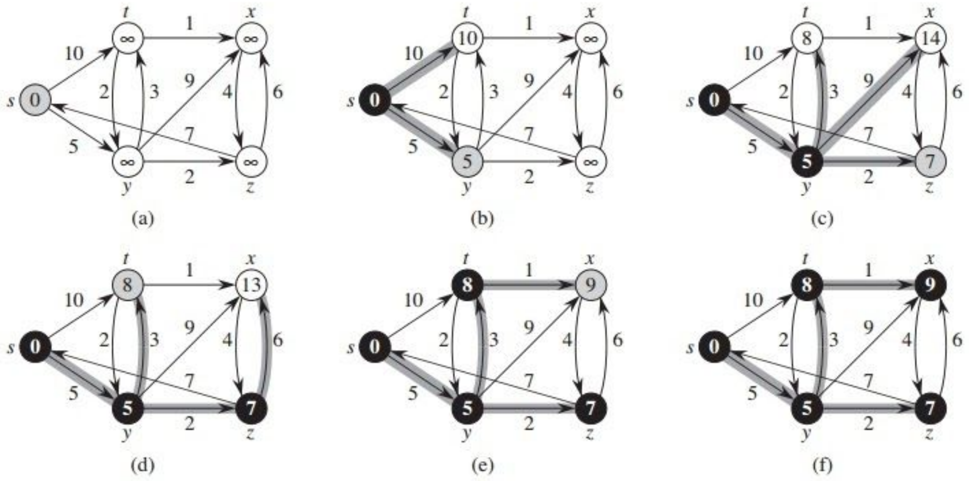
```
function Dijkstra(Graph, source):

    for each vertex v in Graph.Vertices:
        dist[v] ← INFINITY
        prev[v] ← UNDEFINED
        add v to Q
    dist[source] ← 0

    while Q is not empty:
        u ← vertex in Q with min dist[u]
        remove u from Q

        for each neighbor v of u still in Q:
            alt ← dist[u] + Graph.Edges(u, v)
            if alt < dist[v]:
                dist[v] ← alt
                prev[v] ← u

    return dist[], prev[]
```



Is the shortest enough?

In many scenarios, the shortest path is not enough:

- *Example 1: navigation systems*
  (longer) **alternative routes** with desirable properties

- *Example 2: humanitarian aid goods transport*
  distribution of vehicles on **non-overlapping routes** increases the chances that goods will be delivered

- *Example 3: emergencies*
  Alternative, **safe routes** in case of earthquakes, terrorist attacks, evacuation plans

Alternative Routing Methods

Alternative Routing (AR) aims to generate a set of ***k*** **good alternative** routes between an **origin** and a **destination**

Route generation framework

Input:

- road network G
- an int $k > 1$
- an (o, d) pair

Output:

- k alternative paths

k-Shortest Paths

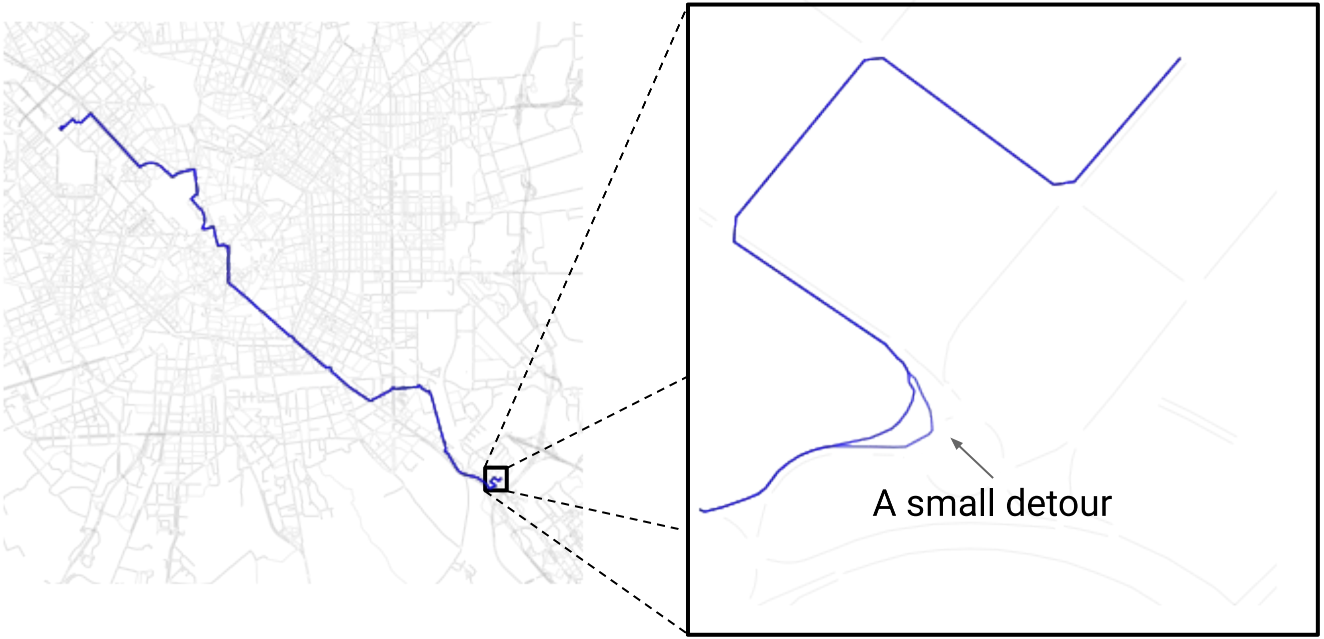
Naive solution: generate **k-shortest paths** between an origin and a destination

Limitations:

- the k-shortest path solutions **fail** to provide significant **path diversification**

- the routes exhibit a **99% overlap**, with minor differences (cutting a corner or small detours)

k-Shortest Paths



k-Disjoint Paths

Generate **k-shortest disjoint paths**, i.e.,
k alternative paths with **no common edges**

- In practice, we put the edge weights of the current shortest path to infinity
- This enforces the diversity among paths

Limitations:

- routes **significantly deviate** from the shortest path
  - increased travel time and length
- **no guarantee** that **k** disjoint paths exists

Alternative Routing Approaches

Several existing Alternative Routing approaches lie between the k-shortest path and k-shortest disjoint paths:

1. Edge Weight Approaches

2. Plateau Approaches

3. Dissimilarity Approaches

Edge Weight Approaches

Compute the shortest paths **iteratively:**

- at each iteration, **manipulate** the road network's **edge weights**

- edge weight manipulation involves the **randomization** of the weights or a **cumulative penalization** of the shortest path's edges

Path Penalization

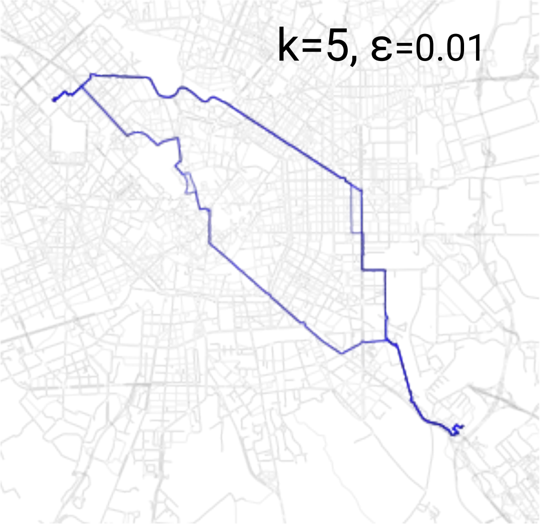
Until # of alternative paths < k:

