Department of Computer Science
University of Pisa

Locality Filtering for efficient ride sharing platforms

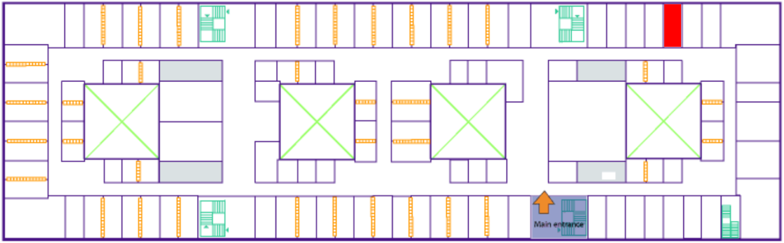
Mauriana Pesaresi Series, Seminar #6

Francesco Tosoni, M.Sc.

About me

I am a first year Ph.D. student in CS (room 300), and a member of the *Acube* Laboratory directed by Professor P. Ferragina.

- Algorithms for real world applications
- Information Retrieval
- Data compression



Outline

What's ride sharing?

Ride sharing is a long-standing proposition for decreasing road traffic.
Many existing variant:

- *carpooling* ⇒reduce the number of cars used by co-workers
- *share-a-ride* (SARP) ⇒ combine human and freight transportation
- shared one-off journeys ... and many others!

Ride sourcing

With regard to shared taxis many **ride sourcing** services are available worldwide (e.g. Uber, Didi, Lyft, Via). Customers performs *e-hailing* requests interacting with a (customer) smartphone app; driver communicate their availability through a different (driver) smartphone app.

Uber DiDi lyft VIA

... soon transition to self-driving vehicles

Ride sourcing

With regard to shared taxis many **ride sourcing** services are available worldwide (e.g. Uber, Didi, Lyft, Via). Customers performs *e-hailing* requests interacting with a (customer) smartphone app; driver communicate their availability through a different (driver) smartphone app.

Uber DiDi lyft VIA

... soon transition to self-driving vehicles $\Rightarrow$ increasing need to develop new operations research models that efficiently combine users and vehicles.

What about scalability? 🤔

MIT SENSEABLE CITY LAB pioneered investigations on urban mobility scenarios.

- In Manhattan (NYC) most of taxi requests can be combined [Santi *et al.*, 2014]
- Recent studies confirmed the benefits of ride sharing in terms of: money savings, traffic reduction, ...
- There is an emerging need for efficient and scalable algorithms to deal with:
  1. geographical road networks 🌍
  2. online scenarios 🕰️

Outline

The ride sharing problem

Definition (The parameters)

1. A city graph $G_A = (V_A, E_A)$ representing a geographical area with crossroads (nodes) and road segments (edges)
2. A set $\mathcal{T}$ of taxi requests T_i for which we consider: starting time st_i, expected arrival time at_i, origin o_i and destination d_i

The ride sharing problem

Definition (The parameters)

1. A city graph $G_A = (V_A, E_A)$ representing a geographical area with crossroads (nodes) and road segments (edges)
2. A set $\mathcal{T}$ of taxi requests T_i for which we consider: starting time st_i, expected arrival time at_i, origin o_i and destination d_i
3. A *shareability parameter* k, limiting the amount of trips that can be “combined" together
4. A *quality of service parameter* Δ, limiting the delay that passengers can face due to ride sharing.
5. A *time window parameter* δ represents an upper bound for the response time of the taxi management system (wait for a batch of requests).

The ride sharing problem (ctd.)

Definition (Ride sharing)

Given the tuple $(G_A, \mathcal{T}, k, \Delta, \delta)$, the **ride sharing problem** consists of finding a feasible matching $\mathcal{M}$ of the trips in $\mathcal{T}$ such that constraints about k, δ and Δ are respected.

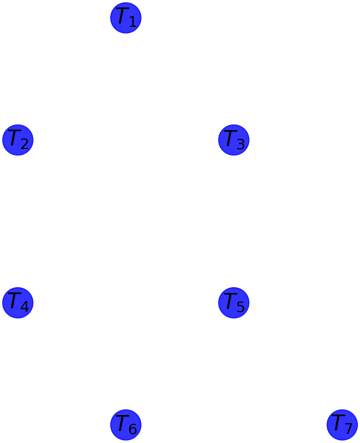
Definition (Shareability Network)

The **shareability network** G_{SN} [P. Santi *et al.*, 2014] is an unweighted non-directed graph, where:

- vertices V_{SN} represent the trips in $\mathcal{T}$
- edges E_{SN} represent opportunities for a match

