Towards Parallel Detection of Movement Patterns in Large Spatio-temporal Datasets

Oral Qualification

Andres Calderon

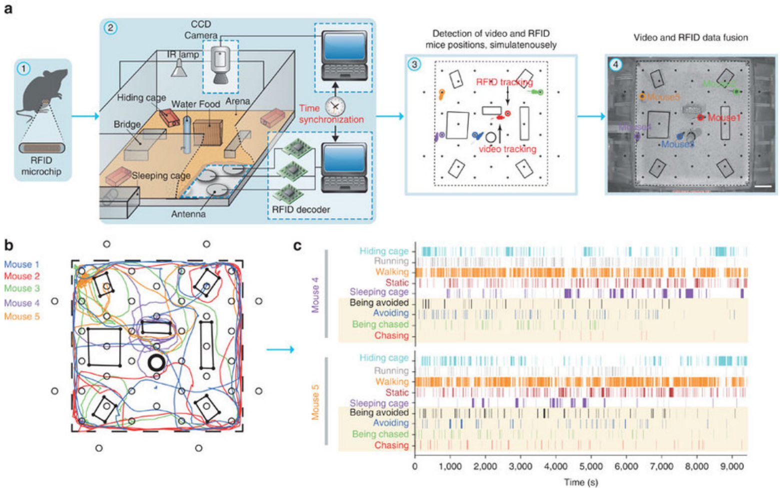
April 2, 2017

Trajectory datasets

- Anything that could move (or could be moved), will be tracked...
  - Sensors, Sensors everywhere...
  - Smart phones, GPS, RFID, WiFi, Bluetooth, ZigBee, IoT, Satellites, Drones, Surveillance cameras...

Applications

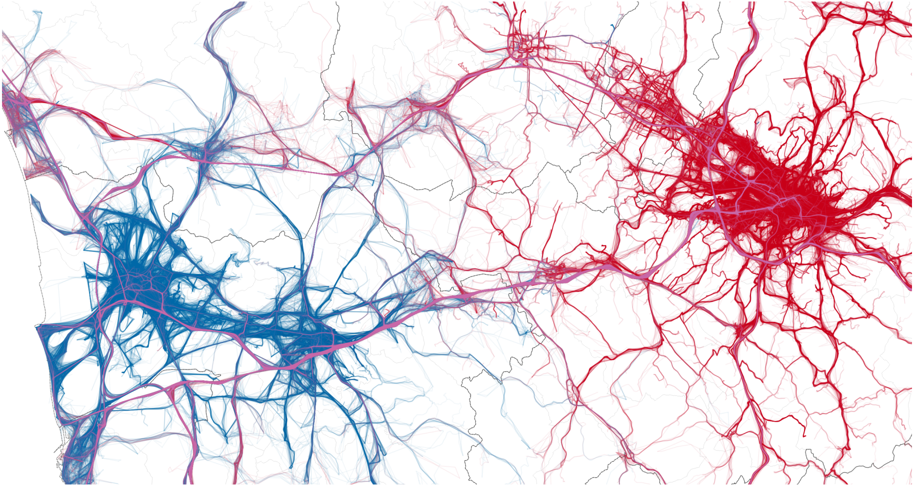
Tracking social behavior of animal colonies.



(Weissbrod et al, Nature'13)

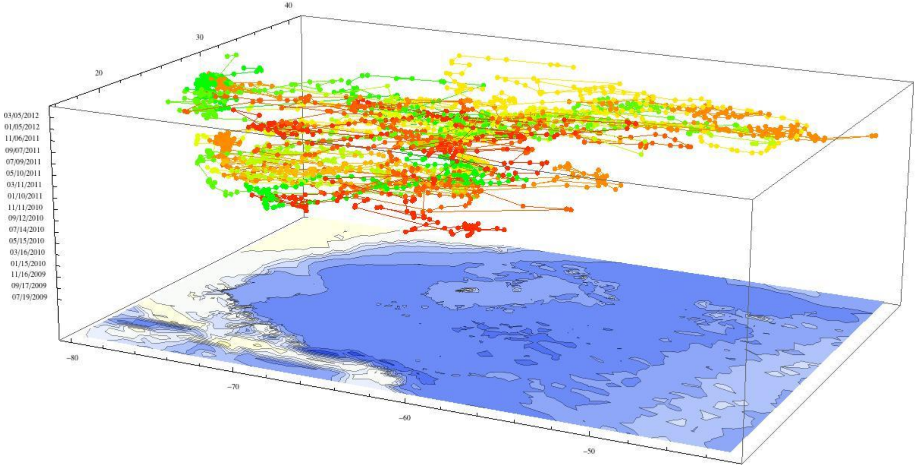
Applications

Are you a returner or an explorer? Ask Big Data.



