Lower bound computation for SecureRide

A cheat-proof ride-hailing service

Privacy Protection

Emanuele Bugliarello
emanuele.bugliarello@epfl.ch
(SC-MA3)

Tutors: Anh Pham, Italo Dacosta

January 27, 2017

Motivation

- **Ride-hailing services are popular**
  e.g., Uber is operating in more than 500 cities in the world

- **Service providers rely on location information reported by the drivers' smartphones to compute the fares**
  a dishonest driver might try to manipulate the GPS information to over-charge the riders

- **It is important to have a mechanism to guarantee the integrity of the fares of the trips**

Goal

Provide stronger integrity guarantees for the distance of the trip without relying on the smartphone of the driver

Approach

- Provide *lower-bound and upper-bound distance for a trip*, given its source, destination and duration.
- We rely on:
  - Access Point-based Location Proofs
  - Street-map networks

Related work

1. A. Pham, K. Huguenin, I. Bilogrevic, I. Dacosta, and J.-P. Hubaux.
   SecureRun: Cheat-Proof and Private Summaries for Location-Based Activities.
   Transactions on Mobile Computing, 2015.

2. A. Pham, I. Dacosta, B. Jacot-Guillarmod, K. Huguenin, T. Hajar, F. Tramèr, V. Gligor, and J.-P. Hubaux
   PrivateRide: A Privacy-Enhanced Ride-Hailing Service
   Proceedings on Privacy Enhancing Technologies

3. L. Magarshack.
   SecureRide: A cheat-proof ride-hailing service.
   Master thesis, 2016.

4. J. Balasch, A. Rial, C. Troncoso and C. Geuens.
   PrETP: Privacy-Preserving Electronic Toll Pricing.
   USENIX Security, 2010.

5. S. Saroiu and A. Wolman.
   Enabling new mobile applications with location proofs.
   ACM HotMobile, 2009.

Actors & threat model

- **Drivers.** Assumed malicious: can spoof their locations to increase the computed fare.

- **Riders.** Assumed malicious: can attack the driver's smartphone to send wrong location updates so reduce the computed fare.

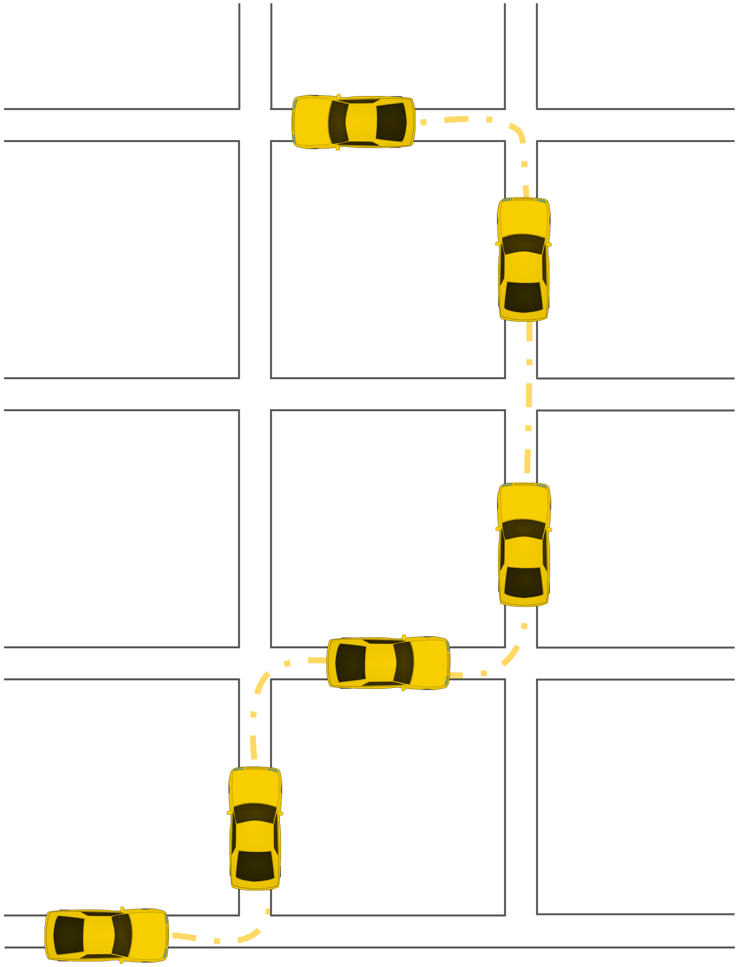
- **Service providers.** Assumed honest: it is in their interest to avoid overcharging riders, they would opt for other service providers.

- **Access point operators.** Assumed honest: APs are assumed to follow the protocol specified in our solution. In particular, they are assumed not to collude with drivers.

Lower bound distance

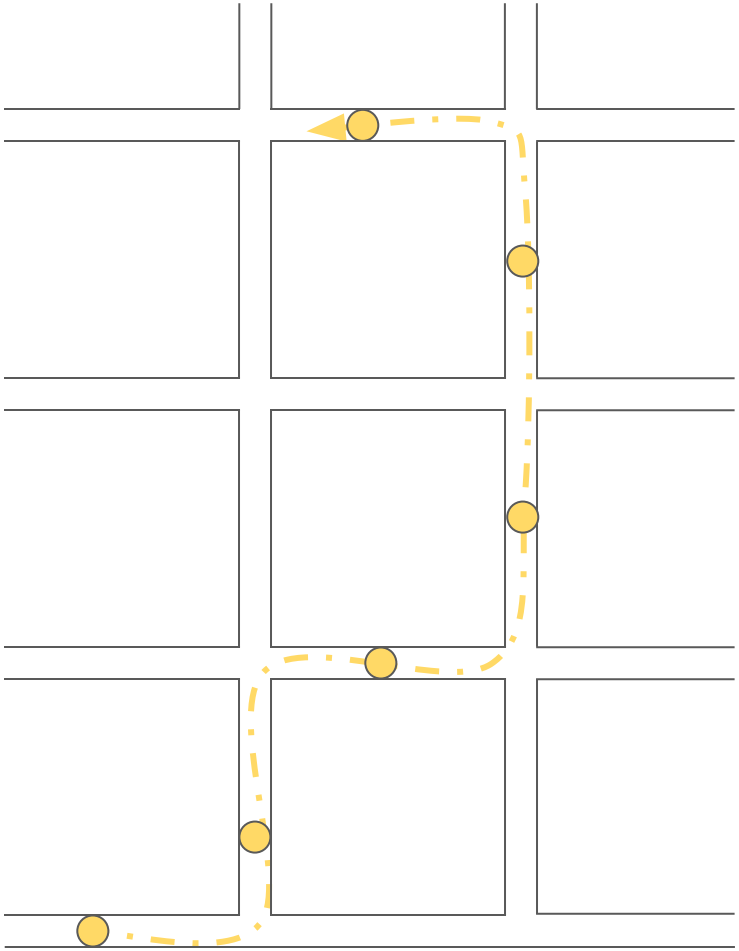
Current scenario (1)

Driver periodically sends GPS coordinates to the SP.



Current scenario (2)

Driver periodically sends GPS coordinates to the SP.