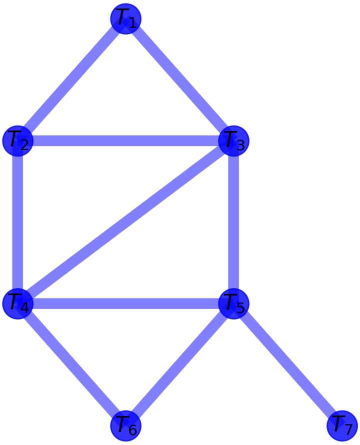
The shareability network 🤝

1. Build an edge-empty G_{SN}



The shareability network 🤝

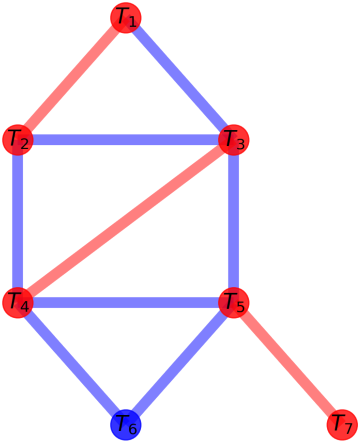
1. Build an edge-empty G_{SN}
2. Create an edge for each feasible match



Many shortest path calculations involved at this step!

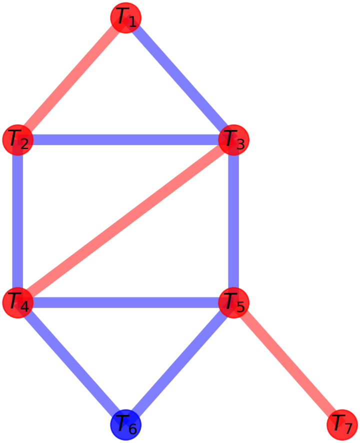
The shareability network 🤝

1. Build an edge-empty G_{SN}
2. Create an edge for each feasible match
3. Execute a matching algorithm on top of the G_{SN}



The shareability network 🤝

1. Build an edge-empty G_{SN}
2. Create an edge for each feasible match
3. Execute a matching algorithm on top of the G_{SN}



Combined trips: (T_1, T_2), (T_3, T_4), (T_5, T_7).

Need for efficient solutions 💡

- The original paper [Santi *et al.*, 2014] shows that is possible to combine **92%** of trips (δ =2 minutes, $\Delta = 5$ minutes, $k = 2$)
- The time to solve the matching problem is negligible (<0.1 secs) over a commodity workstation
- But...

Need for efficient solutions

- The original paper [Santi *et al.*, 2014] shows that is possible to combine **92%** of trips (δ =2 minutes, Δ = 5 minutes, k = 2)
- The time to solve the matching problem is negligible (<0.1 secs) over a commodity workstation
- But...

Populating the G_{SN} is expensive!

1. Many shortest path computations needed for all-to-all trip comparisons (~10^7 in Manhattan)

Need for efficient solutions 💡

- The original paper [Santi *et al.*, 2014] shows that is possible to combine **92%** of trips (δ =2 minutes, $\Delta = 5$ minutes, $k = 2$)
- The time to solve the matching problem is negligible (<0.1 secs) over a commodity workstation
- But...

Populating the G_{SN} is expensive!

1. Many shortest path computations needed for all-to-all trip comparisons (~10^7 in Manhattan)
2. Furthermore users are *geographically dispersed* $\Rightarrow$ for an all-to-all comparison we need to possibly explore the whole city graph when using Dijkstra-like solutions

What’s wrong? ❌

Two naïve approaches available to compute shortest paths:

What's wrong? ❌

Two naïve approaches available to compute shortest paths:

1. Run $\Theta(n^2)$ Dijkstra computations, each one costing $\Theta(n \log n)$

What's wrong? ❌

Two naïve approaches available to compute shortest paths:

1. Run $\Theta(n^2)$ Dijkstra computations, each one costing $\Theta(n \log n)$
   $\Rightarrow$ Too much time!

What's wrong? ❌

Two naïve approaches available to compute shortest paths:

1. Run $\Theta(n^2)$ Dijkstra computations, each one costing $\Theta(n \log n)$
   $\Rightarrow$ Too much time!
2. Precompute a $\Theta(n^2)$ all-pair-shortest-path matrix (e.g. using Floyd-Warshall algorithm, repeated Dijkstra executions)

What's wrong? ❌

Two naïve approaches available to compute shortest paths:

1. Run $\Theta(n^2)$ Dijkstra computations, each one costing $\Theta(n \log n)$
   $\Rightarrow$ Too much time!
2. Precompute a $\Theta(n^2)$ all-pair-shortest-path matrix (e.g. using Floyd-Warshall algorithm, repeated Dijkstra executions)
   $\Rightarrow$ Too much space!