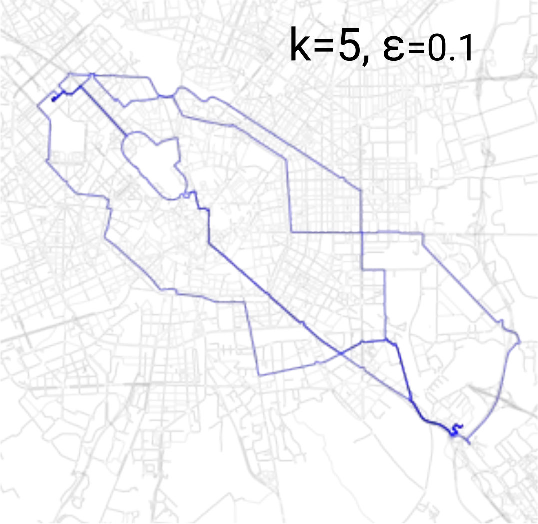
- Compute the shortest path using the current edge weights

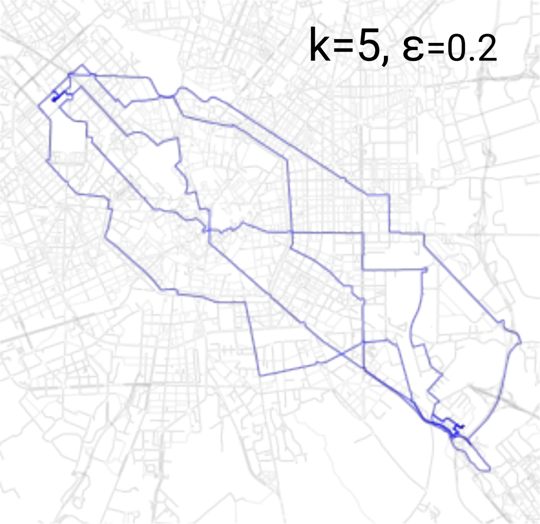
- Apply a **penalization factor ε** to each edge weight in the shortest path

$$\forall e \in p_s, w(e) = w(e)(1 + \epsilon)$$

Path Penalization







- The penalty factor **ε** controls the degree of deviation of an alternative path from previously generated ones
- It influences the geographic distribution of alternative paths

Graph Randomization

Until # of alternative paths < k:

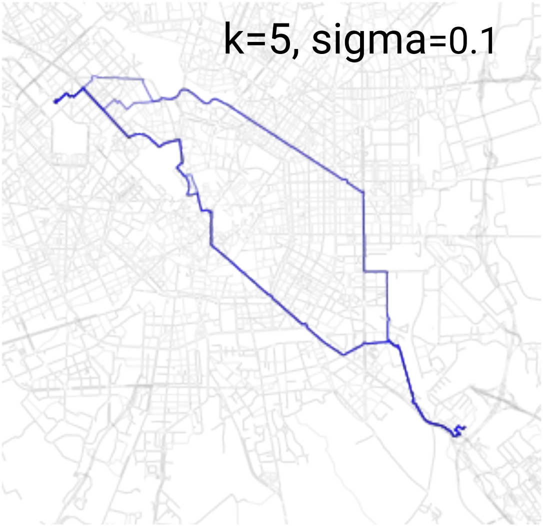
- Compute the shortest path using the current edge weights
- Randomize the weights of **all edges in G** adding a value v drawn from a normal distribution

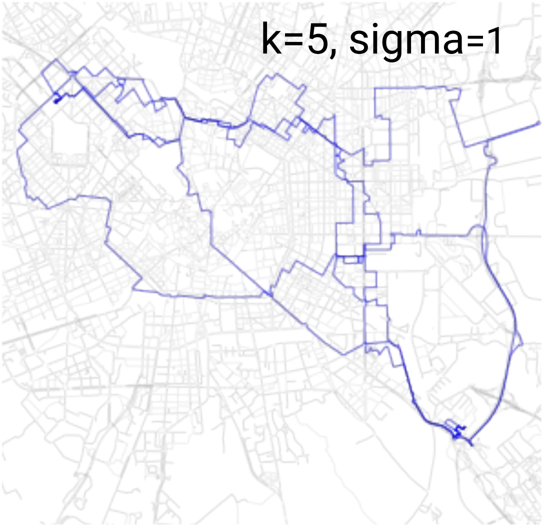
$$\forall e \in G, w(e) = w(e) + v$$

$$N(0, w(e)^2 \cdot \sigma^2)$$

normal distribution

Graph Randomization





Path Randomization

Until # of alternative paths < k:

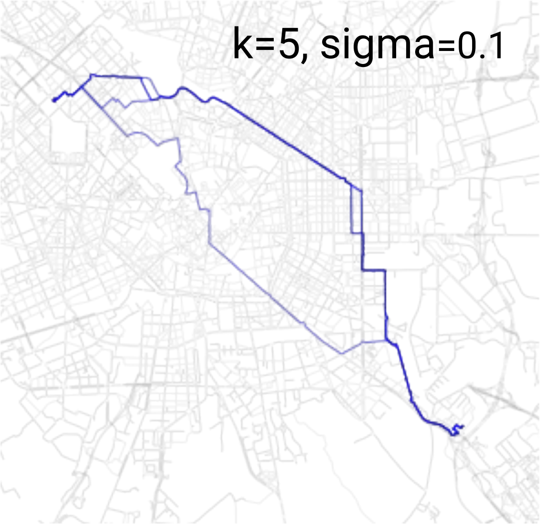
- Compute the shortest path using the current edge weights
- Randomize the weights of the shortest path adding a value v drawn from a normal distribution

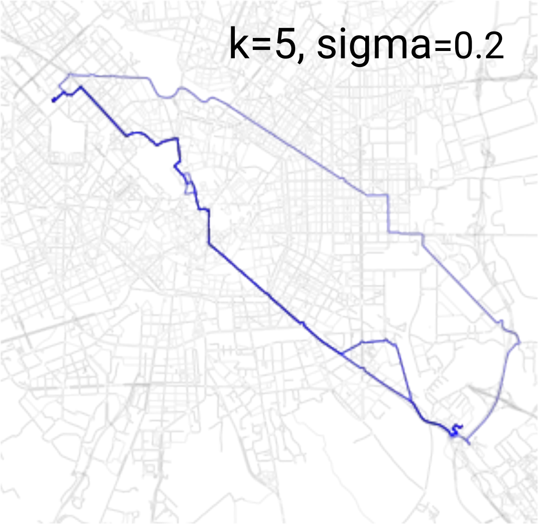
$$\forall e \in p_s, w(e) = w(e) + v$$

$$N(0, w(e)^2 \cdot \sigma^2)$$

normal distribution

Path Randomization





Which ones of the AR algorithms are deterministic?