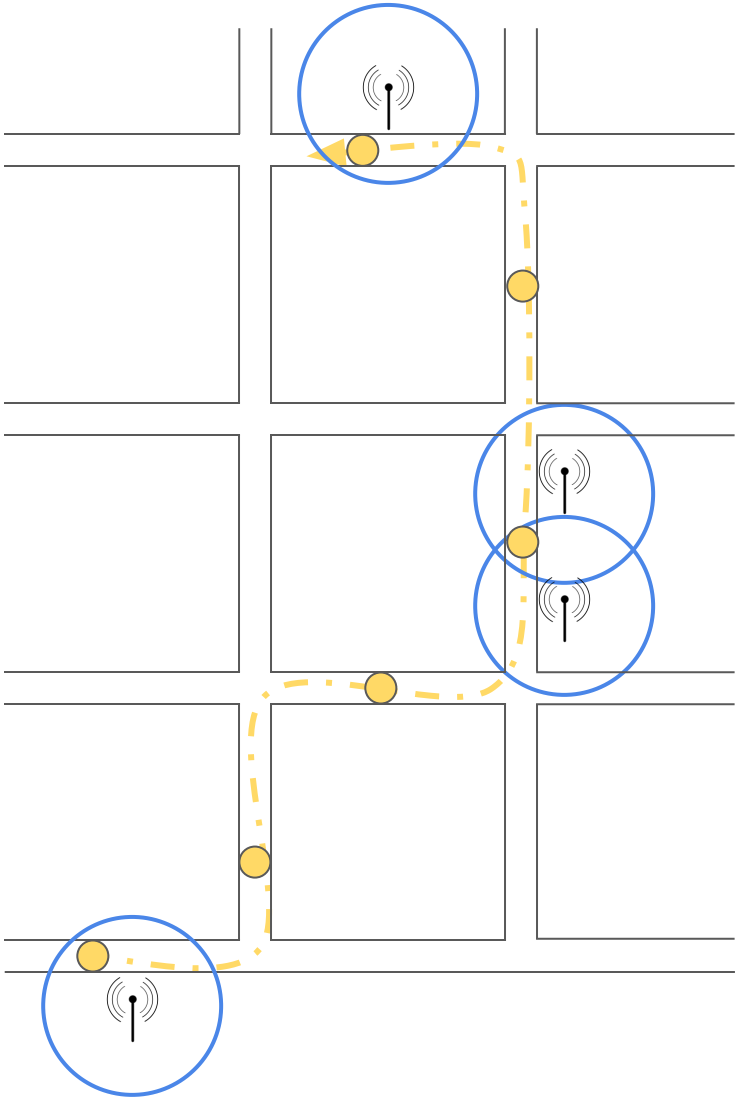
Overview of our solution (1)

Driver collects Location Proofs from unit-disc Access Points.

- A location proof is a piece of data that certifies a geographical location [4]

SP defines location proof areas.

SP computes the lower bound of the distance by summing up the shortest paths between any two consecutive location proof areas.



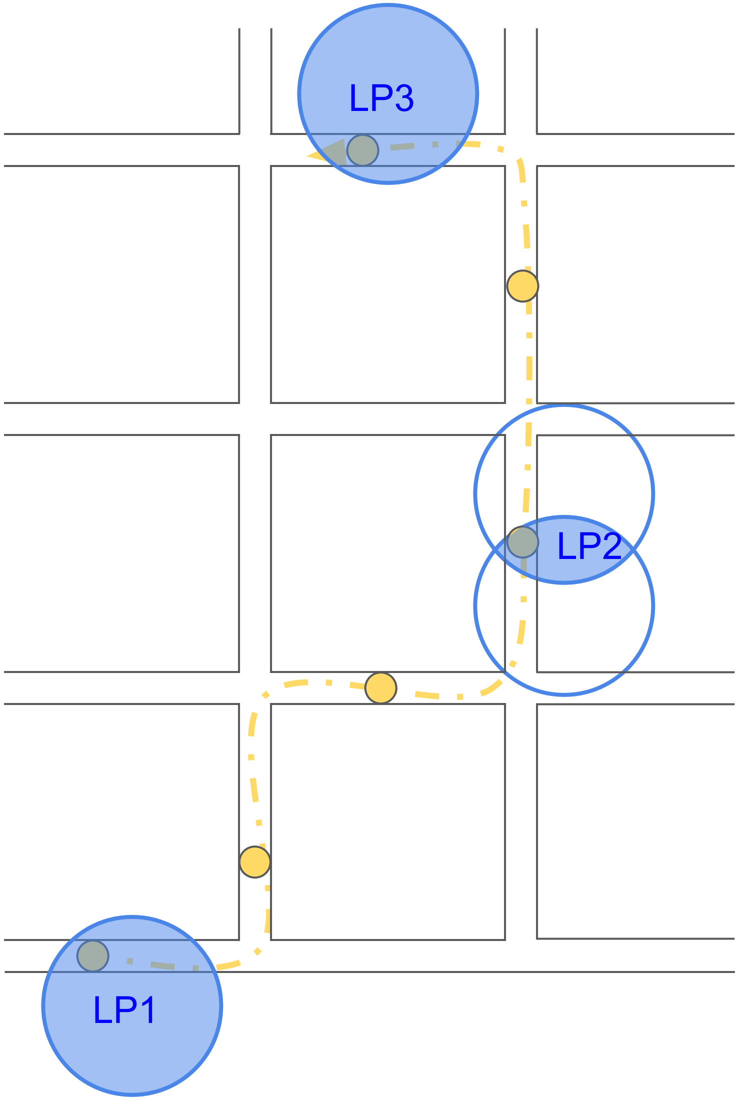
Overview of our solution (2)

Driver collects Location Proofs from unit-disc Access Points.

- A location proof is a piece of data that certifies a geographical location [4]

SP defines location proof areas.

SP computes the lower bound of the distance by summing up the shortest paths between any two consecutive location proof areas.



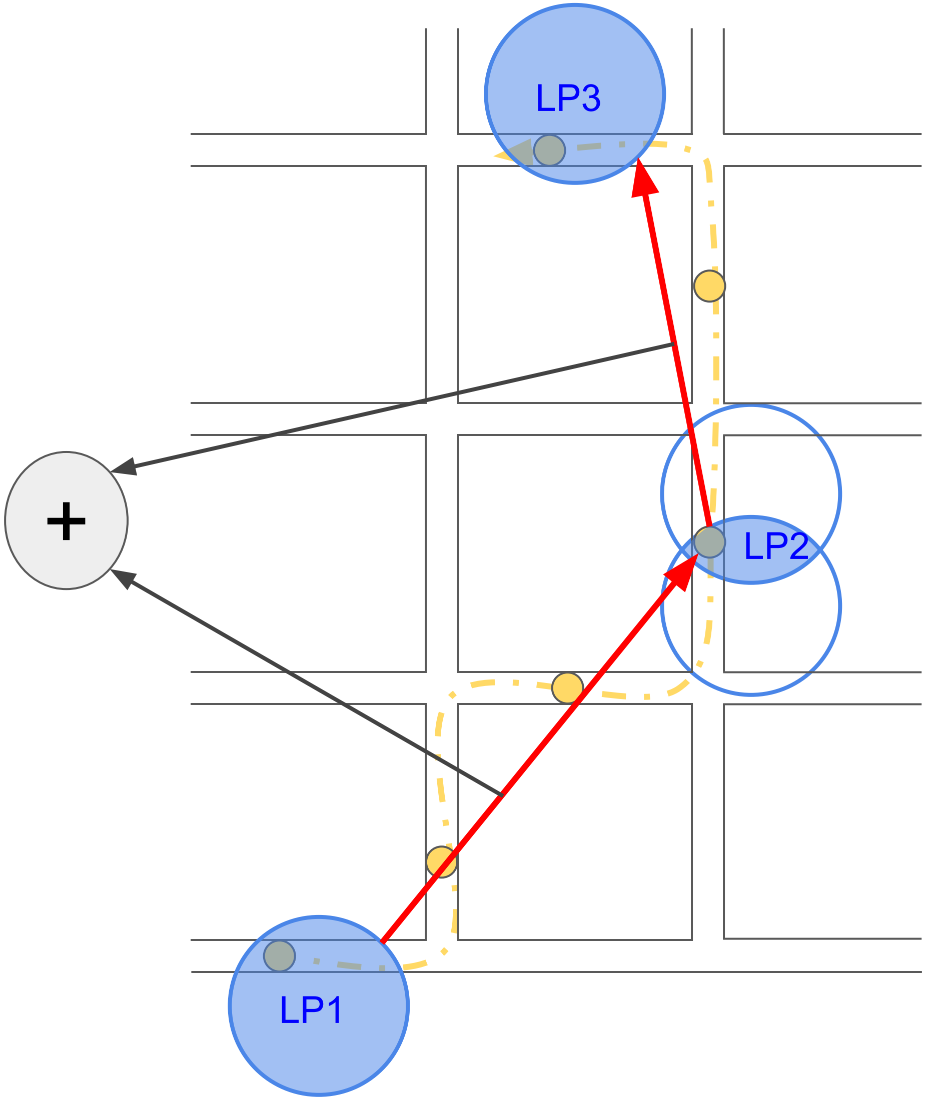
Overview of our solution (3)

Driver collects Location Proofs from unit-disc Access Points.