What's wrong? ❌

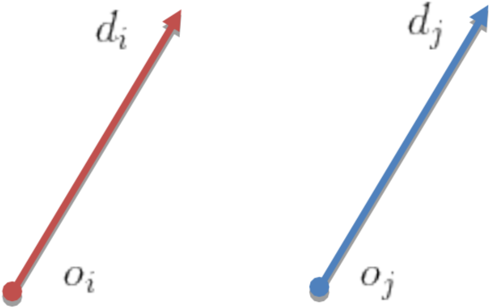
Two naïve approaches available to compute shortest paths:

1. Run $\Theta(n^2)$ Dijkstra computations, each one costing $\Theta(n \log n)$
   $\Rightarrow$ Too much time!
2. Precompute a $\Theta(n^2)$ all-pair-shortest-path matrix (e.g. using Floyd-Warshall algorithm, repeated Dijkstra executions)
   $\Rightarrow$ Too much space!

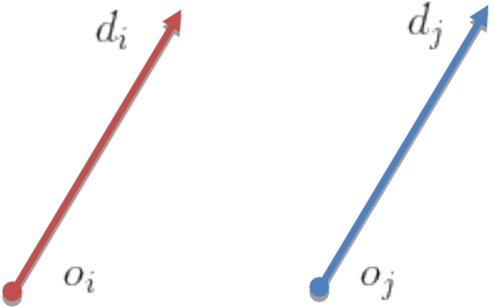
No way!

In order to speed up the solution to the ride sharing problem we need to reduce the number of shortest path queries.

These are two taxi rides!

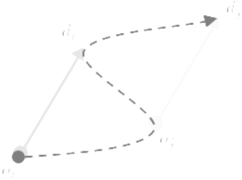


These are two taxi rides!

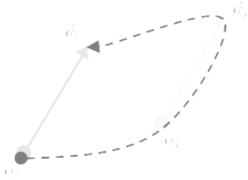


Let 's see how can we (possibly) match them together to form a combined trip.

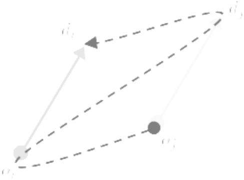
Possible trip combinations



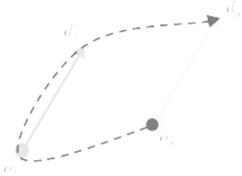
(a) Case 1.



(b) Case 2.



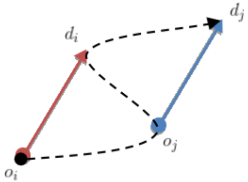
(c) Case 3.



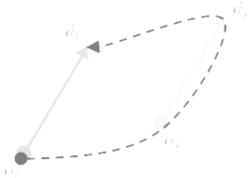
(d) Case 4.

It is possible to combine rides according to *four* different patterns.

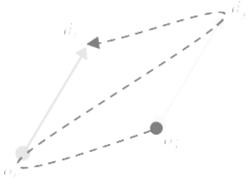
Possible trip combinations



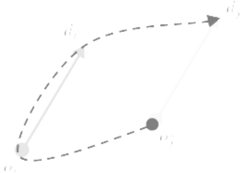
(a) Case 1. $A(i, j)$



(b) Case 2.



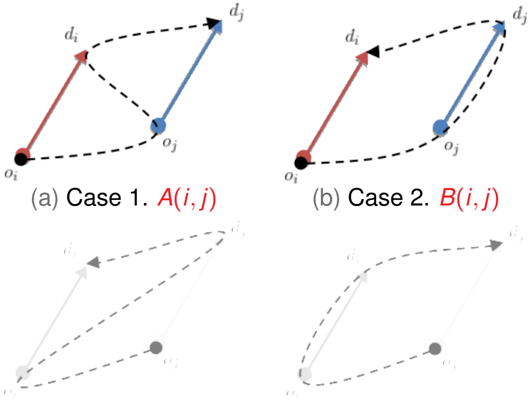
(c) Case 3.



(d) Case 4.

We define few conditions – called $A(i, j)$ – for a *match of the first kind*, which take into account time constraints (on *st* and *at*) and traversal times of the city between locations *o*/*d*.

Possible trip combinations



(a) Case 1. $A(i,j)$

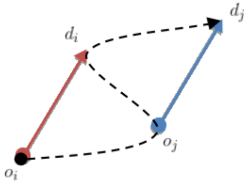
(b) Case 2. $B(i,j)$

(c) Case 3.

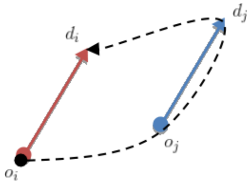
(d) Case 4.

We define few conditions – called $B(i,j)$ – for a *match of the second kind*, which take into account time constraints (on *st* and *at*) and traversal times of the city between locations *o*/*d*.

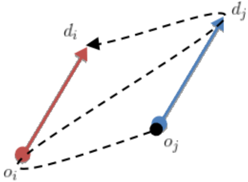
Possible trip combinations



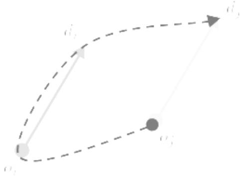
(a) Case 1. $A(i,j)$



(b) Case 2. $B(i,j)$



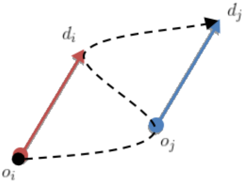
(c) Case 3. $A(j,i)$



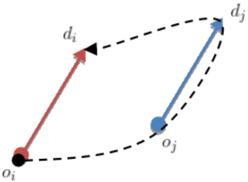
(d) Case 4.