A. K-shortest ✓
B. K-disjoint ✓
C. Path Randomization
D. Path Penalization ✓
E. Graph Randomization

In which AR algorithm weight can also decrease?

A. Path Randomization ✔
B. Path Penalization
C. Graph Randomization ✔

What is the range of possible path counts between locations O and D that Path Penalization may return after N iterations?

Minimum:

- 0 paths if O and D are not connected.
- 1 path if only the fastest path is found.

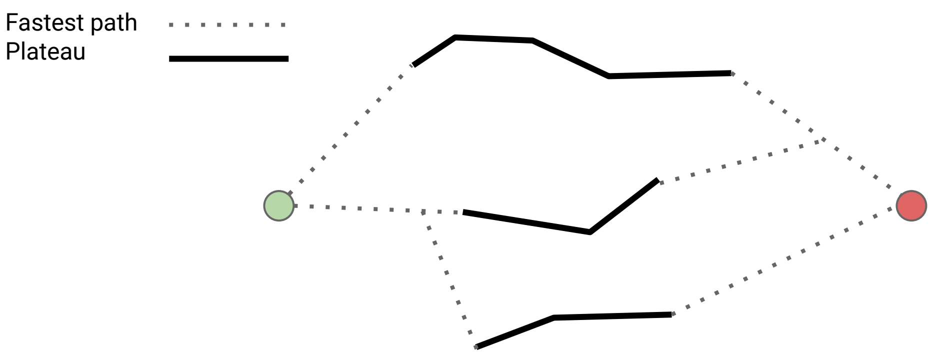
Maximum:

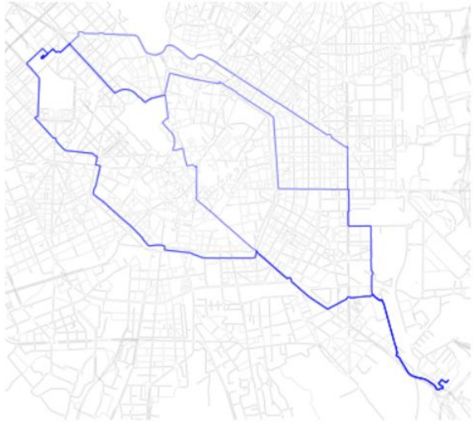
- N paths if a new path is discovered at each of the N iterations.

Plateau Approaches

Build two shortest-path trees, one from the source and one from the destination:

- identify their common branches (**plateaus**)
- select top-k plateaus by length
- append the shortest paths from the source to the plateau's first edge and from the last edge to the target





Dissimilarity Approaches

Dissimilarity approaches generate k alternative paths that satisfy a **dissimilarity constraint** and a desired **property**

- k-Shortest Paths with Limited Overlap
- k-Dissimilar Paths with Minimum Collective Length
- k-Most Diverse Near Shortest Paths

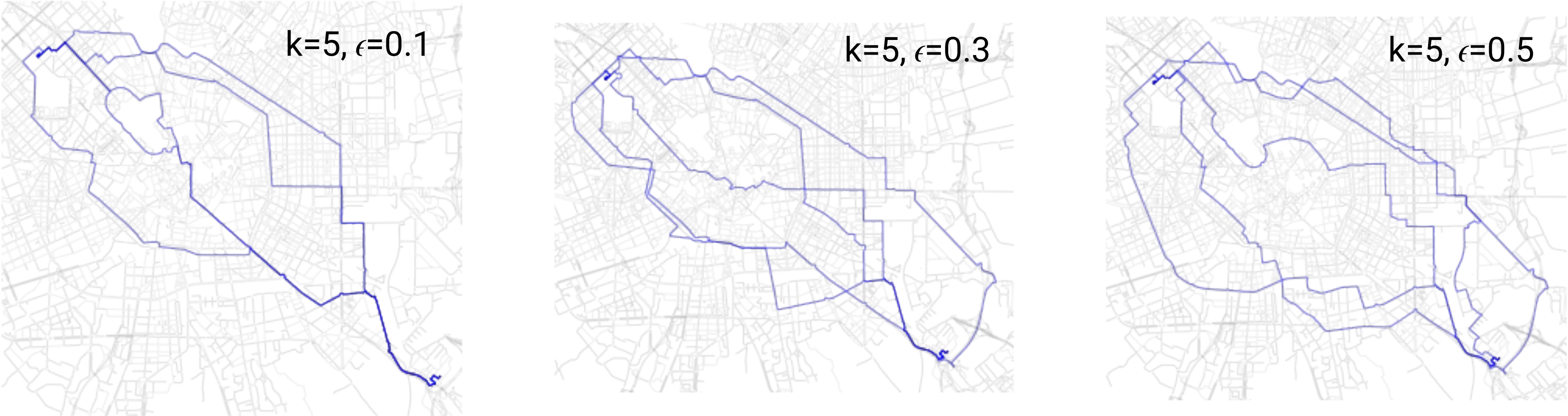
k-Most Diverse Near Shortest Paths (KMD)

- KMD generates k routes with the **highest dissimilarity** while still adhering to a user-defined **cost threshold** ϵ
- It is **NP-Hard**: we need a **heuristic**

Given an origin o and a destination d:

1. Define a cost threshold $c \cdot$ (1+ϵ) for a path to be Near Shortest (NSP)
2. Until no more near-shortest paths can be found, repeat:
   - generate a new NSP **p** and adds it to the set of NSPs **S**.
     - Use path penalization algorithm
   - generate all possible **subsets** of **S** with **k** elements containing **p** and identifies Sdiv as the most diverse one (based on jaccard). If it is the most diverse found up to this point Pkmd = Sdiv.
3. Return the subset of k paths with the highest diversity, i.e., Pkmd.

k-Most Diverse Near Shortest Paths (KMD)



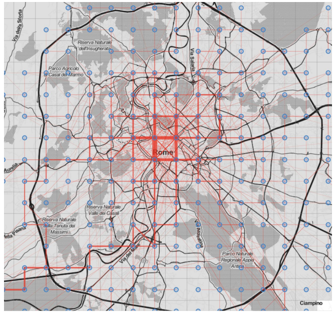
Traffic Assignment Problem (TA)

Given a demand, assign each trip with a route

Traffic Assignment Problem (TA)

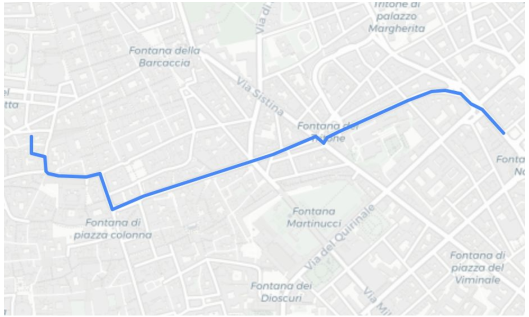
Given a **demand**, assign each **trip** with a **route**

collection of origin-destination pairs



a single origin-destination pair

sequence of road edges



AON vs ITA

All or Nothing (AON): assign the fastest path to each trip

- It creates concentration of the traffic on a few routes

Incremental Traffic Assignment (ITA): extends AON incorporating the dynamic travel time changes within a road edge