- A location proof is a piece of data that certifies a geographical location [4]

SP defines location proof areas.

SP computes the lower bound of the distance by summing up the shortest paths between any two consecutive location proof areas.



Achieving tighter bounds (1)

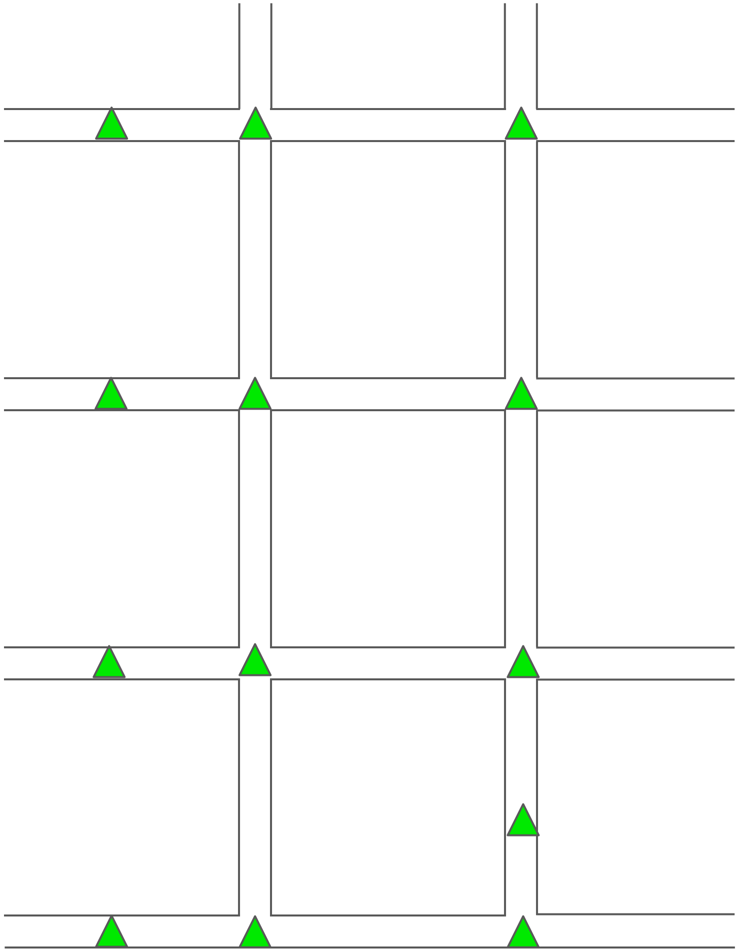
Street-map networks:

- Nodes (at least) at street intersections
- Additional information for each street (e.g., if one-way only, maximum speed)

SP uses street maps to compute the distance between LPs.

Add nodes every 0.5 seconds.

Add nodes at intersections of LPs and streets.



Achieving tighter bounds (2)

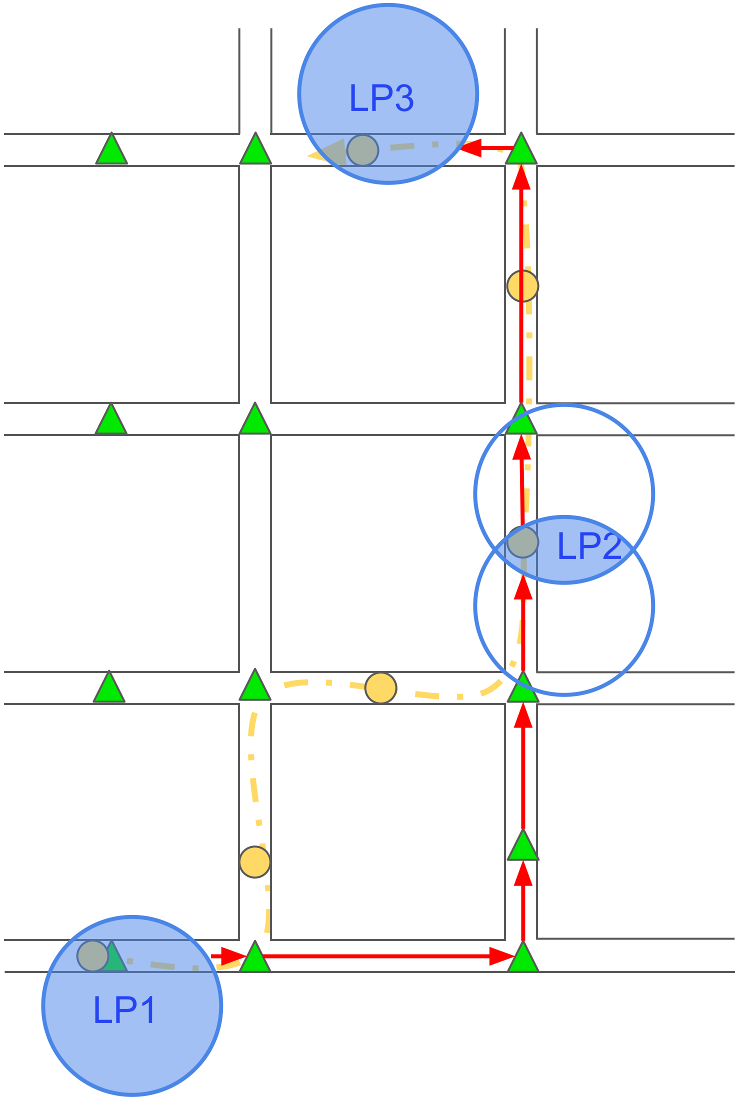
Street-map networks:

- Nodes (at least) at street intersections
- Additional information for each street (e.g., if one-way only, maximum speed)

SP uses street maps to compute the distance between LPs.

Add nodes every 0.5 seconds.

Add nodes at intersections of LPs and streets.



Achieving tighter bounds (3)

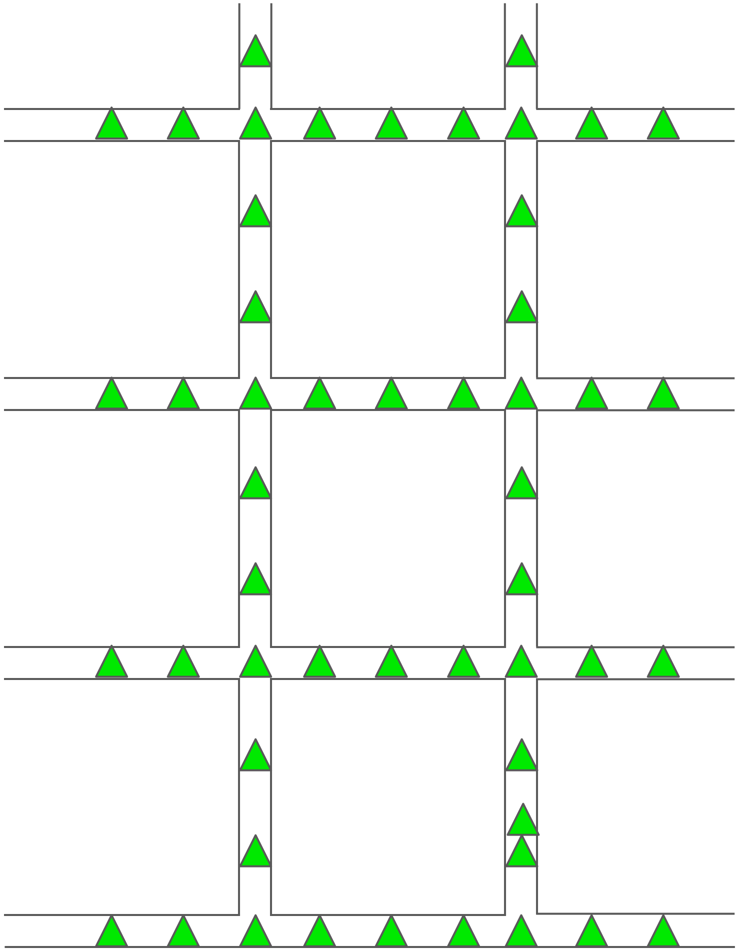
Street-map networks:

- Nodes (at least) at street intersections
- Additional information for each street (e.g., if one-way only, maximum speed)

SP uses street maps to compute the distance between LPs.