Case 3 is dual of case 1.

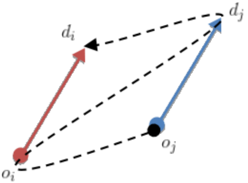
Possible trip combinations



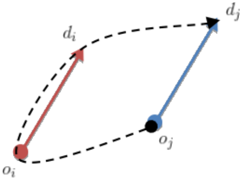
(a) Case 1. $A(i, j)$



(b) Case 2. $B(i, j)$



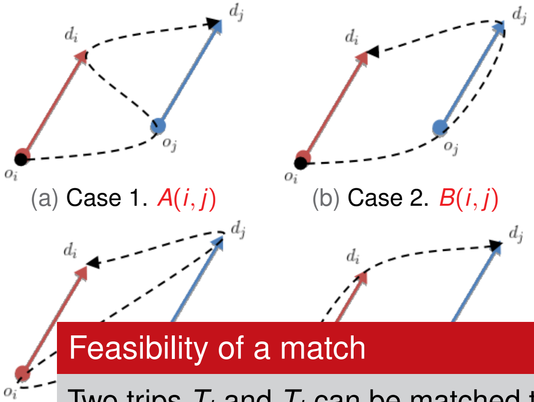
(c) Case 3. $A(j, i)$



(d) Case 4. $B(j, i)$

Case 4 is dual of case 2.

Possible trip combinations



(a) Case 1. $A(i, j)$ (b) Case 2. $B(i, j)$

(c)

Feasibility of a match

Two trips T_i and T_j can be matched together *iff*

- $|st_i - st_j| \leq \delta$
- $A(i, j) \vee B(i, j) \vee A(j, i) \vee B(j, i)$

Outline

Spatiotemporal correlation

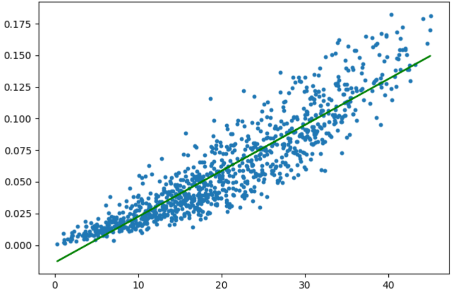
Claim

It does exist a strong correlation between:

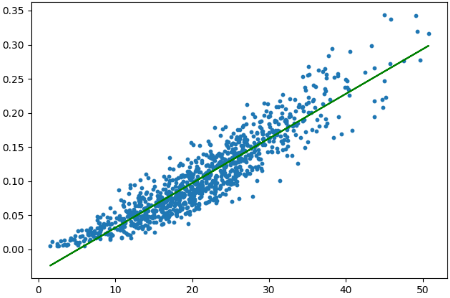
- the **traversal time** of a source-dest path (shortly, s-d path)
- the s-d **euclidean distance**

- from every single source consider s-d paths which can be traversed in less than Δ_S time (e.g. 5 minutes)
- compute s-d euclidean distances for those paths
- estimate distribution of these distances in order to derive a robust mapping between time-space

Spatiotemporal correlation (ctd.)



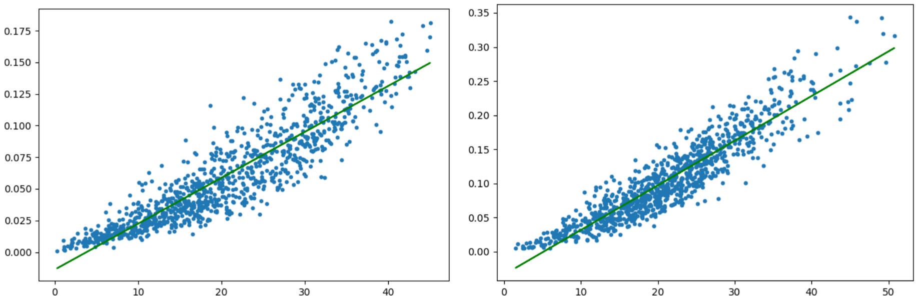
(a) Manhattan (NYC)



(b) Singapore

Correlation between traversal times (x-axis, minutes) and euclidean distances (y-axis). The maximum displayed time value (x-axis) corresponds to $\Delta_S = 5$ minutes.

Spatiotemporal correlation (ctd.)



(a) Manhattan (NYC) (b) Singapore

90% of paths in Manhattan – FIGURE (A) – cover a distance which is $\leq$ 140.04 in less than 5 minutes.

