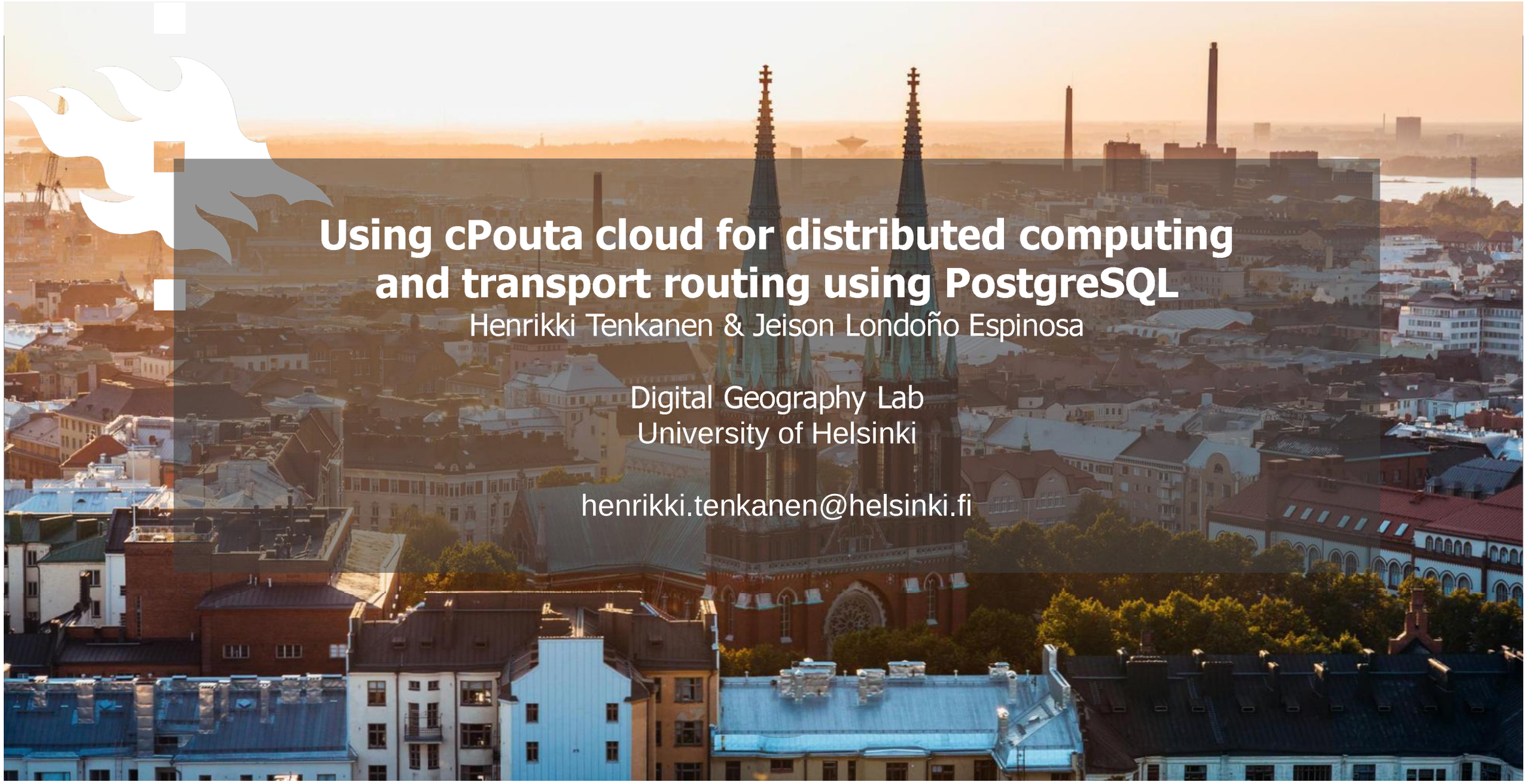


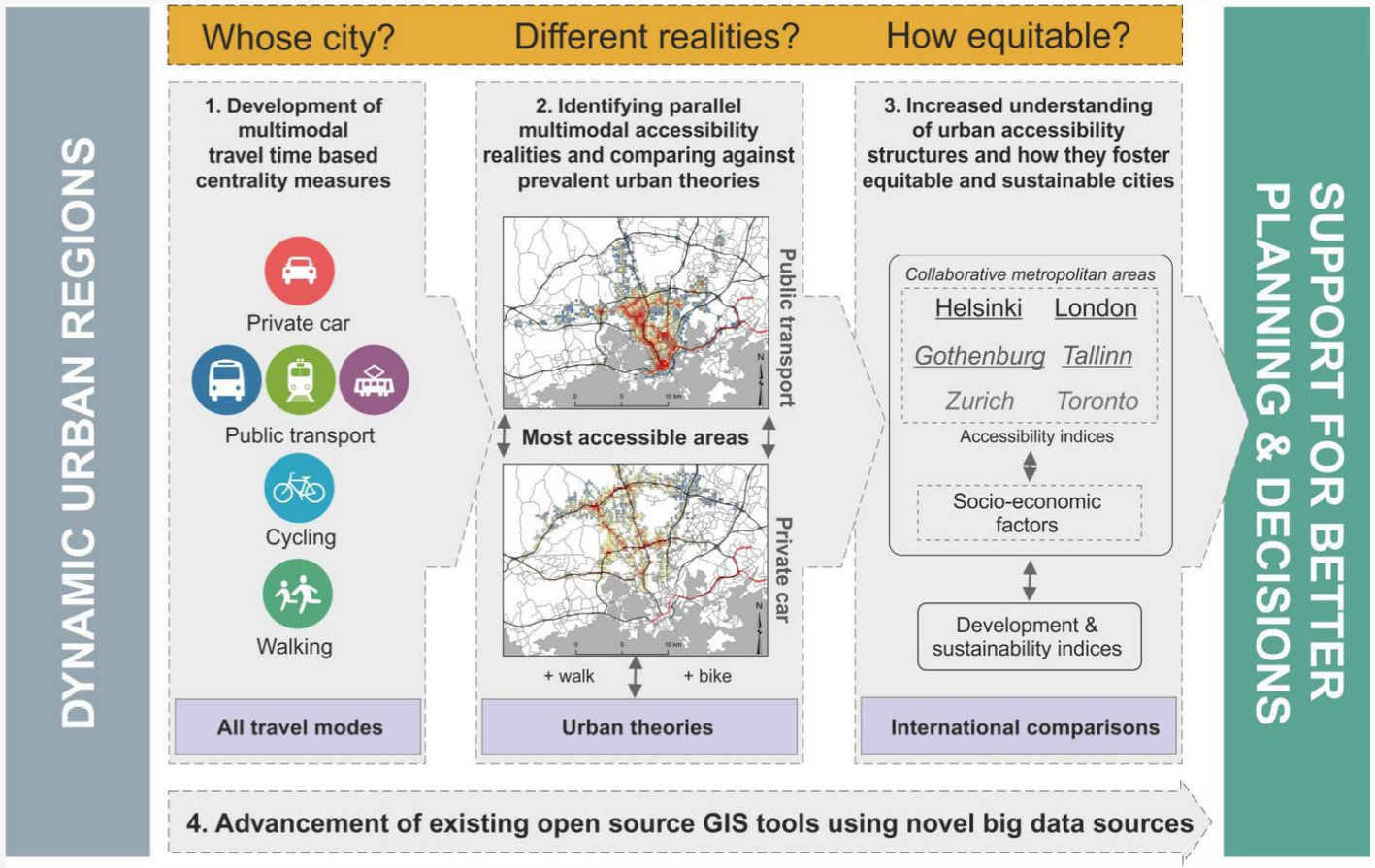
Using cPouta cloud for distributed computing and transport routing using PostgreSQL

Henrikki Tenkanen & Jeison Londoño Espinosa

Digital Geography Lab
University of Helsinki

henrikki.tenkanen@helsinki.fi

STUDY DESIGN



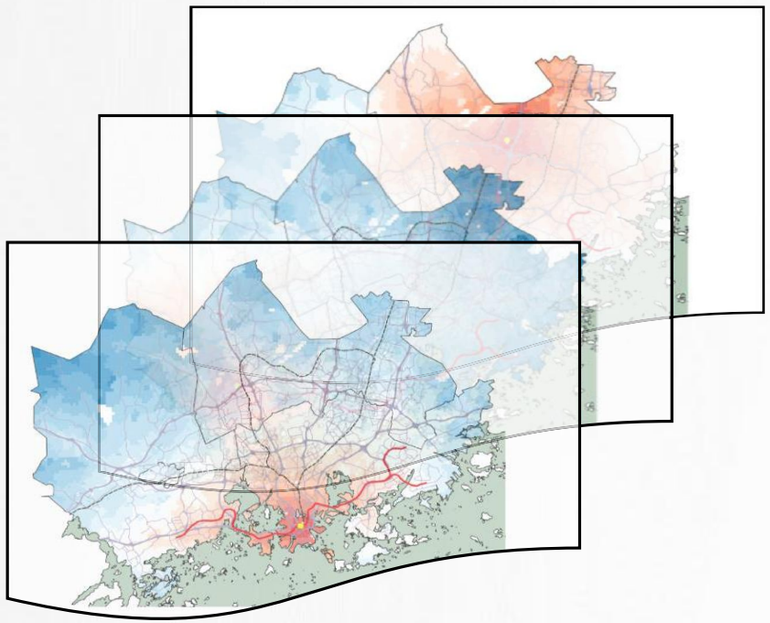