Add nodes every 0.5 seconds.

Add nodes at intersections of LPs and streets.



Achieving tighter bounds (4)

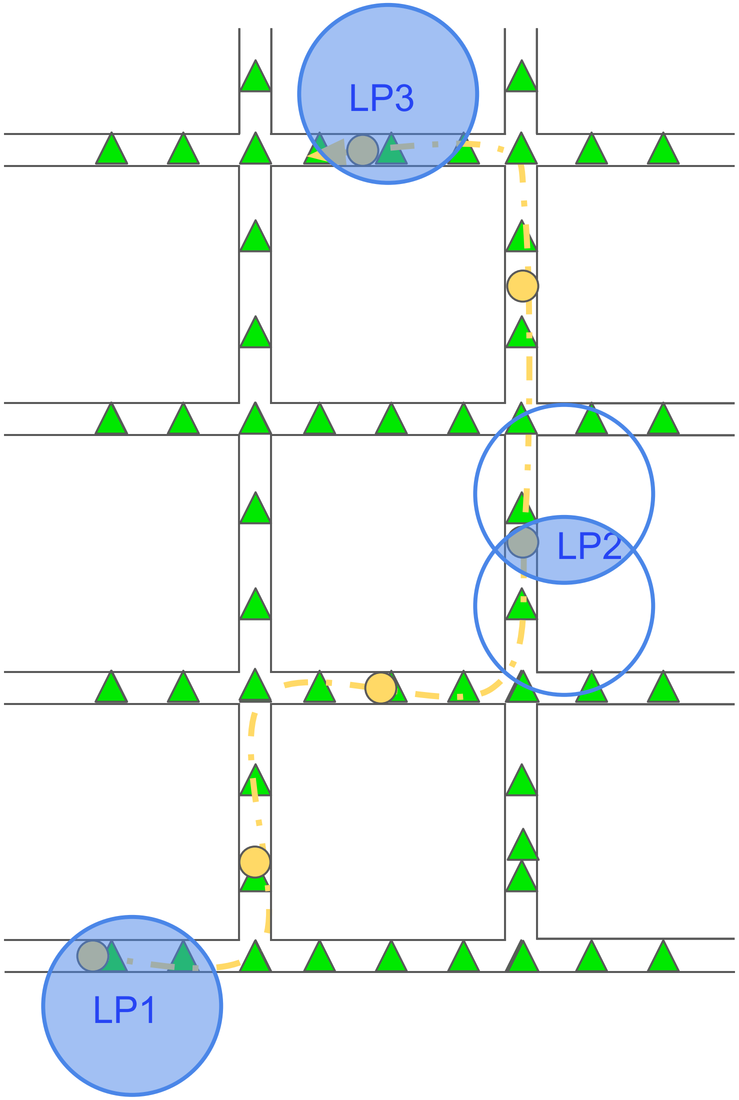
Street-map networks:

- Nodes (at least) at street intersections
- Additional information for each street (e.g., if one-way only, maximum speed)

SP uses street maps to compute the distance between LPs.

Add nodes every 0.5 seconds.

Add nodes at intersections of LPs and streets.



Achieving tighter bounds (5)

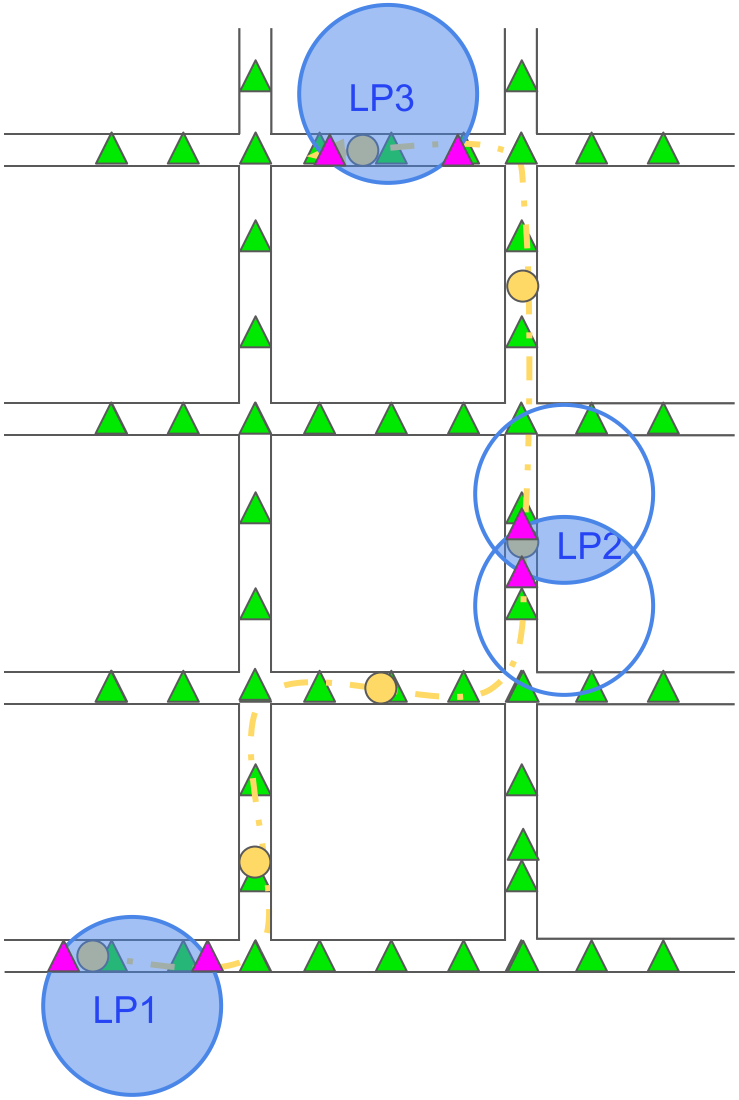
Street-map networks:

- Nodes (at least) at street intersections
- Additional information for each street (e.g., if one-way only, maximum speed)

SP uses street maps to compute the distance between LPs.

Add nodes every 0.5 seconds.

Add nodes at intersections of LPs and streets.



Lower bound computation (1)

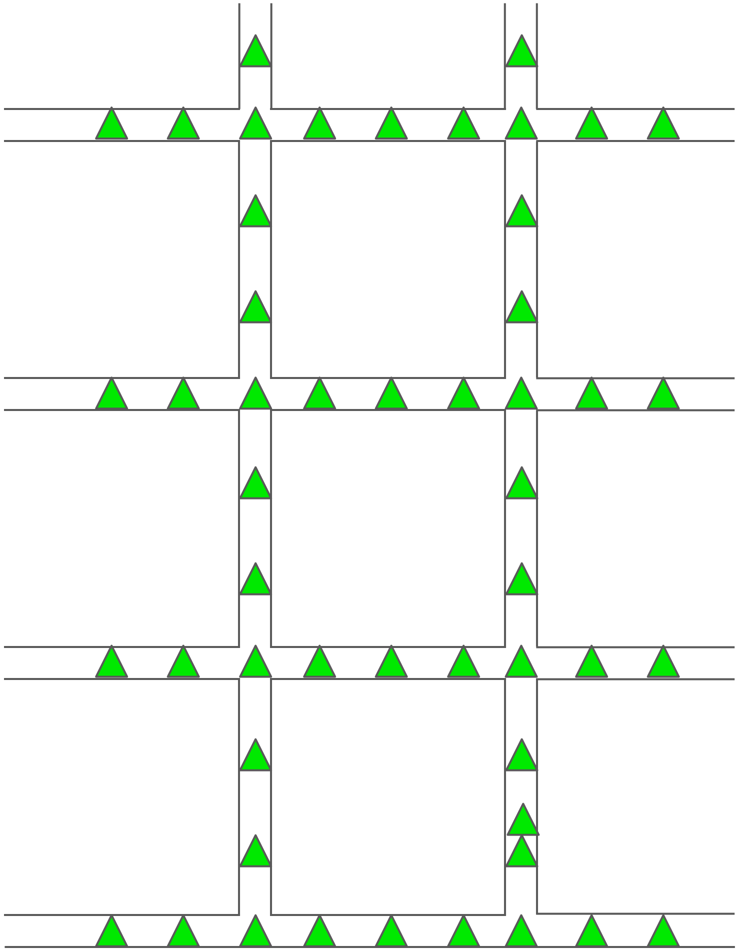
SP builds a basic graph from the dense street-map network.