Spatiotemporal correlation (ctd.)

- We repeated the experiment multiple times in order to consider different values for Δ_S
- We associate to Δ_S the **distance corresponding to the 90-percentile** of the empirical euclidean distance distribution. While doing this, we are discarding just 10% of valid paths.

Spatiotemporal correlation (ctd.)

- We repeated the experiment multiple times in order to consider different values for Δ_S
- We associate to Δ_S the **distance corresponding to the 90-percentile** of the empirical euclidean distance distribution. While doing this, we are discarding just 10% of valid paths.

Locality filtering

We designed a locality filter able to translate ***time-based*** conditions for a match of trips into (almost equivalent) ***geographical-based*** conditions.

Geographical relationships

The distance values in which we are interested are:

- the distance l_i to the time $tt(o_i, d_i)$ (i.e. time needed to serve T_i in isolation)
- the distance D_i to the time Δ

Geographical relationships

The distance values in which we are interested are:

- the distance l_i to the time $tt(o_i, d_i)$ (i.e. time needed to serve T_i in isolation)
- the distance D_i to the time Δ

Time-to-distance specialyzation

These time-to-distance association have been specialyzed for each different district within the city graph. We are hence able to capture the complexity of the speed networks of cities from the old world (i.e. Pisa)

For this reason we have in general that $l_i \neq l_j$, as well as $D_i \neq D_j$.

Distance relationships

Lemma (Match of the first kind)

We translate $A(i,j)$ for a FM of the first kind into the geographic-based set of conditions:

$$\begin{cases} d(o_i, o_j) + d(o_j, d_i) \leq l_i + D_i \\ d(o_j, d_i) + d(d_i, d_j) \leq l_j + D_j \end{cases}$$

Distance relationships

Lemma (Match of the first kind)

We translate $A(i, j)$ for a FM of the first kind into the geographic-based set of conditions:

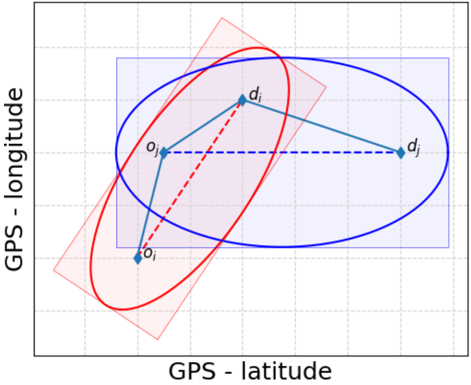
$$\begin{cases} d(o_i, o_j) + d(o_j, d_i) \leq l_i + D_i \\ d(o_j, d_i) + d(d_i, d_j) \leq l_j + D_j \end{cases}$$

...in other words:

Locality areas are defined by ellipses.

- Origin o_j belongs to locality area of T_i
- Destination d_i belongs to locality area of T_j

NEW Geometric based conditions



Match of the first kind

We translate conditions $A(i,j)$ for a feasible match of the *first kind* into the geographic-based set of conditions:

- Origin o_j belongs to locality area of T_i
- Destination d_i belongs to locality area of T_j

Locality areas are defined by ellipses. The dimensions of the ellipse axes are expressed as a function of l_i and D_i.

Distance relationships (ctd.)

Lemma (Match of the second kind)

We translate $B(i,j)$ for a FM of the first kind into the geographic-based set of conditions:

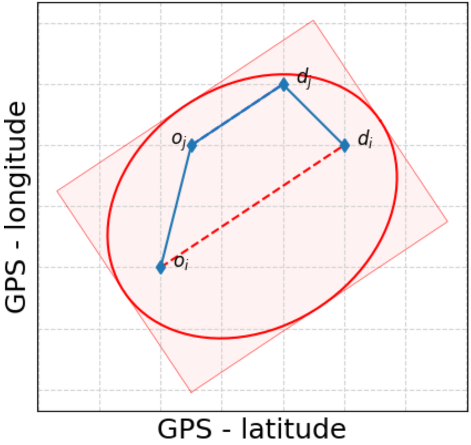
$$\begin{cases} d(o_i, o_j) + d(o_j, d_i) \leq l_i + D_i \\ d(o_i, d_j) + d(d_j, d_i) \leq l_i + D_i \end{cases}$$

In other words:

- Origin o_j belongs to locality area of T_i (as before)
- Destination d_j belongs to locality area of T_i (dual)

Again: locality areas are defined by ellipses.

NEW Geometric based conditions (ctd.)



Match of the second kind

We translate conditions $B(i, j)$ for a feasible match of the *second kind* into the geographic-based condition:
$\Rightarrow$ *Both* o_j and d_j belong to locality area of T_i

Again: locality areas are defined by ellipses. Again: we use l_i and D_i to dimension the ellipses.