- create n splits of the demand (typically n = 4 with 40%, 30%, … 10%)
- Split 1: trips are assigned using AON; each edge’s travel time is updated using the BPR function (Bureau of Public Roads)
- Split 2: trips are assigned using AON, considering the updated travel time
- Iterate

METIS

Algorithm 1: METIS

Input : road network G, mobility demand D, penalization factor p, slow factor s

Output : sequence of assigned routes R

// Initialization Phase

$K_{\text{road}}^{(\text{source})}, K_{\text{road}}^{(\text{end})} \leftarrow KRoadEstimation(G, D)$;

$R \leftarrow \emptyset$;

// Perform the Traffic Assignment (TA)

foreach $j = (o, d, t) \in D$ **do**

  // Apply the Forward-Looking Edge Penalization (FLEP)

  $H \leftarrow FLEP(G, R, D, (p, s), t)$;

  // Generate a set of k candidates on the penalized road network

  $P \leftarrow kMDNSP(H, o, d)$;

  // Select the route that minimizes the score function

  $r \leftarrow RouteSelection(P, K_{\text{road}}^{(\text{source})}, K_{\text{road}}^{(\text{end})})$;

  // Update the assigned routes set

  $R \leftarrow R \cup \{r\}$;

return R;

METIS

For each trip request (trips are time-sorted):

- **Forward-Looking Edge Penalization** (FLEP)
  - discourage selection of congested edges

Algorithm 1: METIS

Input : road network G, mobility demand D, penalization factor p, slow factor s

Output : sequence of assigned routes R

// Initialization Phase

$K^{(source)}_{road}, K^{(end)}_{road} \leftarrow KRoadEstimation(G, D)$;

$R \leftarrow \emptyset$;

// Perform the Traffic Assignment (TA)

foreach $j = (o, d, t) \in D$ **do**

    // Apply the Forward-Looking Edge Penalization (FLEP)

    $H \leftarrow FLEP(G, R, D, (p, s), t)$;

    // Generate a set of k candidates on the penalized road network

    $P \leftarrow kMDNSP(H, o, d)$;

    // Select the route that minimizes the score function

    $r \leftarrow RouteSelection(P, K^{(source)}_{road}, K^{(end)}_{road})$;

    // Update the assigned routes set

    $R \leftarrow R \cup \{r\}$;

return R;

METIS

For each trip request (trips are time-sorted):

- **Forward-Looking Edge Penalization** (FLEP)
  - discourage selection of congested edges

- **Alternative Routing**
  - generate routed candidates

Algorithm 1: METIS

Input : road network G, mobility demand D, penalization factor p, slow factor s

Output : sequence of assigned routes R

// Initialization Phase

$K_{\text{road}}^{(\text{source})}, K_{\text{road}}^{(\text{end})} \leftarrow KRoadEstimation(G, D)$;

$R \leftarrow \emptyset$;

// Perform the Traffic Assignment (TA)

foreach $j = (o, d, t) \in D$ **do**

// Apply the Forward-Looking Edge Penalization (FLEP)

$H \leftarrow FLEP(G, R, D, (p, s), t)$;

// Generate a set of k candidates on the penalized road network

$P \leftarrow kMDNSP(H, o, d)$;

// Select the route that minimizes the score function

$r \leftarrow RouteSelection(P, K_{\text{road}}^{(\text{source})}, K_{\text{road}}^{(\text{end})})$;

// Update the assigned routes set

$R \leftarrow R \cup \{r\}$;

return R;

METIS

For each trip request (trips are time-sorted):

- **Forward-Looking Edge Penalization** (FLEP)
  - discourage selection of congested edges
- **Alternative Routing**
  - generate routed candidates
- **Route Scoring**
  - rank routes based on popularity and capacity

Algorithm 1: METIS

Input : road network G, mobility demand D, penalization factor p, slow factor s

Output : sequence of assigned routes R

// Initialization Phase

$K_{road}^{(source)}, K_{road}^{(end)} \leftarrow KRoadEstimation(G, D);$

$R \leftarrow \emptyset;$

// Perform the Traffic Assignment (TA)

foreach $j = (o, d, t) \in D$ **do**

  // Apply the Forward-Looking Edge Penalization (FLEP)

  $H \leftarrow FLEP(G, R, D, (p, s), t);$

  // Generate a set of k candidates on the penalized road network

  $P \leftarrow kMDNSP(H, o, d);$

  // Select the route that minimizes the score function

  $r \leftarrow RouteSelection(P, K_{road}^{(source)}, K_{road}^{(end)});$

  // Update the assigned routes set

  $R \leftarrow R \cup \{r\};$

return R;

Forward-Looking Edge Penalization (FLEP)

- **Penalizing road edges** weight reflects dynamic changes in travel time due to traffic volume
    - Existing methods penalize the entire routes assigned to vehicles
    - FLEP estimates vehicle current position applying penalties to the un-visited edges only

Algorithm 1: METIS

Input : road network G, mobility demand D, penalization factor p, slow factor s

Output : sequence of assigned routes R

// Initialization Phase

$K_{\text{road}}^{(\text{source})}, K_{\text{road}}^{(\text{end})} \leftarrow KRoadEstimation(G, D)$;

$R \leftarrow \emptyset$;

// Perform the Traffic Assignment (TA)

foreach $j = (o, d, t) \in D$ **do**

// Apply the Forward-Looking Edge Penalization (FLEP)

$H \leftarrow FLEP(G, R, D, (p, s), t)$;

// Generate a set of k candidates on the penalized road network

$P \leftarrow kMDNSP(H, o, d)$;

// Select the route that minimizes the score function

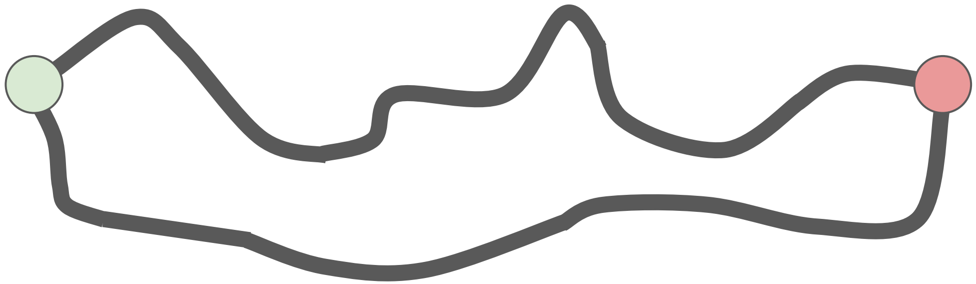
$r \leftarrow RouteSelection(P, K_{\text{road}}^{(\text{source})}, K_{\text{road}}^{(\text{end})})$;

// Update the assigned routes set

$R \leftarrow R \cup \{r\}$;