Given GPS updates and LPs of a trace, SP defines LP areas.

SP adds nodes to the graph corresponding to intersections between streets and LP areas.

SP defines LP nodes as the nodes inside each LP area.

SP computes the lower bound of the distance by summing up the shortest paths between any two consecutive sets of LP nodes.



Lower bound computation (2)

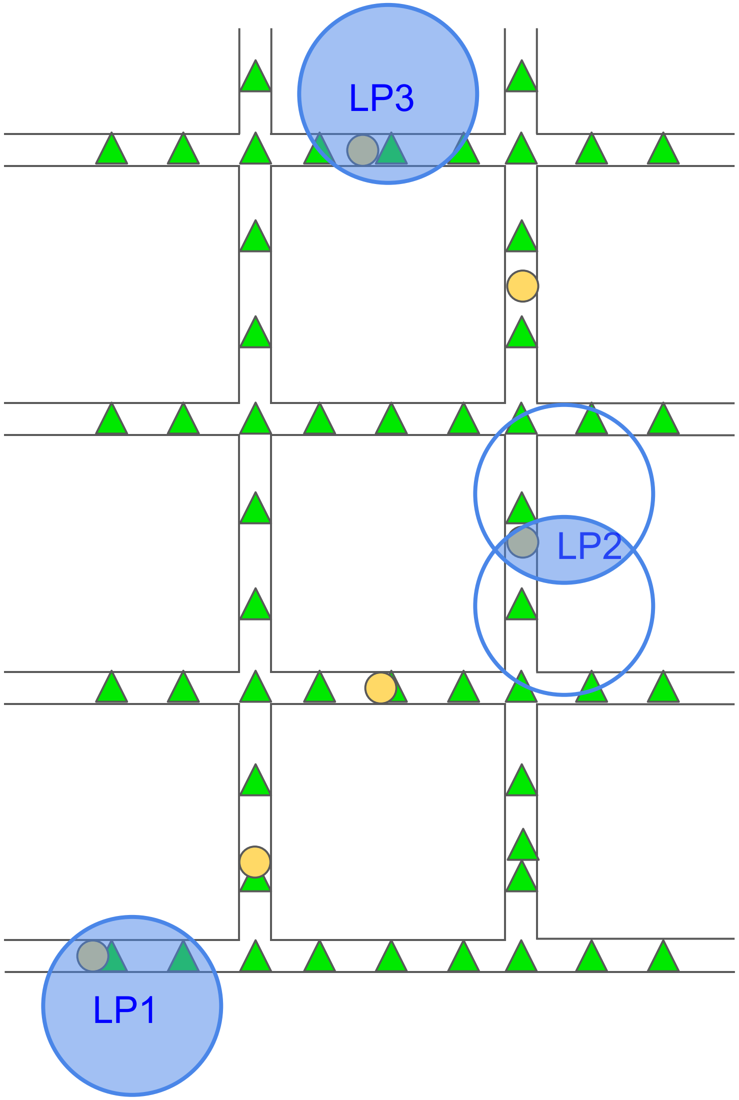
SP builds a basic graph from the dense street-map network.

Given GPS updates and LPs of a trace, SP defines LP areas.

SP adds nodes to the graph corresponding to intersections between streets and LP areas.

SP defines LP nodes as the nodes inside each LP area.

SP computes the lower bound of the distance by summing up the shortest paths between any two consecutive sets of LP nodes.



Lower bound computation (3)

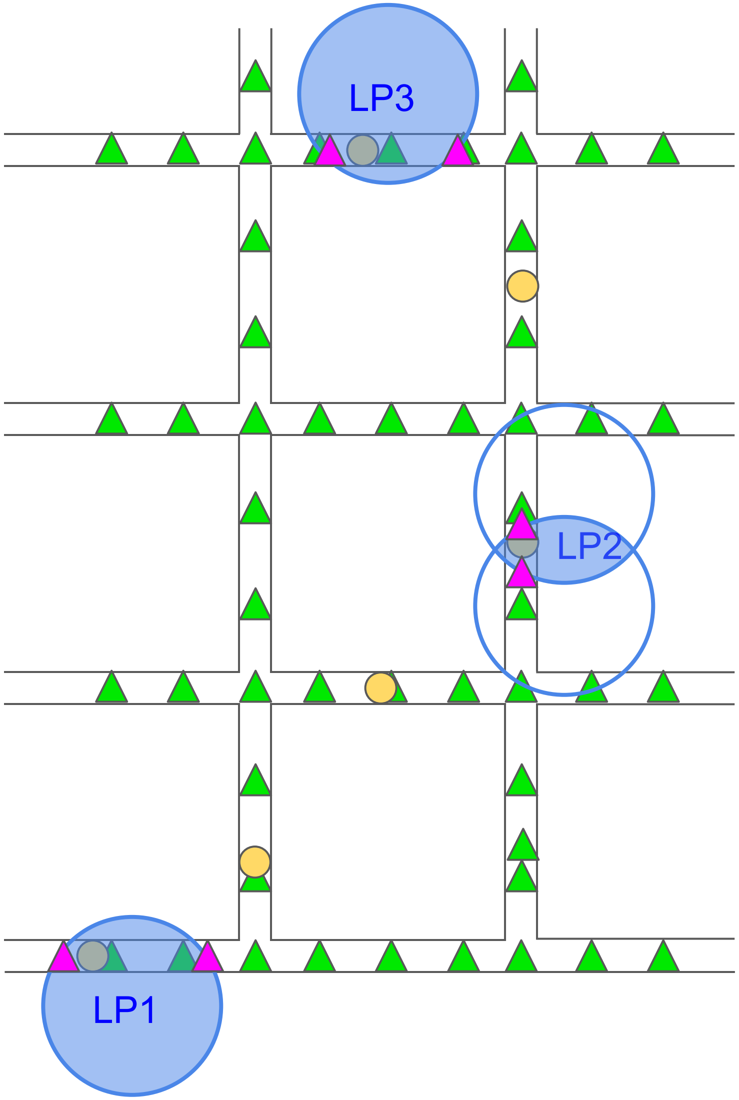
SP builds a basic graph from the dense street-map network.

Given GPS updates and LPs of a trace, SP defines LP areas.

SP adds nodes to the graph corresponding to intersections between streets and LP areas.

SP defines LP nodes as the nodes inside each LP area.

SP computes the lower bound of the distance by summing up the shortest paths between any two consecutive sets of LP nodes.



Lower bound computation (4)