Constructing the ellipses 🏗

- The involved ellipses must be properly dimensioned and placed in the cartesian plane.
- The direction of the trips is not – in general – parallel to the cartesian axis $\Rightarrow$ we need to rotate the ellipses

Constructing the ellipses

- The involved ellipses must be properly dimensioned and placed in the cartesian plane.
- The direction of the trips is not – in general – parallel to the cartesian axis ⇒ we need to rotate the ellipses

We used geometrical formulas based on the parameters l_i and D_i to compute the:

1. Greatest distance r_{max} and smallest distance r_{min} from the focus points
2. Dimensions of major and minor semi-axis
3. Positions of focus points (corresponding to trip origin and destination)
4. Angle of rotation w.r.t. x-axis

Outline

Range search

Range search

Given the previous discussion, we can find the ***candidate*** trips for feasible matches for conditions $A(i,j)$, $B(i,j)$, $A(j,i)$, $B(j,i)$ by executing (ellipse based) range searches over the cartesian plane.

- For this purpose we use the `Boost c++ Library` (very well maintained).
- We combine geographical filter with a cosine similarity filter (very simple and very effective)
- **Problem:** we can have false positives $\Rightarrow$ we need to execute a check on the candidate set

Details on our algorithm

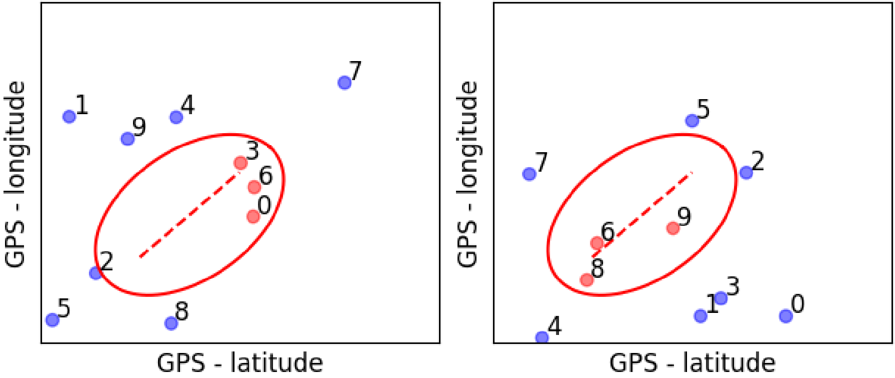
We construct two *postings lists* `OT[i]` and `DT[i]` for each trip T_i.

Definition (postings OT and DT)

- $OT[i]$ is the set of trips whose *origin* belongs to the ellipse of T_i
- $DT[i]$ is the set of trips whose *destination* belongs to the ellipse of T_i
- we also build the inverse DT^{-1} such that: $i \in DT^{-1}[j]$ if, and only if, $j \in DT[i]$

Candidates from set intersections

- First kind match candidates $= OT[i] \cap DT^{-1}[i]$
- Second kind match candidates $= OT[i] \cap DT[i]$



(a) Set of origins. (b) Set of destinations.

Trip T_6 appears on both sides $\Rightarrow$ it is a good candidate for a second kind match.

Outline

Experimental evaluation

We executed some tests using taxi hailings issued in Manhattan (NYC), and Singapore.

DATASET	TRIP REQUESTS	CROSSROADS (NODES)	ROADS (EDGES)	WD SAT/SUN
manhattan	581 631	4 091	9 452	✗
singapore	688 383	11 789	26 223	✓

Experimental evaluation (ctd.)

We compared our proposal and the naïve approach (i.e. all-to-all shortest-path comparisons) in terms of:

- Number of shortest paths queries issued to the system
- Number of feasible matches found
- Number of matched taxi requests
- Completion time

Experimental evaluation (ctd.)

We compared our proposal and the naïve approach (i.e. all-to-all shortest-path comparisons) in terms of:

- Number of shortest paths queries issued to the system $\Rightarrow$ from quadratic to near linear
- Number of feasible matches found
- Number of matched taxi requests
- Completion time

Experimental evaluation (ctd.)

We compared our proposal and the naïve approach (i.e. all-to-all shortest-path comparisons) in terms of:

- Number of shortest paths queries issued to the system $\Rightarrow$ from quadratic to near linear
- Number of feasible matches found $\Rightarrow$ sparsification of the shareability network
- Number of matched taxi requests
- Completion time

Experimental evaluation (ctd.)

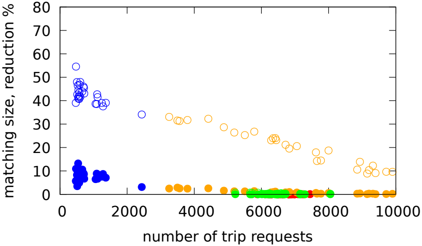
We compared our proposal and the naïve approach (i.e. all-to-all shortest-path comparisons) in terms of:

- Number of shortest paths queries issued to the system ⇒ from quadratic to near linear
- Number of feasible matches found ⇒ sparsification of the shareability network
- Number of matched taxi requests ⇒ same results as [P. Santi *et al.*, 2014]
- Completion time

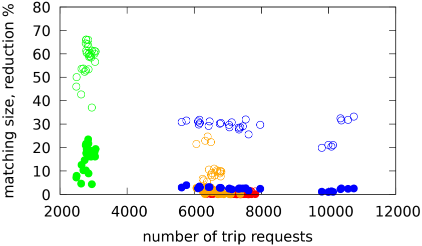
Experimental evaluation (ctd.)