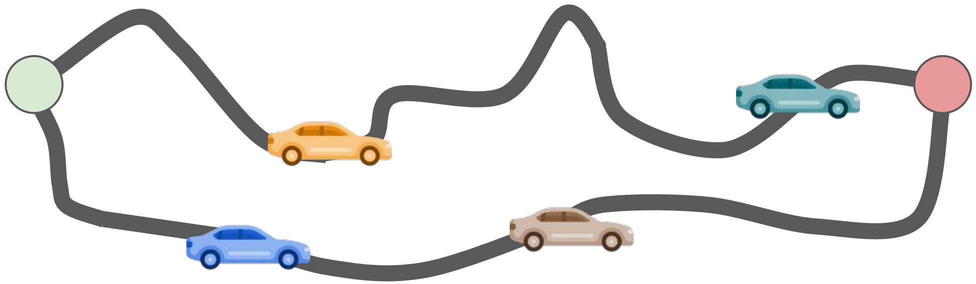
return R;

Forward-Looking Edge Penalization (FLEP)



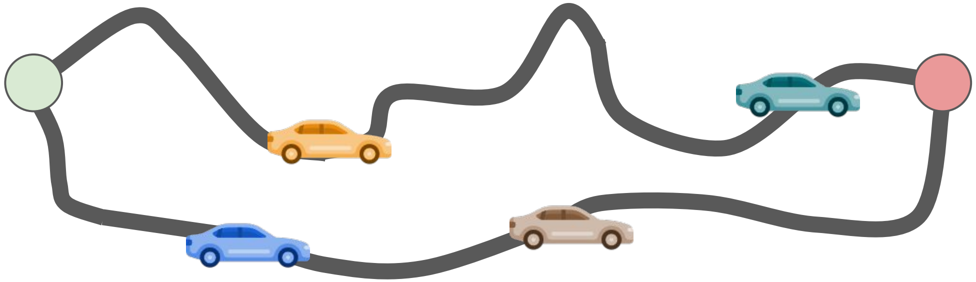
Forward-Looking Edge Penalization (FLEP)

The position of already departed vehicles is **estimated**.



Forward-Looking Edge Penalization (FLEP)

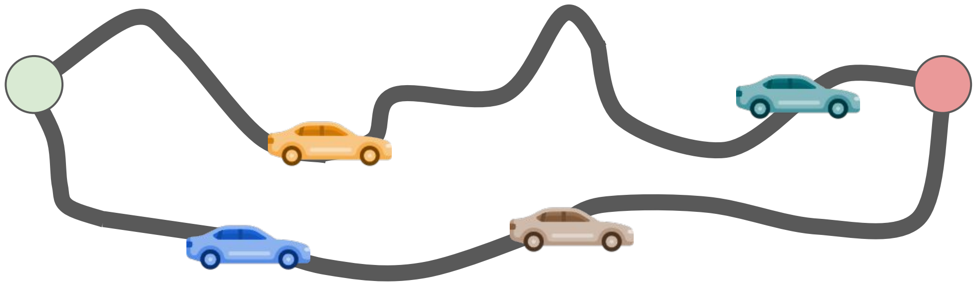
The position of already departed vehicles is **estimated**.



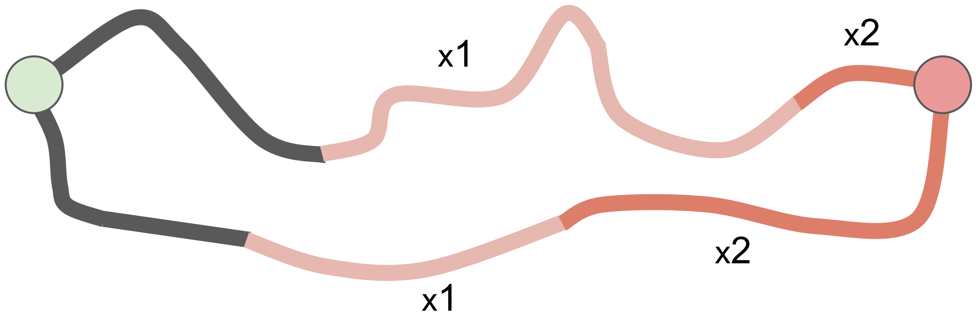
For each vehicle we **penalize** the edges **projected** to be visited by that vehicle

Forward-Looking Edge Penalization (FLEP)

The position of already departed vehicles is **estimated**.



The penalized road network at the current time



Alternative Routing

After the FLEP phase:

- Apply **kMD** [1] on the penalized road network to obtain k (=3) alternative routes

Algorithm 1: METIS

Input : road network G, mobility demand D, penalization factor p, slow factor s

Output : sequence of assigned routes R

// Initialization Phase

$K^{(source)}_{road}, K^{(end)}_{road} \leftarrow KRoadEstimation(G, D)$;

$R \leftarrow \emptyset$;

// Perform the Traffic Assignment (TA)

foreach $j = (o, d, t) \in D$ **do**

  // Apply the Forward-Looking Edge Penalization (FLEP)

  $H \leftarrow FLEP(G, R, D, (p, s), t)$;

  // Generate a set of k candidates on the penalized road network

  $P \leftarrow kMDNSP(H, o, d)$;

  // Select the route that minimizes the score function

  $r \leftarrow RouteSelection(P, K^{(source)}_{road}, K^{(end)}_{road})$;

  // Update the assigned routes set

  $R \leftarrow R \cup \{r\}$;

return R;

[1] Christian Häcker, Panagiotis Bouros, Theodoros Chondrogiannis, and Ernst Althaus. 2021. Most Diverse Near-Shortest Paths. In ACM SIGSPATIAL GIS. 229–239

Route Scoring

After the route generation:

1. Compute a score for each route (based on **K-Road** [2]) that favours high-capacity roads and disfavour popular ones
2. Select the route with the minimum score

$$\frac{\overline{KR_s} \cdot \overline{KR_d}}{\overline{C}}$$

Algorithm 1: METIS

Input : road network G, mobility demand D, penalization factor p, slow factor s
Output : sequence of assigned routes R

// Initialization Phase
$K^{(\text{source})}_{\text{road}}, K^{(\text{end})}_{\text{road}} \leftarrow KRoadEstimation(G, D)$;
$R \leftarrow \emptyset$;

// Perform the Traffic Assignment (TA)
foreach $j = (o, d, t) \in D$ **do**
  // Apply the Forward-Looking Edge Penalization (FLEP)
  $H \leftarrow FLEP(G, R, D, (p, s), t)$;
  // Generate a set of k candidates on the penalized road network
  $P \leftarrow kMDNSP(H, o, d)$;
  // Select the route that minimizes the score function
  $r \leftarrow RouteSelection(P, K^{(\text{source})}_{\text{road}}, K^{(\text{end})}_{\text{road}})$;
  // Update the assigned routes set
  $R \leftarrow R \cup \{r\}$;
return R;

Wang, P. et al. **Understanding Road Usage Patterns in Urban Areas.** *Scientific Reports* 2, 1001 (2012)