Large scale international comparisons of accessibility and (modal) centrality in urban regions

Computing intensive

Based on travel time matrices

HOW CENTRALITY IS CALCULATED?

INPUT

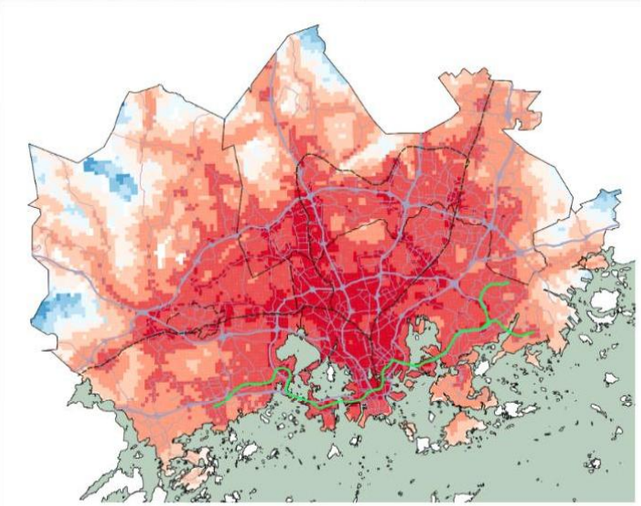


As input we have travel time layers to all locations in the study region

ANALYSIS

Calculation of Median Travel Times as a cross-section of all layers with different travel modes using door-to-door approach