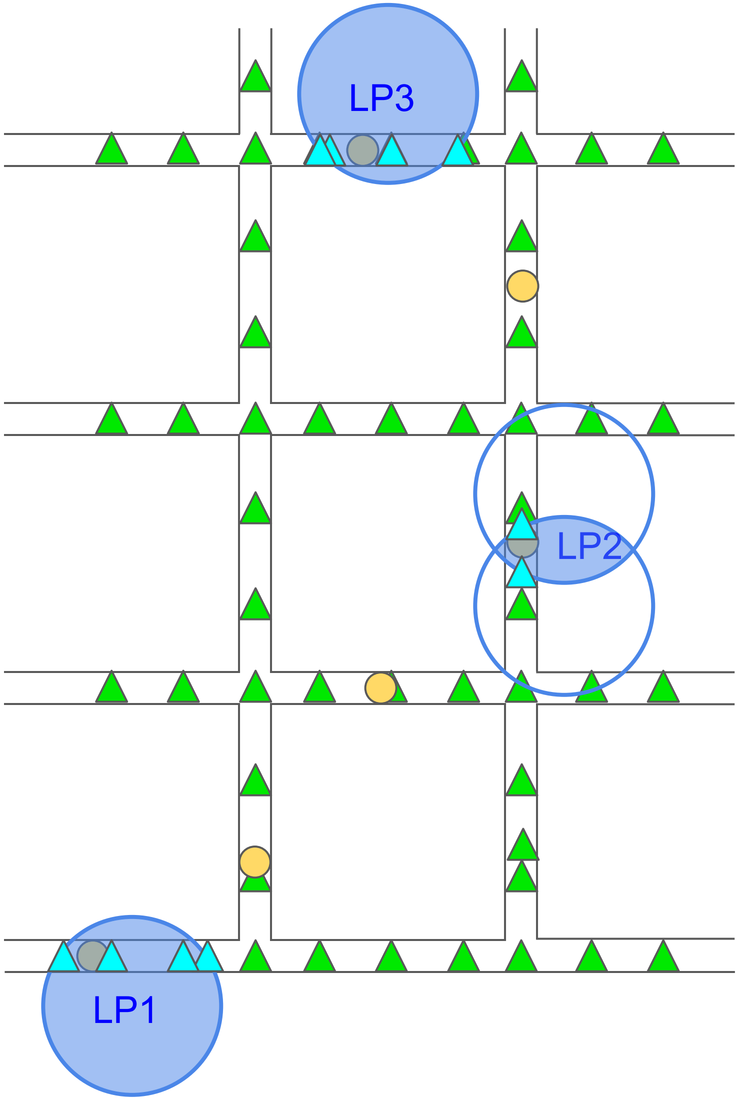
SP builds a basic graph from the dense street-map network.

Given GPS updates and LPs of a trace, SP defines LP areas.

SP adds nodes to the graph corresponding to intersections between streets and LP areas.

SP defines LP nodes as the nodes inside each LP area.

SP computes the lower bound of the distance by summing up the shortest paths between any two consecutive sets of LP nodes.



Lower bound computation (5)

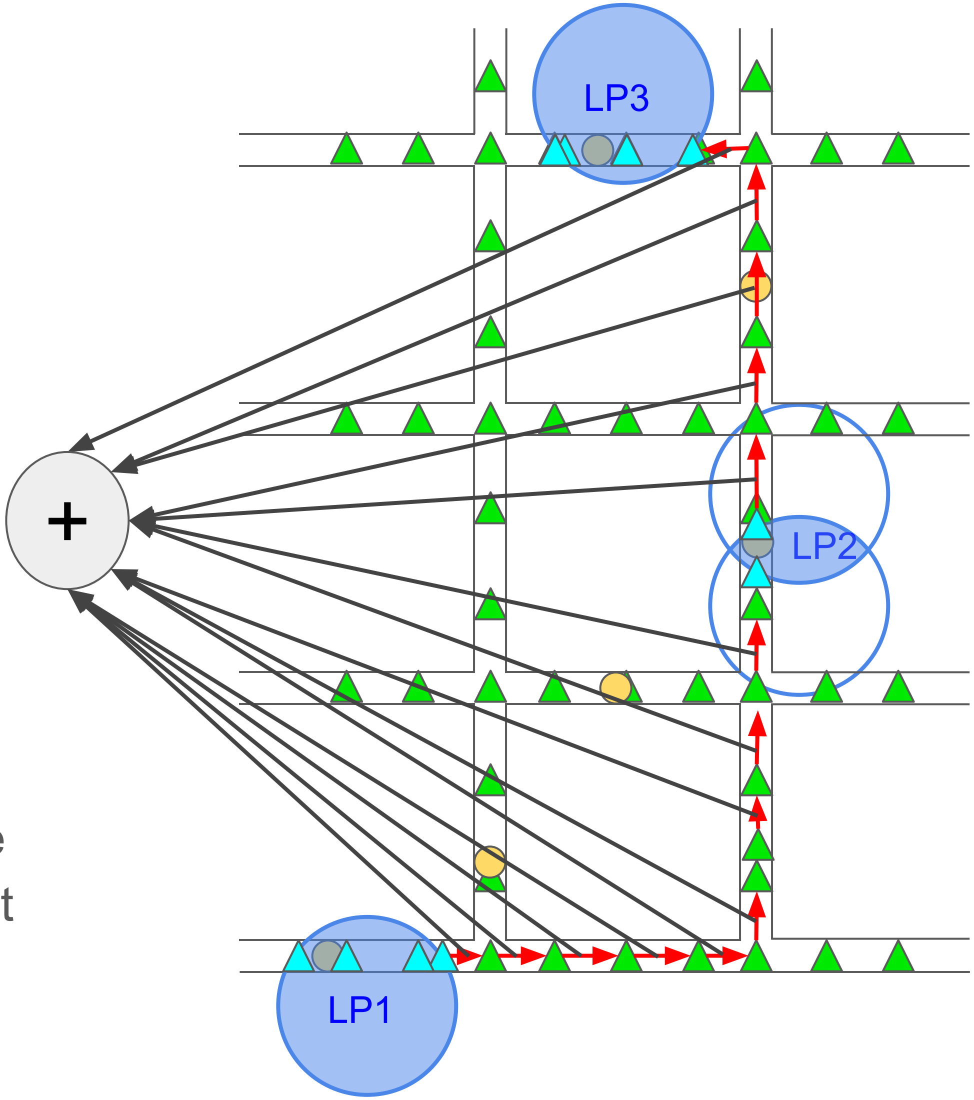
SP builds a basic graph from the dense street-map network.

Given GPS updates and LPs of a trace, SP defines LP areas.

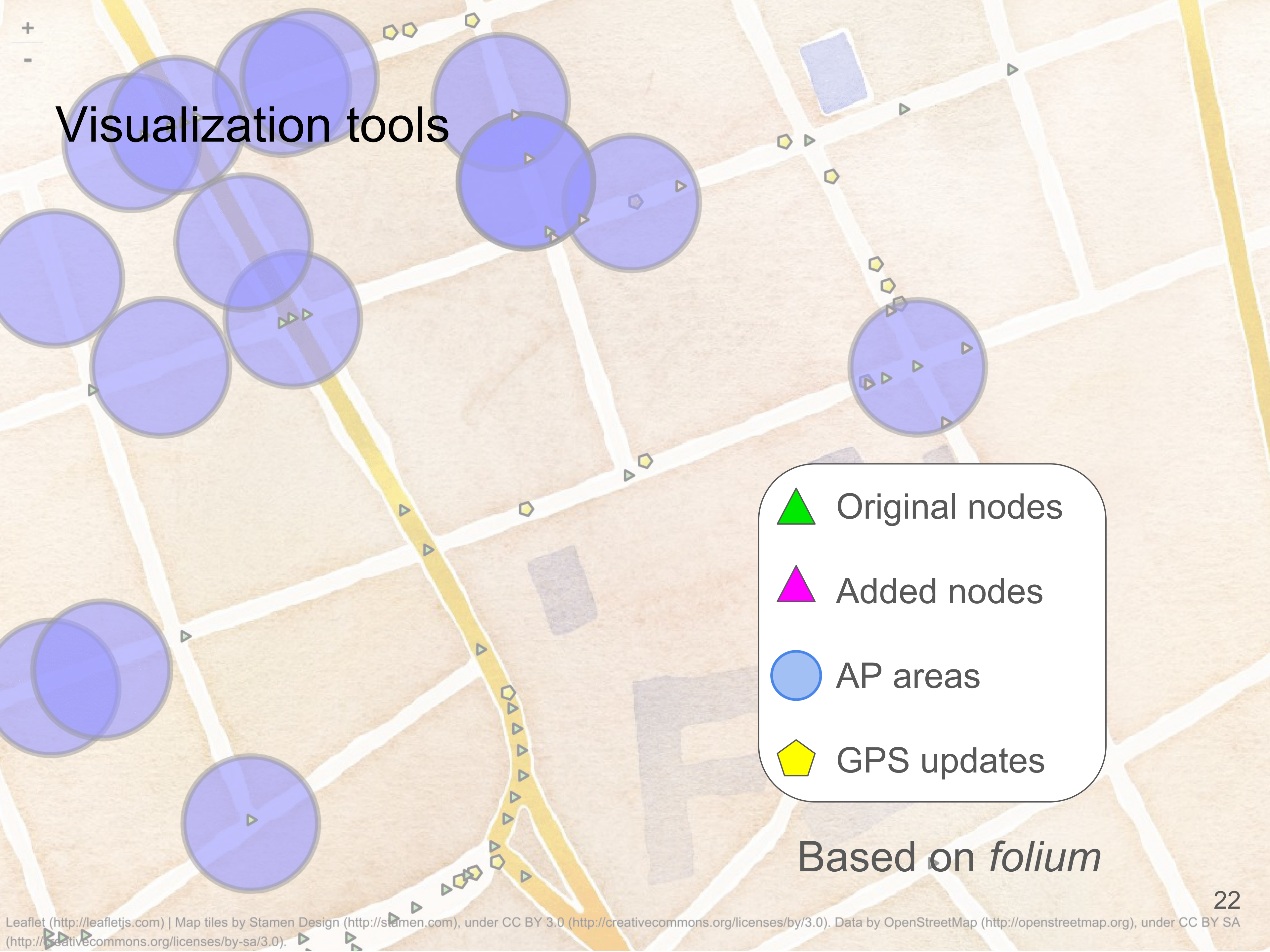
SP adds nodes to the graph corresponding to intersections between streets and LP areas.