Route Scoring

$$K^{(\text{source})}_{\text{road}}(e_1) = 2$$

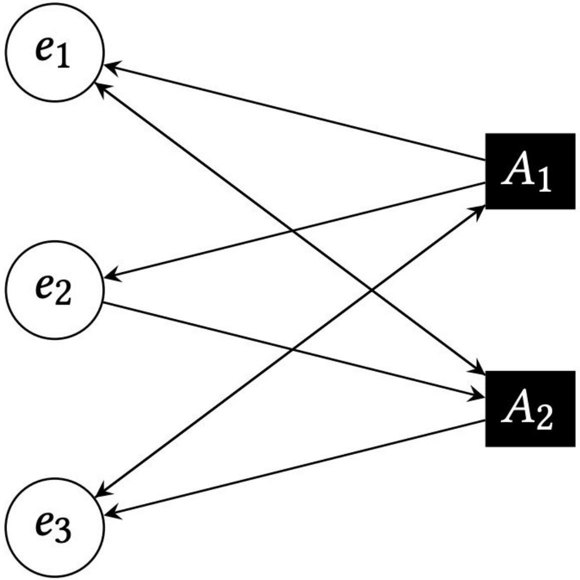
$$K^{(\text{end})}_{\text{road}}(e_1) = 1$$

$$K^{(\text{source})}_{\text{road}}(e_2) = 1$$

$$K^{(\text{end})}_{\text{road}}(e_2) = 1$$

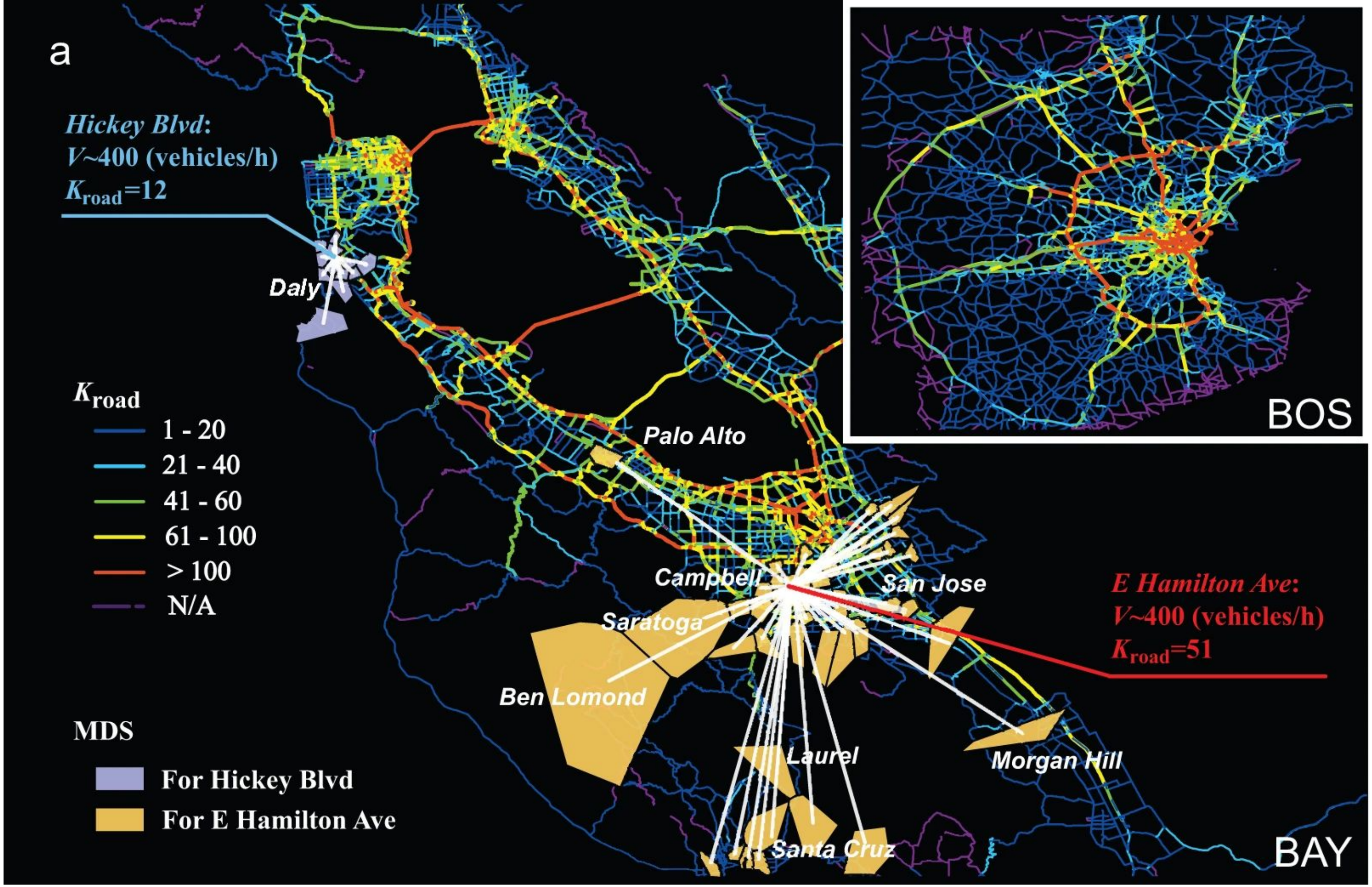
$$K^{(\text{source})}_{\text{road}}(e_3) = 2$$

$$K^{(\text{end})}_{\text{road}}(e_3) = 1$$



$$\frac{\overbrace{\overline{KR_s} \cdot \overline{KR_d}}^{\textit{Route Popularity}}}{\underbrace{\overline{C}}_{\textit{Route capacity}}}$$

Wang, P. et al. **Understanding Road Usage Patterns in Urban Areas.** *Scientific Reports* 2, 1001 (2012)



Wang, P. et al. **Understanding Road Usage Patterns in Urban Areas.** *Scientific Reports* 2, 1001 (2012)

Evaluation Metrics

We can characterize the paths generate by the TA algorithms with several metrics.

For example:

- Road Coverage
- Redundancy
- Time Redundancy
- CO2 emissions

Road Coverage (RC)

length of edge

- Given a set of routes R, and their edges

$$S_R = \bigcup_{p \in R} \{e \in p\}$$

$$RC(R) = \frac{\sum_{e \in S_R} l(e)}{L(E)} \cdot 100$$

$$L(E) = \sum_{e \in E} l(e)$$

RC characterizes road infrastructure usage:

- A higher road coverage indicates a larger proportion of the G being utilized

Redundancy

- Given a set of routes R, and their edges

$$S_R = \bigcup_{p \in R} \{e \in p\} \qquad \mathrm{Red}(R) = \frac{\sum_{p \in R} |p|}{|S_R|}$$

If $Red(R)$ = 1, there is no overlap among the routes in R, while $Red(R) = |R|$ when all routes are identical

- average utilization of edges that appear in at least one route

Time Redundancy (RED)