RESULT (centrality layer)

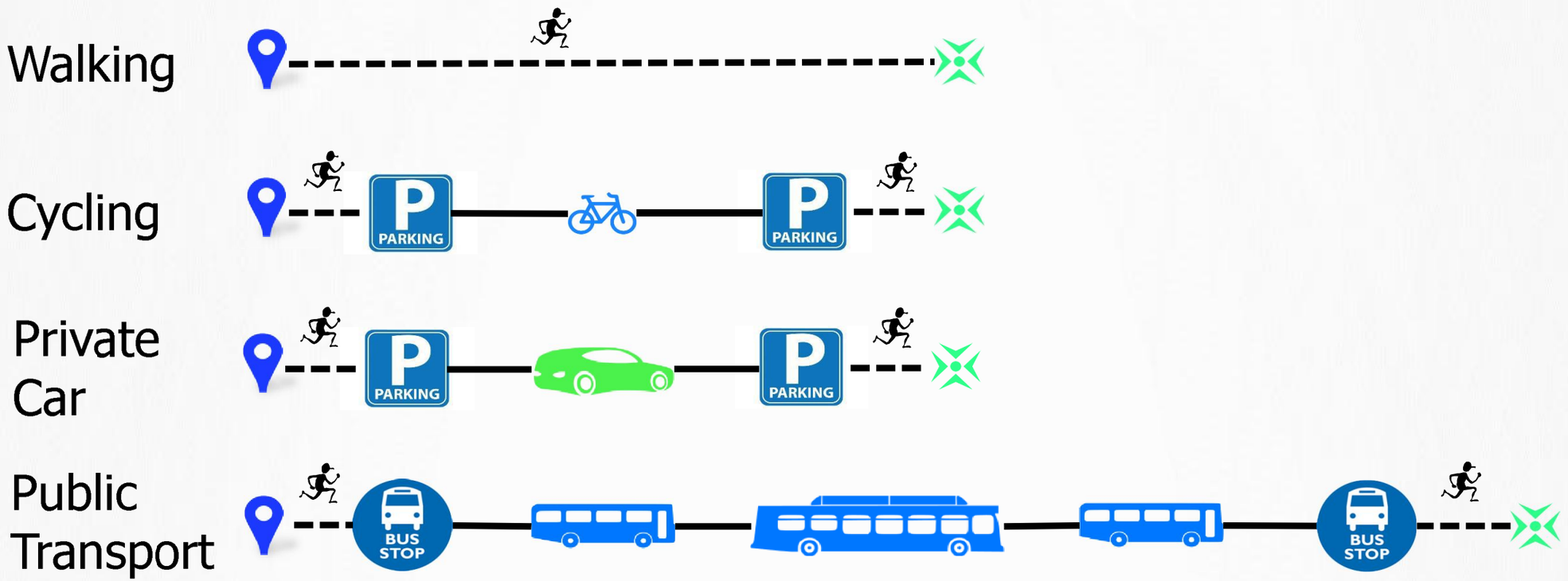


The **more red** the **more central**/lower median time

The **more blue** the **less central**/higher median time

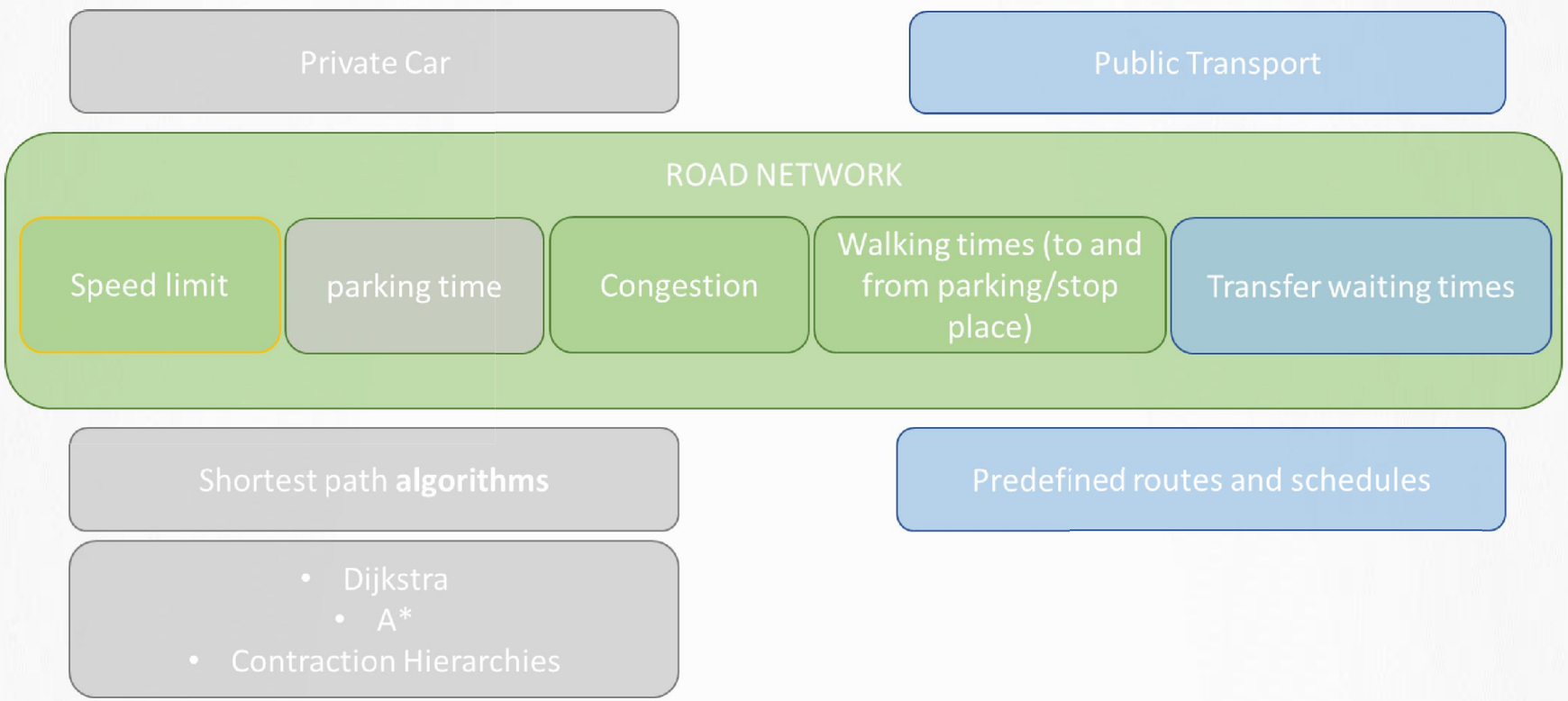
TOOLS

DOOR-TO-DOOR APPROACH



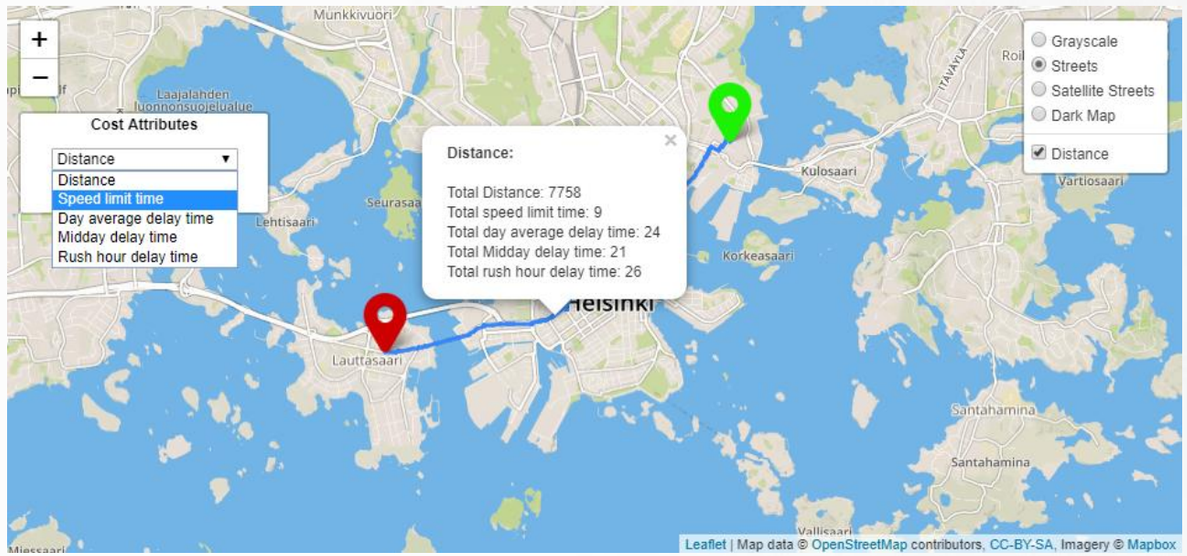
http://www.iconsdb.com

MEASURING TRAVEL TIME



DORA: DOOR-TO-DOOR ROUTING ANALYST

- Open source multi modal routing tool using door-to-door approach
- Works with car, cycling and walking routes
- Based on pgRouting (v2.3.2) extension
- Analyses conducted using cPouta instances