(Pappalardo et al, Nature'15)

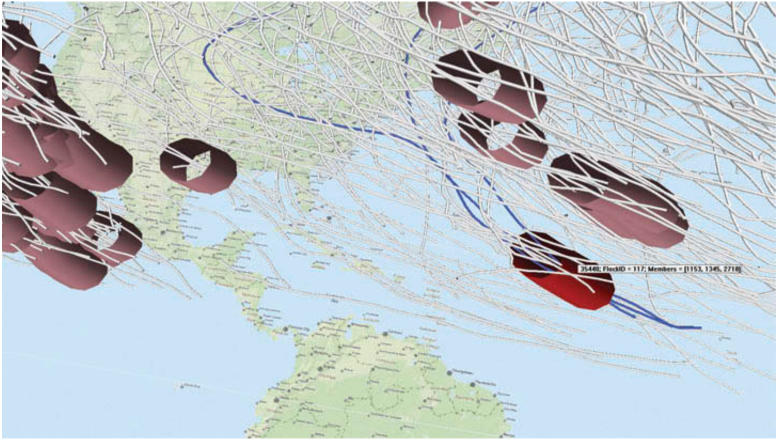
Tiger shark data analysis.



(Antonov, 2012)

Applications

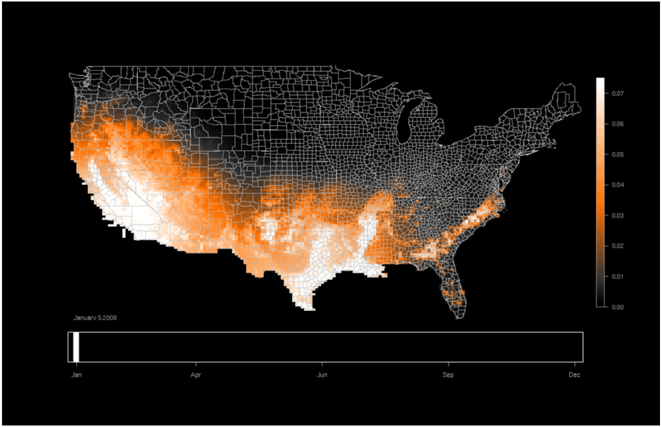
Visual mining of cyclone paths.



(Turdukulov, IJGIS'14)

Applications

Bird migration forecast.

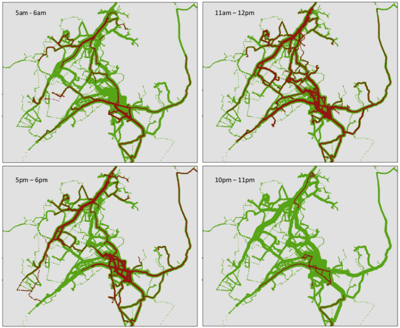


(Sullivan et al, Biological Conservation'14)

Complex movement patterns

Previous works focus on traditional queries:

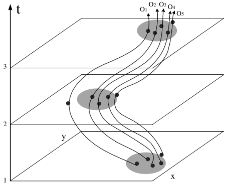
- Range
- Nearest Neighbors
- Similarity



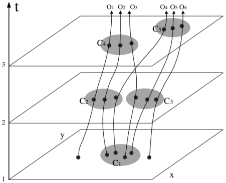
Complex movement patterns

Recent works look for the aggregate behavior:

- Moving clusters
- Convoys
- Flocks
- Swarms
- Gatherings



(b) Flock, convoy and swarm



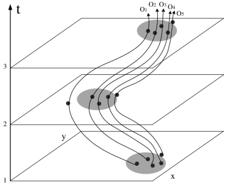
(c) Gathering

(Zheng et al, ICDE'13)

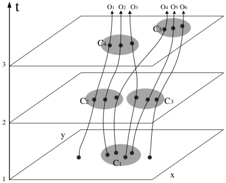
Complex movement patterns

Recent works look for the aggregate behavior of trajectories as groups:

- Moving clusters
- Convoys
- **Flocks**
- Swarms
- Gatherings



(b) Flock, convoy and swarm



(c) Gathering

(Zheng et al, ICDE'13)

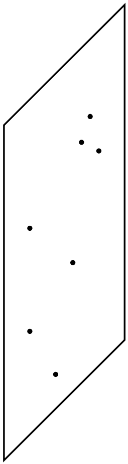
Outline

What is a flock???