SP defines LP nodes as the nodes inside each LP area.

SP computes the lower bound of the distance by summing up the shortest paths between any two consecutive sets of LP nodes.



Visualization tools



Leaflet (http://leafletjs.com) | Map tiles by Stamen Design (http://stamen.com), under CC BY 3.0 (http://creativecommons.org/licenses/by/3.0). Data by OpenStreetMap (http://openstreetmap.org), under CC BY SA (http://creativecommons.org/licenses/by-sa/3.0).

Based on *folium*

Datasets

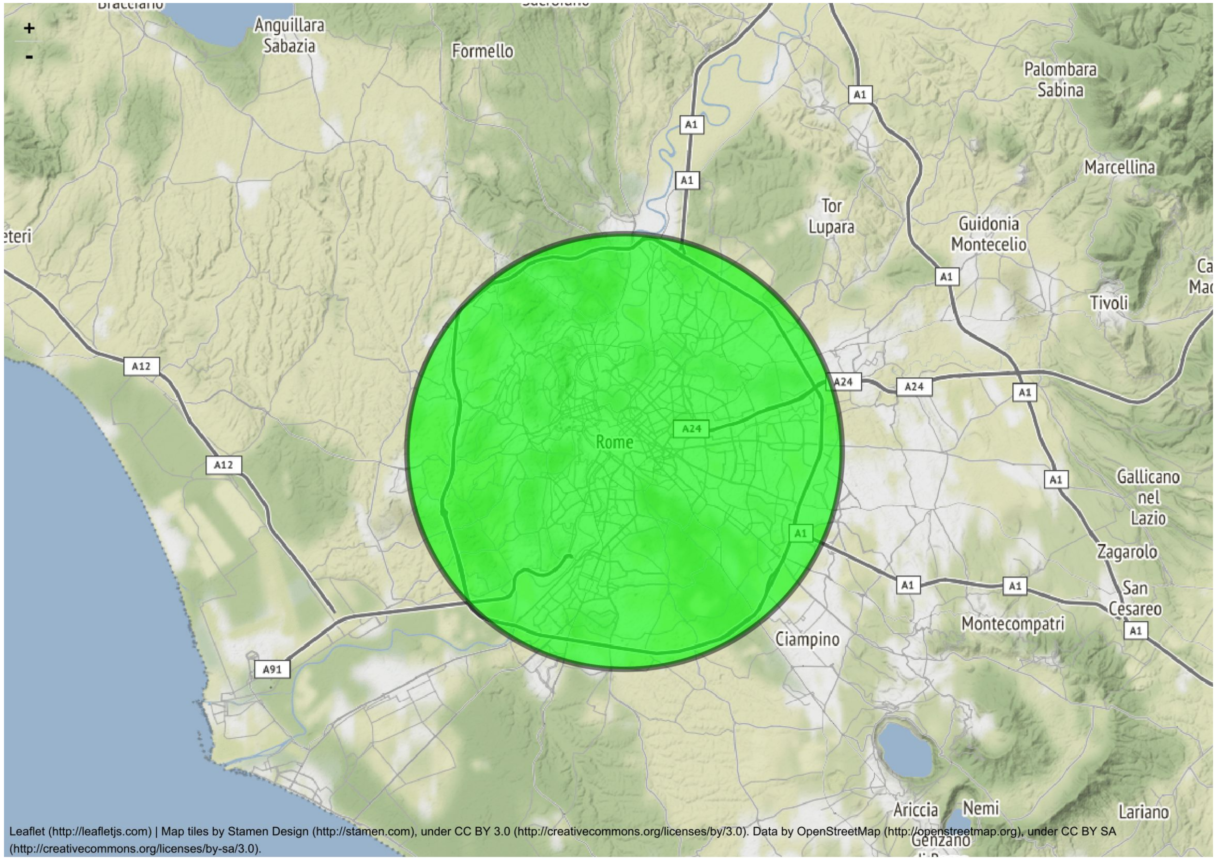
- Mobility traces of 316 taxi cabs in Rome, Italy: GPS coordinates collected in February 2014[1]
- 9,883,501 FON Wi-Fi APs[2]
- OSM data[2]:
  - 150,493 roads
  - 1,400,348 nodes

[1] Available at: crawdad.org/roma/taxi/20140717/

[2] Collected by L. Magarshack

Filtered datasets (1)

We consider the following region as Rome:



Leaflet (http://leafletjs.com) | Map tiles by Stamen Design (http://stamen.com), under CC BY 3.0 (http://creativecommons.org/licenses/by/3.0). Data by OpenStreetMap (http://openstreetmap.org), under CC BY SA (http://creativecommons.org/licenses/by-sa/3.0).

Filtered datasets (2)

Given this region, we filter out data not lying inside it.

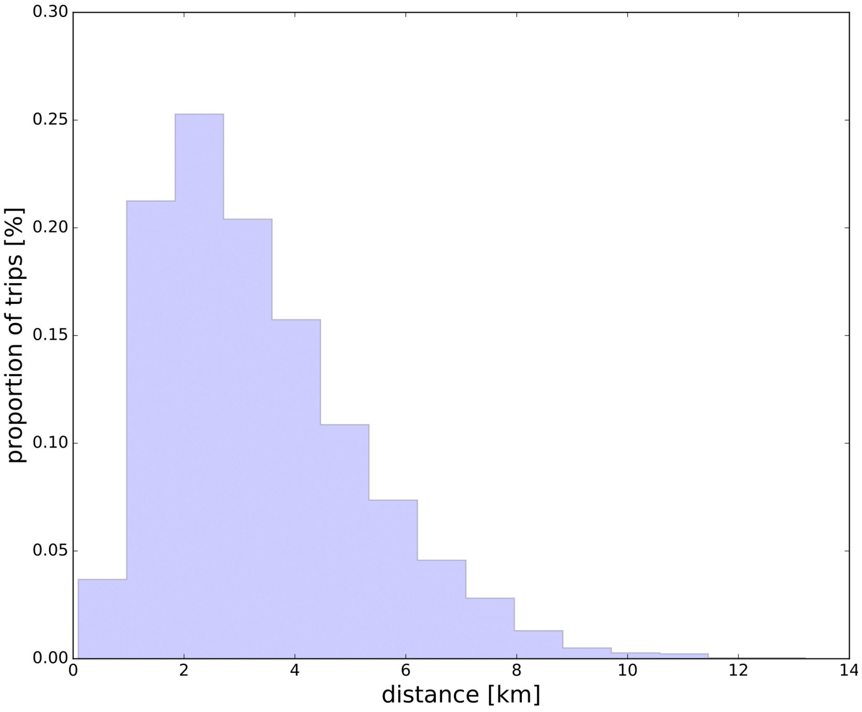
The filtered datasets consist of:

- 44,243 FON Wi-Fi APs
- 811,306 OSM nodes
  793, 433 in the giant component (disconnected graph)

Filtered datasets (3)

We split GPS updates according to the following distribution, derived from the distribution of average trip distances by yellow taxis in NYC in 2014.