We compared our proposal and the naïve approach (i.e. all-to-all shortest-path comparisons) in terms of:

- Number of shortest paths queries issued to the system ⇒ from quadratic to near linear
- Number of feasible matches found ⇒ sparsification of the shareability network
- Number of matched taxi requests ⇒ same results as [P. Santi *et al.*, 2014]
- Completion time ⇒ significant reduction, especially during the “rush time"

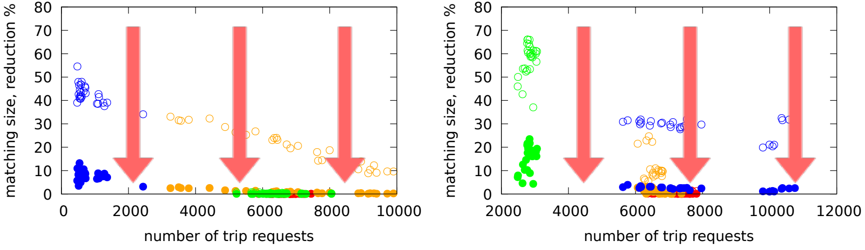


(a) Manhattan (NYC)



(b) Singapore

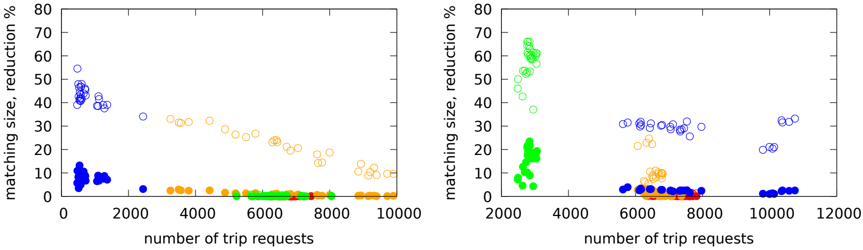
Figure: Number of issued trip requests (x-axis) VS matching size reduction (y-axis)



(a) Manhattan (NYC) (b) Singapore

Figure: Number of issued trip requests (x-axis) VS matching size reduction (y-axis)

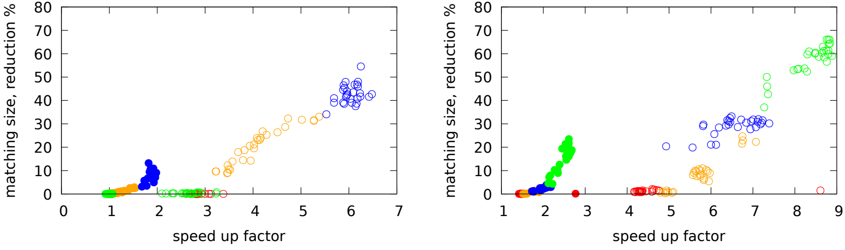
Each dot corresponds to the result obtained for ride matching for a different day of February/March 2011. Each color corresponds to a different time intervals: [1.00–1.20 a.m.], [7.00–7.20 a.m.], [1.00–1.20 p.m.], [7.00–7.20 p.m.]



(a) Manhattan (NYC) (b) Singapore

Figure: Number of issued trip requests (x-axis) VS matching size reduction (y-axis)

Each dot corresponds to the result obtained for ride matching for a different day of February/March 2011. Each color corresponds to a different time intervals: [1.00–1.20 a.m.], [7.00–7.20 a.m.], [1.00–1.20 p.m.], [7.00–7.20 p.m.]



(a) Manhattan (NYC) (b) Singapore

Figure: Achieved time speed up (x-axis) vs matching size reduction (y-axis)