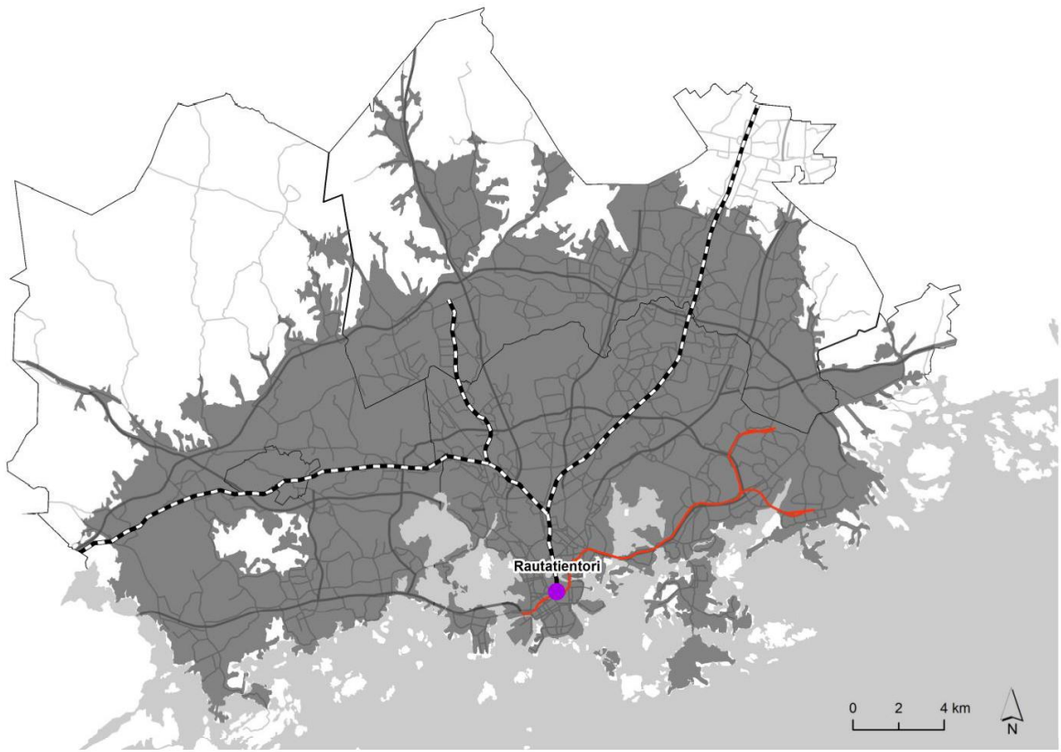


The tool is open source:
https://github.com/DigitalGeographyLab/DORA

MORE REALISTIC MODELLING REQUIRED

Analysis based on:

- Road network geometry + speed limits

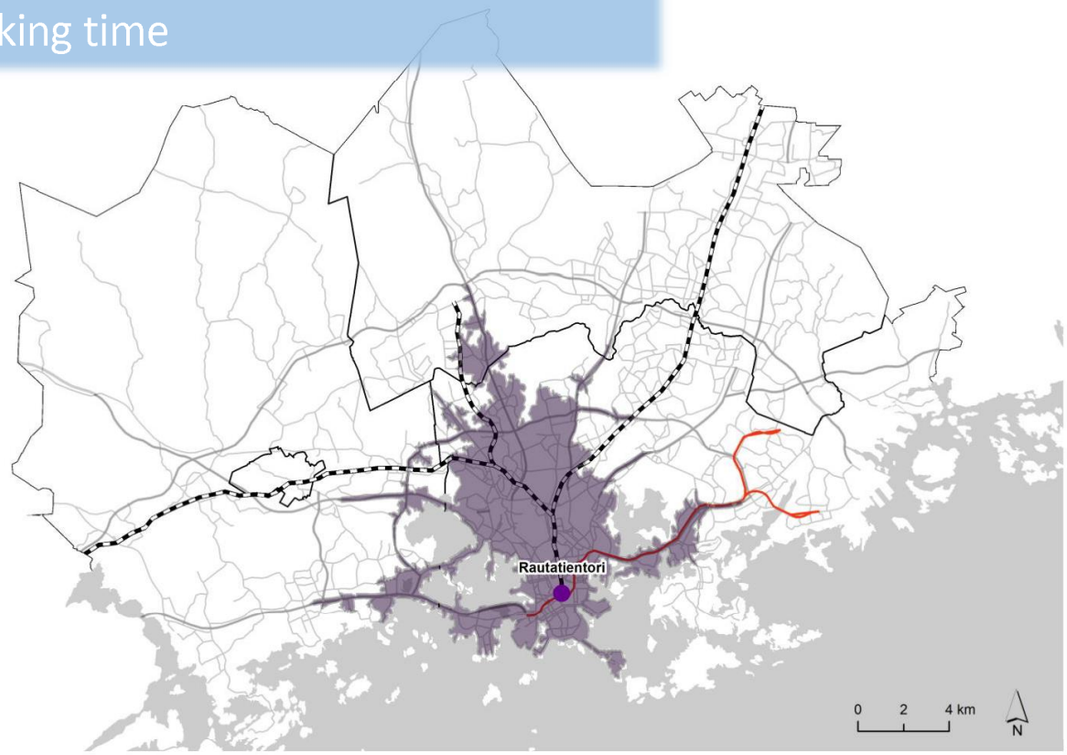


< 20 MIN

MORE REALISTIC MODELLING REQUIRED

Analysis based on:

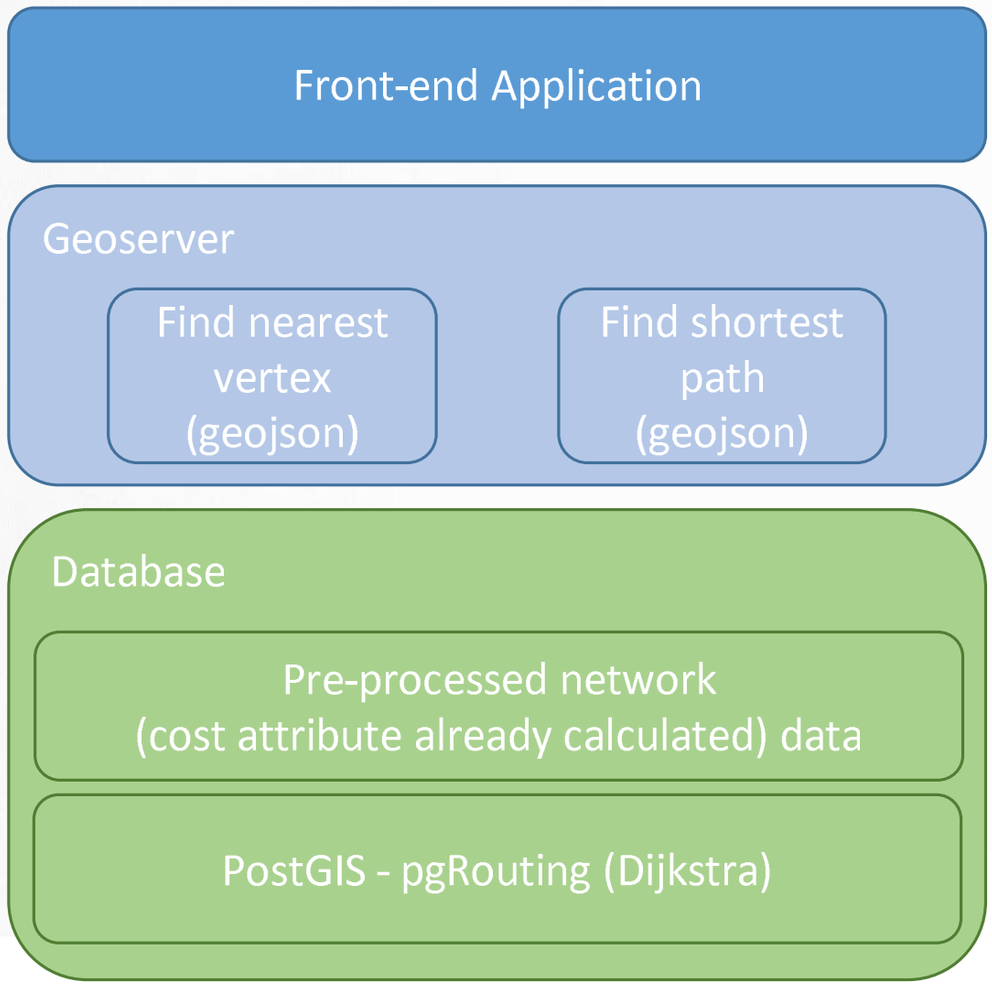
- Road network geometry + speed limits
- deceleration values on cross roads
- estimated parking time



< 20 MIN

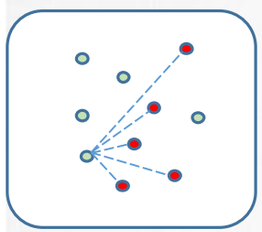
TECH STACK + ANALYSIS APPROACHES

DORA: 1. ROUTE VISUALIZATION



DORA: 2. OD-MATRIX ANALYST

Cost attribute (or all)



Geojson

Car-Routing

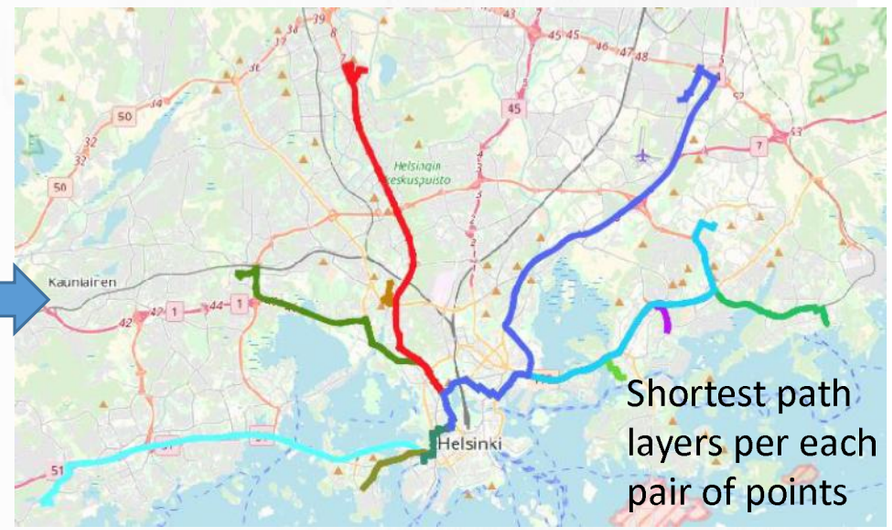
- Merge Additional Layers
- Abstract Additional Operations
  - Euclidean distance
  - Walking time

Geoserver

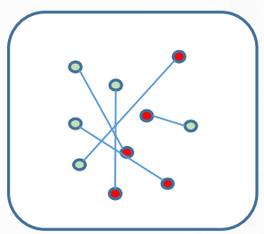
- Find nearest vertex (geojson)
- Find shortest path (geojson)

Database

- OSM road network / Digiroad2
- Door-to-Door function (Calculate cost attributes for the road segments)
- PostGIS - pgRouting (Dijkstra)



Shortest path layers per each pair of points

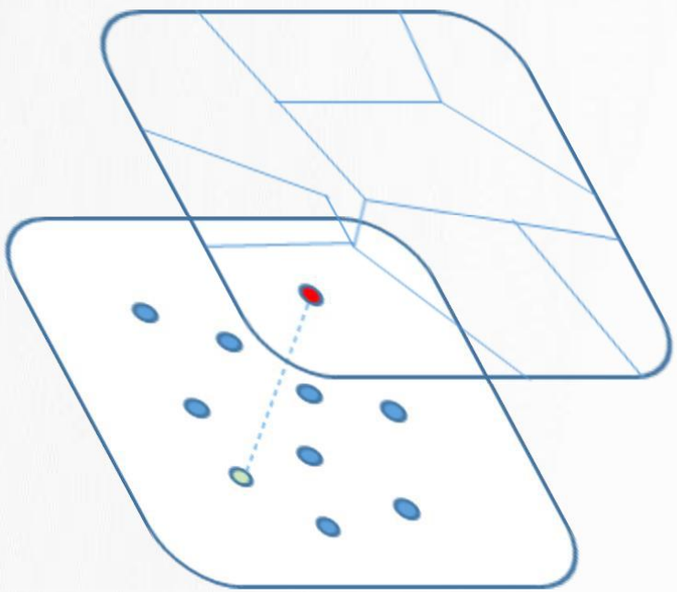


Summary of **total time** for the cost attribute used:

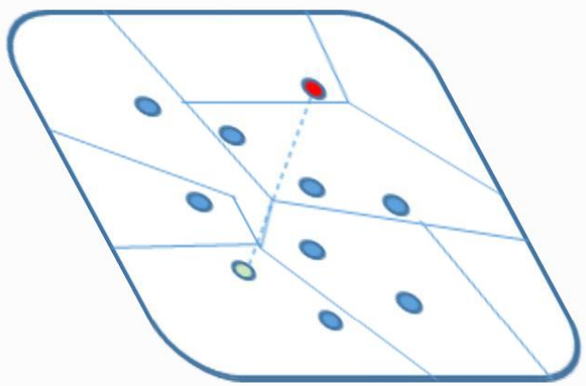
- Distance
- Euclidean Distance (S, E)
- Walking Time (S, E)
- Speed limit-based time
- Day avg delay time
- Midday avg delay time
- Rush hour avg delay time

HELSINGIN YLIOPISTO
HELSINGFORS UNIVERSITET
UNIVERSITY OF HELSINKI

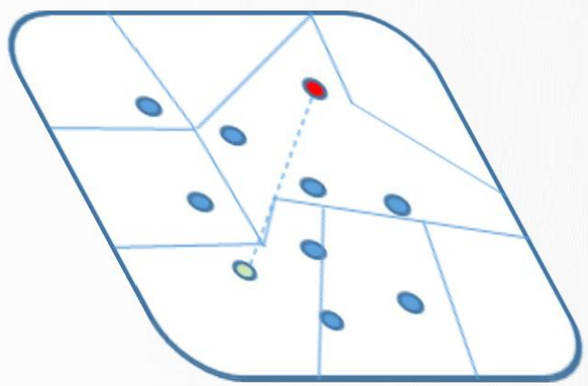
DORA: D2D CONTROL LAYERS



Walking distances



Parking time



DORA: WALKING LEG PROCESSING

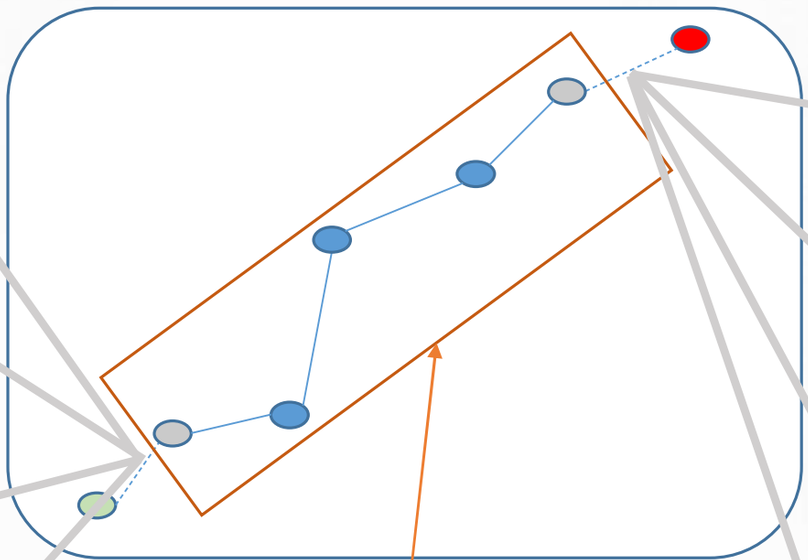
Start point feature

1. Find nearest vertex to the road network of the starting point

3. Calculate the **Euclidean distance** from the start point to its nearest vertex

5. Calculate **walking time** of the Euclidean distance from the first part of the path

7. Add average walking time to the parking place



7. Calculate **shortest path** from the start nearest vertex to the end nearest vertex

End point feature

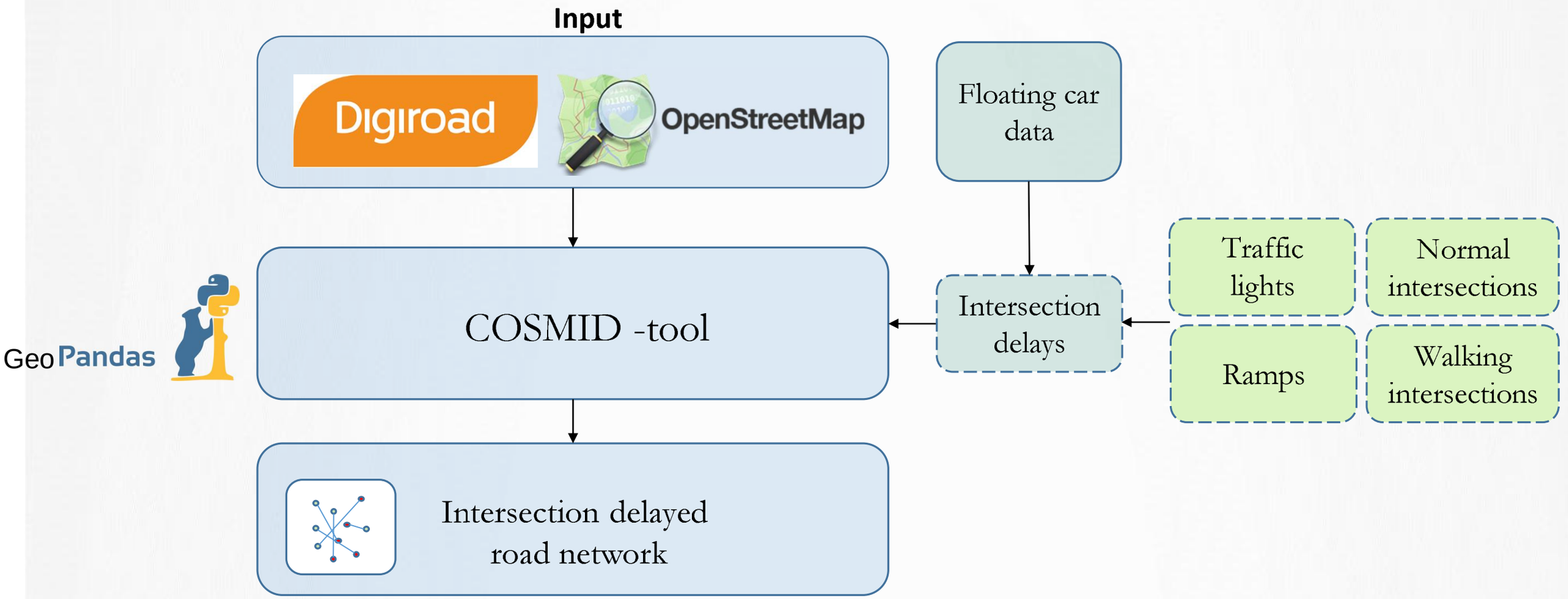
2. Find nearest vertex to the road network of the ending point

4. Calculate the **Euclidean distance** from the end point to its nearest vertex