- Given a set of routes R, their edges, and a time window t

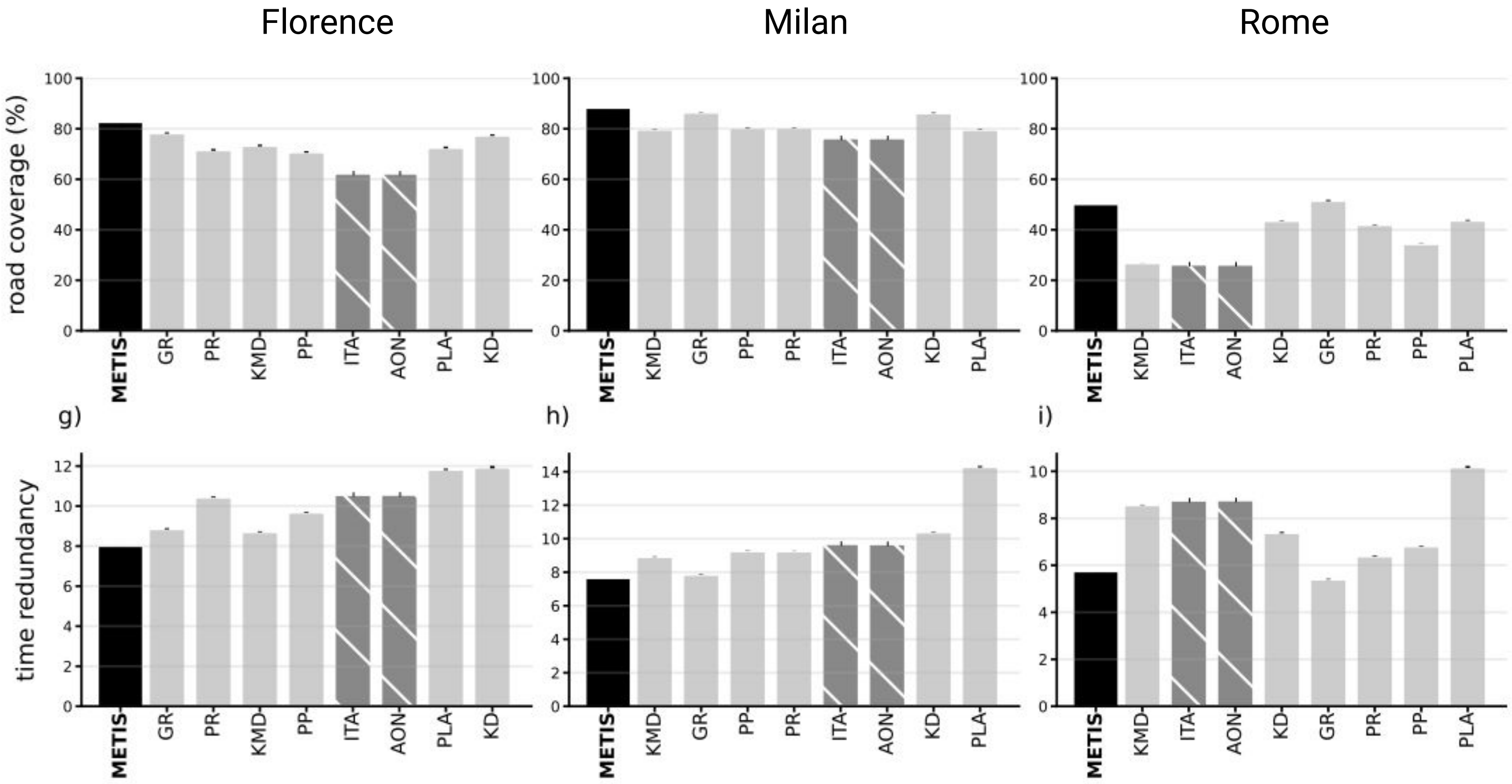
$$S_R = \bigcup_{p \in R} \{e \in p\} \qquad RED(R,t) = \frac{1}{|I|} \sum_{i \in I} \mathrm{RED}(R_{i,t})$$

$$I = \{t_0, t_0 + \sigma, t_0 + 2\sigma, \ldots, t_{max}\}$$

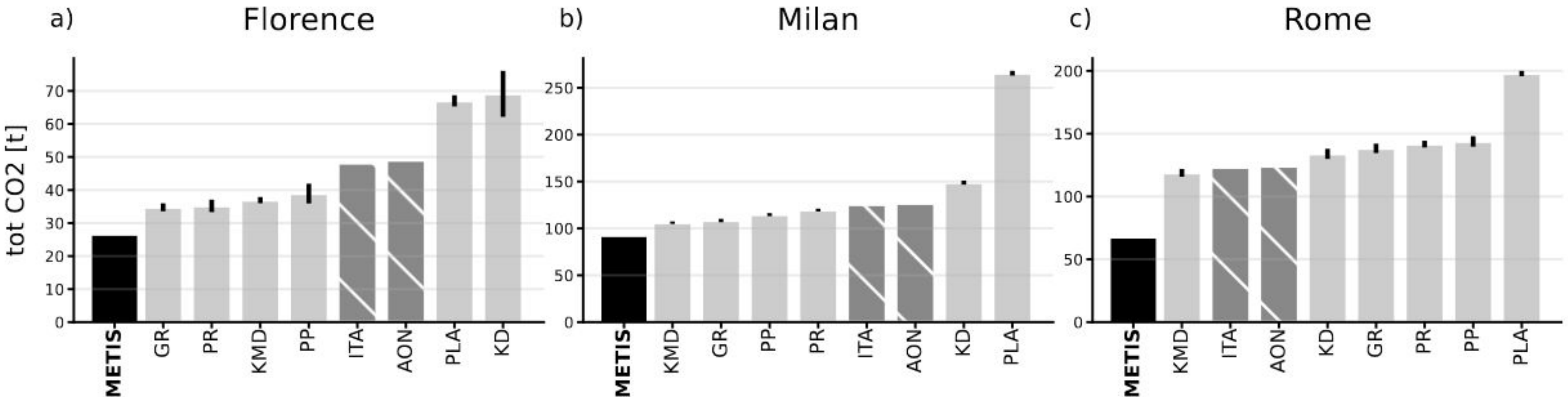
RED(Ri,t) is the Red of trips in R departed within time interval $[i, i + t)$.

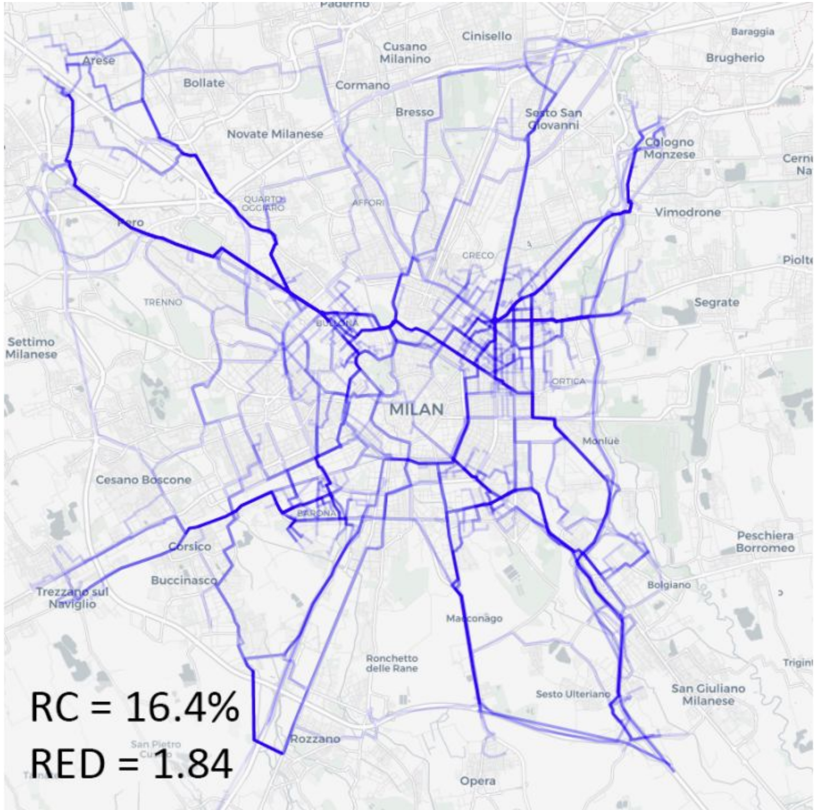
- Low $RED(R, t)$ indicates that routes close in time are better distributed across edges