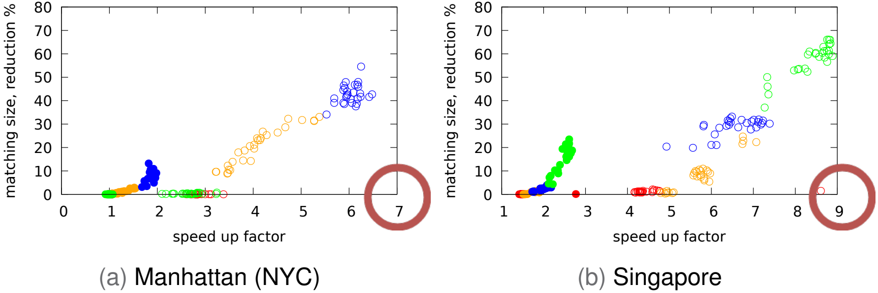
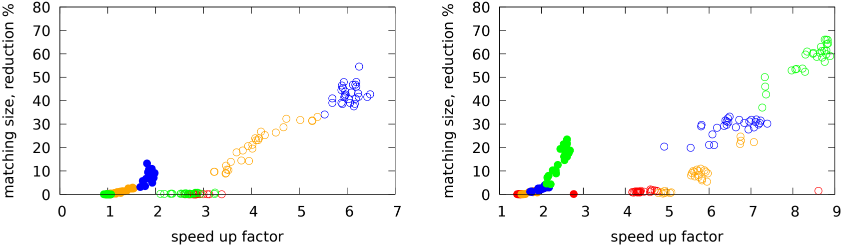
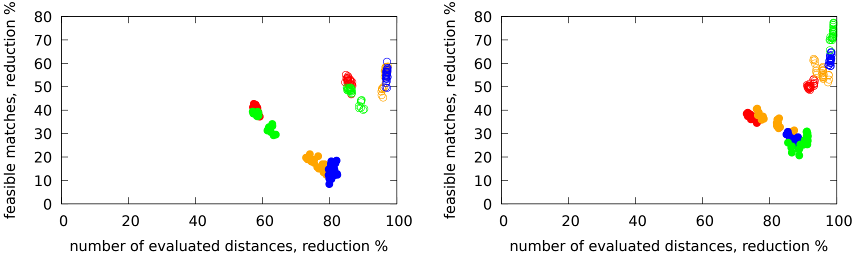


Figure: Achieved time speed up (x-axis) vs matching size reduction (y-axis)



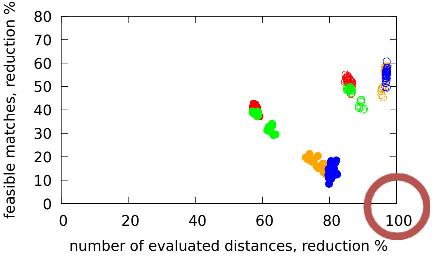
(a) Manhattan (NYC) (b) Singapore

Figure: Achieved time speed up (x-axis) vs matching size reduction (y-axis)

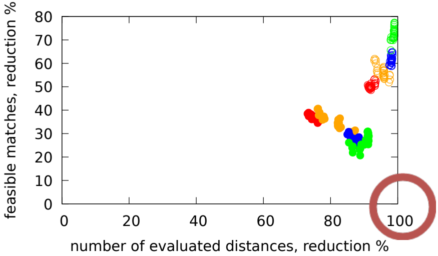


(a) Manhattan (NYC) (b) Singapore

Figure: Number of shortest path queries (x-axis) VS feasible matches reduction (y-axis)

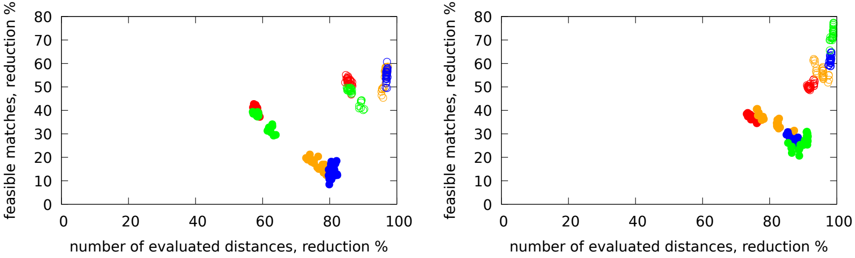


(a) Manhattan (NYC)



(b) Singapore

Figure: Number of shortest path queries (x-axis) vs feasible matches reduction (y-axis)



(a) Manhattan (NYC) (b) Singapore

Figure: Number of shortest path queries (x-axis) vs feasible matches reduction (y-axis)

Outline

Conclusion & Future Work

Conclusion.

- We managed to reduce the number of shortest path calculations by means of our novel locality filtering approach
- We managed to sparsify the shareability network $\Rightarrow$ we saved space and reduced time needed to execute the matching phase
- We obtain the same quality of results as the approach of [P. Santi *et al.*, 2014]

Future Work.

- Investigate related problems (e.g. the minimum fleet problem)
- Provide a data-parallel version of the code.
- Implement privacy mechanisms.

Outline

References

P. Santi, G. Resta, M. Szell, S. Sobolevsky, S.H. Strogatz, and C. Ratti
Quantifying the benefits of vehicle pooling with shareability networks.
Proceedings of the National Academy of Sciences, 2014.

Boost c++ Library.
...one of the most highly regarded and expertly designed C++ library projects in the world.
www.boost.org

Francesco Tosoni, M.Sc.

Locality Filtering for efficient ride sharing platforms

Mauriana Pesaresi Series, Seminar #6