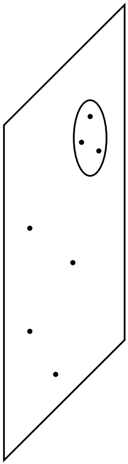
Definition ($(\mu, \epsilon, \delta) - flock$)

Sets of at least μ objects moving close enough (ε) for at least δ time intervals (Benkert et al, 2008).

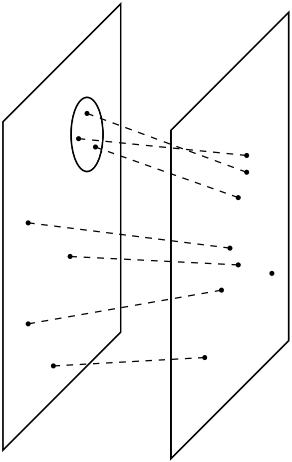
BFE algorithm (Vieira et al, 2009)



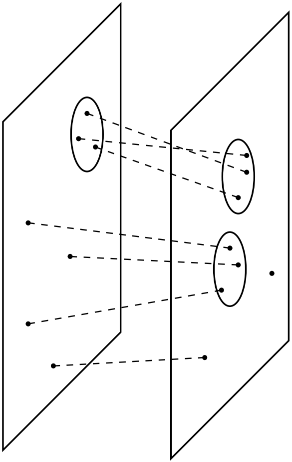
BFE algorithm (Vieira et al, 2009)



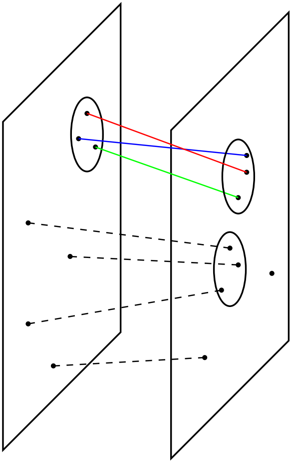
BFE algorithm (Vieira et al, 2009)



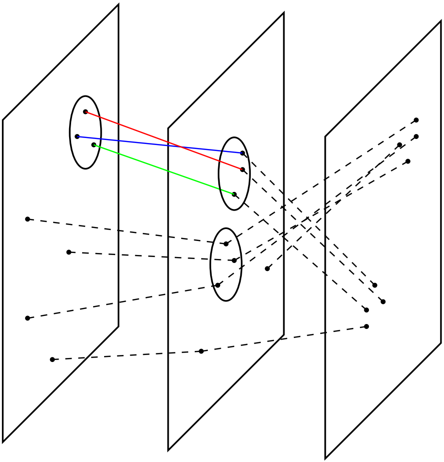
BFE algorithm (Vieira et al, 2009)



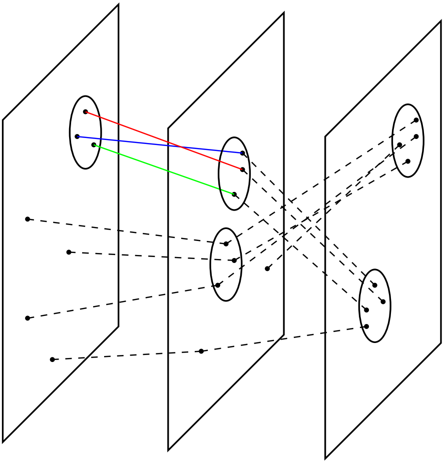
BFE algorithm (Vieira et al, 2009)



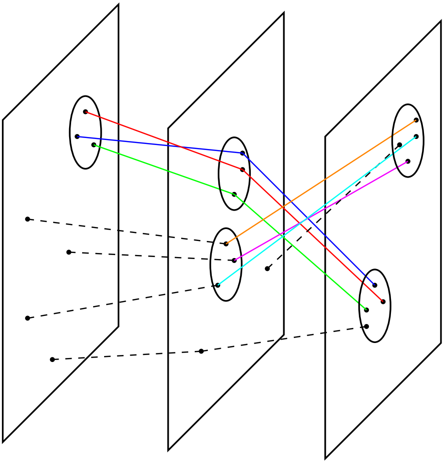
BFE algorithm (Vieira et al, 2009)