Characterization Metrics

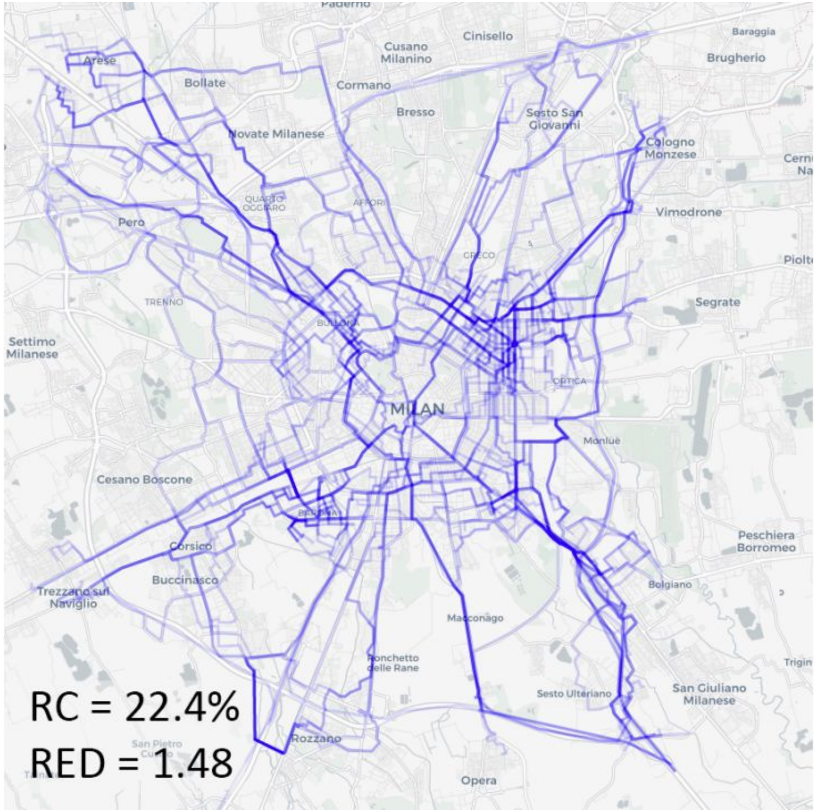


Characterization Metrics





KMD



METIS

INTERVALLO

Satellite Navigation Services: What Impact?

The New York Times

Navigation Apps Are Turning Quiet Neighborhoods Into Traffic Nightmares

The corner of Fort Lee Road and Broad Avenue in Leonia, N.J. With traffic apps suggesting shortcuts for commuters through the borough, officials have decided to take a stand. Bryan Anselm for The New York Times

CBC | MENU

radio | Top Stories | Shows | Podcasts | Schedules | Frequency | Listen Live

The Current

Overuse of GPS navigation shrinks part of the brain, says researcher

Home > News > Technology

Navigation App Sends People Looking to Get Tested to Family Farm

Published: 23 Dec 2020, 06:01 UTC • By: Bogdan Popa

While navigation apps are most of the time pretty accurate, errors do tend to happen. In some cases, they could end up becoming fatal for the drivers who follow their guidance blindly.

Verdict

News | Analysis | Sectors | Themes | Insights | Events | Reports | Premium Insights

Disruptor | Hardware | Internet & Media | IT Services | Telecom

April 18, 2019

Almost all young people would be lost without their sat navs, survey finds

By Ellen Daniel

SUBSCRIBE & SAVE Less than $3 per week

THE WEEK

Home | Daily Briefing | Talking Points | The Week Recommends | Newsletters | Cartoons | From the Magazine | More | The Week Junior

Tech

8 drivers who blindly followed their GPS into disaster

Take note: The machine does not always know where it's going

DRIVE | News | Reviews | Guides | Car of the Year | New Car Showroom

News

Sat-nav systems cause accidents: study

NEWS

According to a recent police study, GPS satellite navigation systems have caused a spike in road accidents due to drivers paying attention to the guidance rather than the road. The study says GPS devices distract drivers in a similar way to mobile phones.

Mirror

Sat Nav (Pic:PA)

NEWS | POLITICS | FOOTBALL | CELEBS | TV | CHOICE

SatNav danger revealed: Navigation device blamed for causing 300,000 crashes

Road safety chiefs last night warned of the dangers of satnavs after figures blaming 300,000 smashes and near prangs.

All mapped out? Using satnav 'switches off' parts of the brain, study suggests

Brain activity linked to simulating possible journeys appears to be absent when a person is following directions rather than independently planning a route

www.salernonotizie.it

HOME | CAMPANIA | SALERNO | SPORT | PROVINCIA | RUBRICHE | SPETTACOLI | R. STAMPA | ALTRO

Il gps fa sbagliare strada e l'auto si rompe: vacanze da incubo

1 Agosto 2023 | 1 Min. di lettura

WalesOnline | NEWS | RUGBY | FOOTBALL | IN YOUR AREA | WHAT'S ON | MORE

BUY A PAPER | FUNERAL NOTICES | JOBS | VOUCHER CODES | DIRECTORY | ADVERTISE WITH US | BOOK AN AD | MARKETPLACE | BUY A PHOTO | NEWSLETTER SIGNUP

News › Latest Wales News › Pembrokeshire

Driver jams car in narrow footpath after relying on sat-nav

The white Nissan Juke was stuck for a week after the two visitors to the seaside town went the wrong way

NEWS By Wales News Service & Branwen Jones Reporter

07:41, 9 OCT 2023

Bookmark

Vox | EXPLAINERS | CROSSWORD | VIDEO | PODCASTS | POLITICS | POLICY | CULTURE | SCIENCE | MORE | Give

TECHNOLOGY | SCIENCE

Have we become too reliant on GPS? This satellite expert thinks so.

By Brad Plumer | @bradplumer | brad@vox.com | Apr 10, 2016, 10:00am EDT

Homeworks

- Given the path penalization algorithm, how does the geographic distribution of paths if we apply the logarithmic function on all edge weights?

- Create your own alternative routing algorithm combining concepts seen during the lesson.

to study for the exam

Material

- Shortest-Path Diversification through Network Penalization: A Washington DC Area Case Study
- One-Shot Traffic Assignment with Forward-Looking Penalization
- Comparing Alternative Route Planning Techniques: A Comparative User Study on Melbourne, Dhaka and Copenhagen Road Networks