We obtain 176,603 traces.

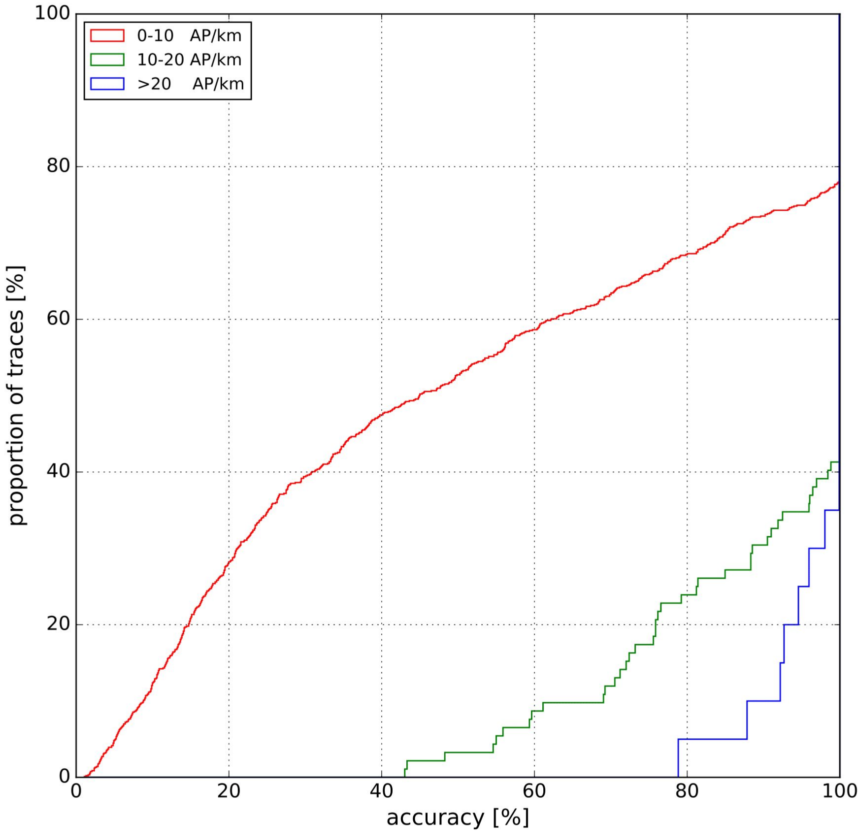


Methodology

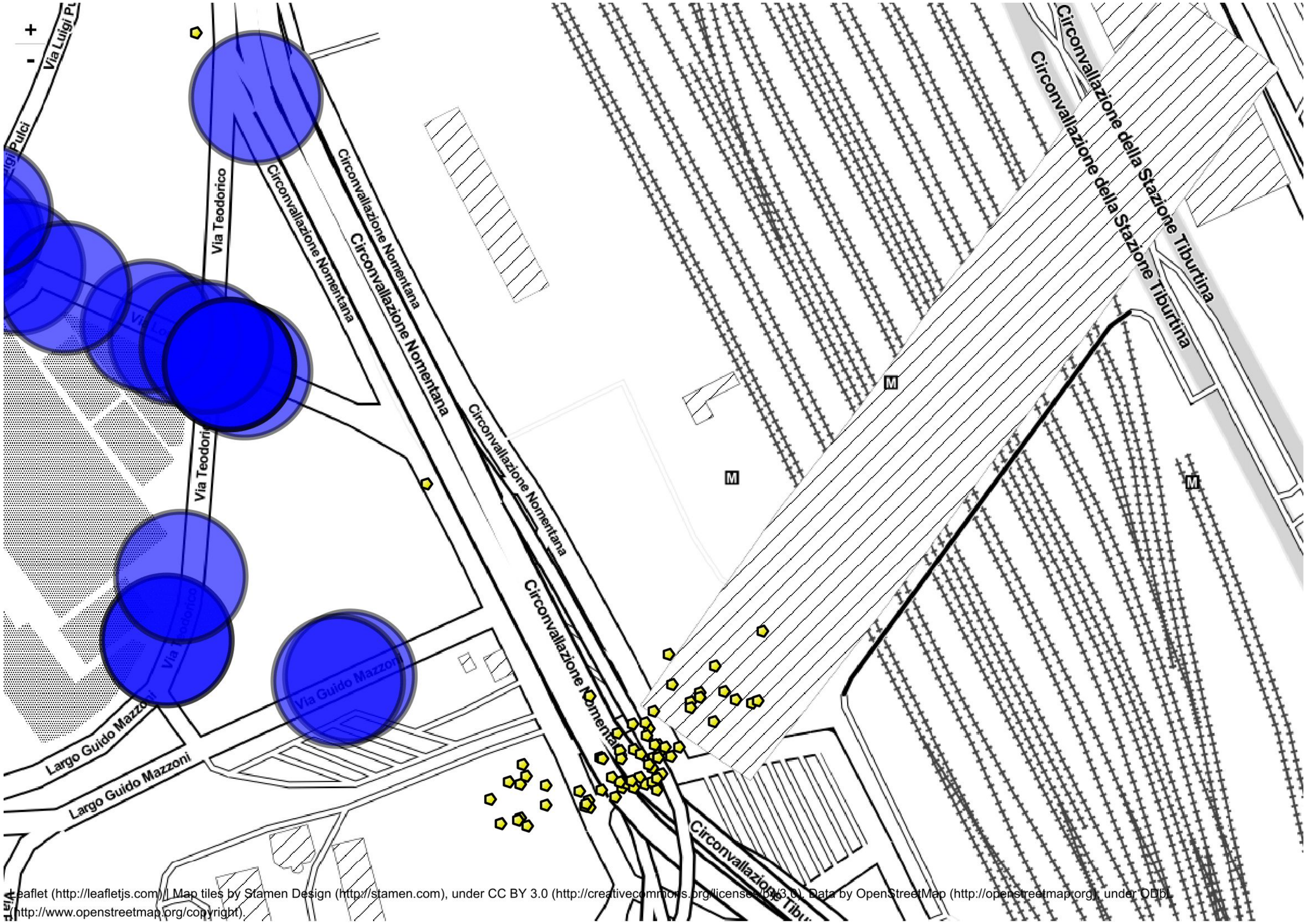
- 20 Jupyter notebooks (code in Python)
- 10 Python scripts
- 2 libraries in Python [1700 LOC]
- Easily extendible for the upper bound computation
- Most challenging part:
  adding new nodes at LP areas’ intersections

Results

- 1,000 random traces
- 55.21% median accuracy
- less than 10 APs/km in most of the traces
- 9 min/trace
    - CPU: Intel Xeon X5650 @ 2.67GHz
    - Memory: 88 GB



A bad case


+
-
Via Luigi Pulci
Via Teodorico
Via Teodorico
Via Teodorico
Circonvallazione Nomentana
Circonvallazione Nomentana
Circonvallazione Nomentana
Circonvallazione Nomentana
Circonvallazione Nomentana
Circonvallazione Tiburtina
Circonvallazione della Stazione Tiburtina
Circonvallazione della Stazione Tiburtina
M
M
M
Via Guido Mazzoni
Largo Guido Mazzoni
Largo Guido Mazzoni

Leaflet (http://leafletjs.com) | Map tiles by Stamen Design (http://stamen.com), under CC BY 3.0 (http://creativecommons.org/licenses/by/3.0). Data by OpenStreetMap (http://openstreetmap.org), under ODbL (http://www.openstreetmap.org/copyright).

Conclusion & future work

Conclusion:

- Infrastructure-based solution
- Visualization tools
- Easily extendible for the upper bound computation

Future steps:

- Maximize lower bound by computing the longest path in a DAG
- Evaluate our solution on a dataset of traces corresponding to real rides.
- Compute the upper bound of the traveled distance

Conclusion & future work

Thank you

Conclusion:

- Infrastructure-based solution
- Visualization tools
- Easily extendible for the upper bound computation

Future steps:

- Maximize lower bound by computing the longest path in a DAG
- Evaluate our solution on a dataset of traces corresponding to real rides.
- Compute the upper bound of the traveled distance