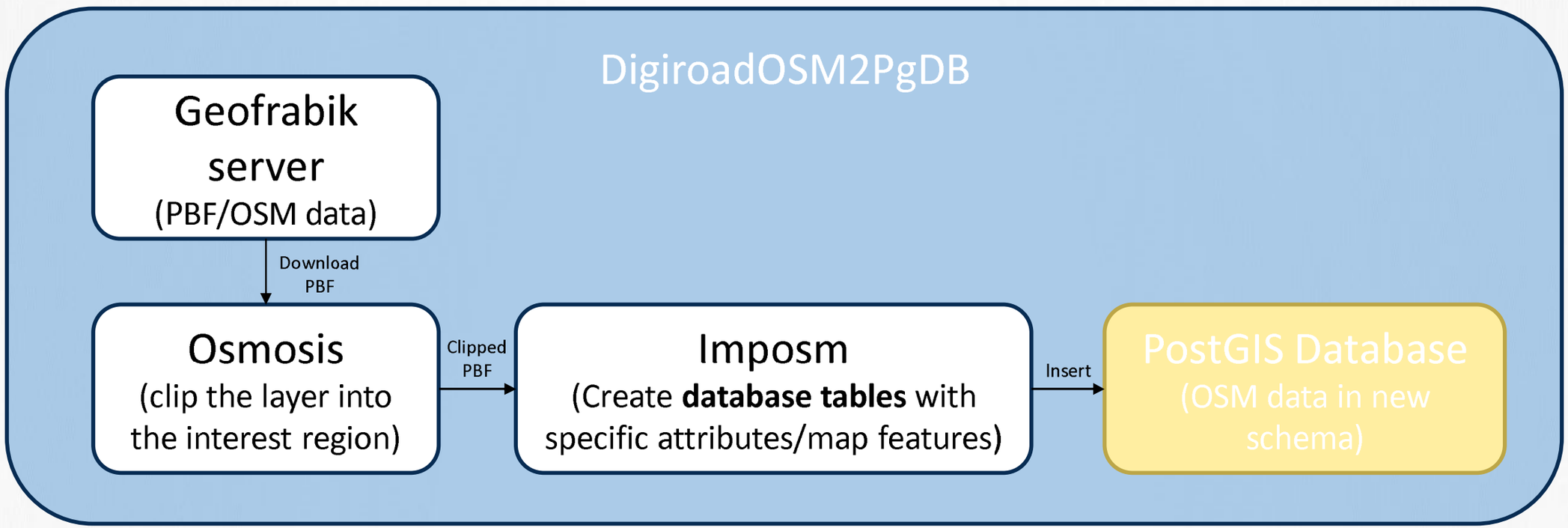
6. Calculate **walking time** of the Euclidean distance from the end part of the path

8. Add the time to search a parking place and average walking time from the parking place to the final destination

COSMID: MODIFYING STREET NETWORKS TO CONSIDER CONGESTION LEVELS



OSM TO POSTGRES



EXPERIENCES

PROS / CONS

Pros:

- Pgrouting has a nice selection of routing algorithms available
- PostgreSQL / PostGIS can handle huge amount of calculations effieciently
- Setting up a system in cPouta is fairly straight forward for a single Postgres node using Docker

Cons:

- The idea was to distribute calculations using PostgresXL – turned out NOT to be possible when we tested because PostgresXL misses some key features required by pgrouting
- Setting up a database based system has a lot of overhead compared to e.g. implementation build with pure Python (using networkX package, slower but easily distributable).


THANK YOU!
@tenkahen
henrikki.tenkanen@helsinki.fi