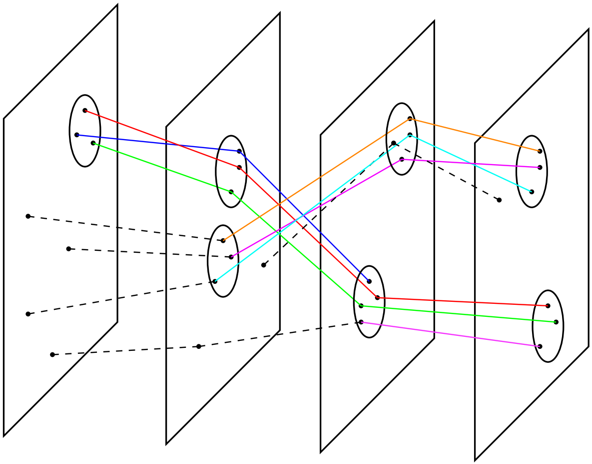
BFE algorithm (Vieira et al, 2009)



BFE algorithm (Vieira et al, 2009)



BFE algorithm (Vieira et al, 2009)



Motivation

Why is important to focus on flocks and finding disks???

- Why are moving flock patterns important?
  - They capture the collective behavior of trajectories as groups.
  - A general model for other movement patterns.
- Why is the finding of disks important?
  - It is no trivial, disks can be at any location.
  - It has a high complexity ($\mathcal{O}(2n^2)$).

Outline

Outline

BFE implementation

- On-Line Discovery of Flock Patterns in Spatio-Temporal Data (Vieira et al, SIGSPATIAL'09).
- Implementation available at `https://github.com/poldrosky/FPFlock`.

PBFE implementation

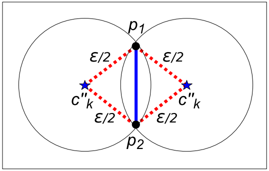
- Written in Scala using Simba 1.6.3.
  - Tested by visual inspection and counting number of found disks.
- Code available at `https://github.com/aocalderon/PhD/tree/master/Y2Q1/SDB/Project/Code/Scripts/pbfe2`.

PBFE implementation

- Simba (Xie et al, SIGMOD'16) provides rich spatial operations through both SQL and the DataFrame API.
- Two-layer spatial indexing, cost-based optimization, open source.
- `DISTANCE JOIN` and `CIRCLERANGE` operators were the key!!!

PBFE implementation

```
SELECT
        *
FROM
        points p1
DISTANCE JOIN
        points p2
ON
        POINT(p2.x, p2.y) IN CIRCLERANGE(POINT(p1.x, p1.y), ε)
WHERE
        p1.id < p2.id
```



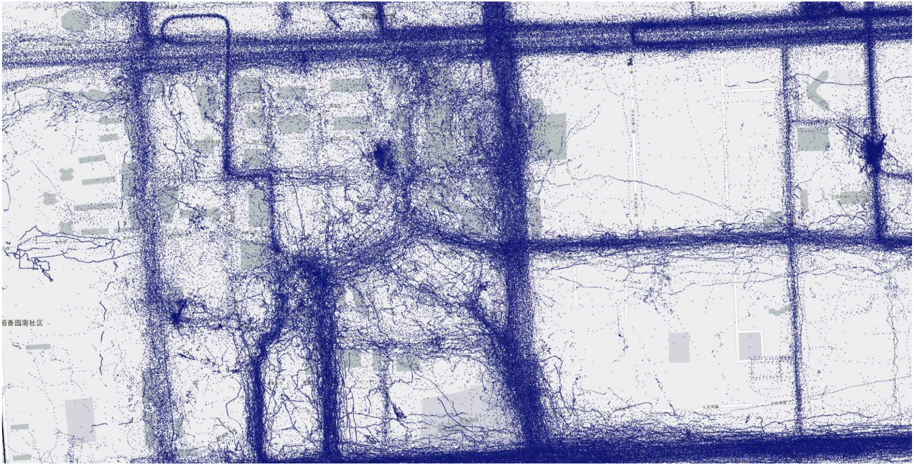
Outline

Dataset

- **Beijing** from Geolife project[1].
  - 182 users in a period of over three years (from April 2007 to August 2012).
  - 17,621 trajectories.
  - $\approx$18 million points (no duplicates).

[1] http://tinyurl.com/j7t2cao

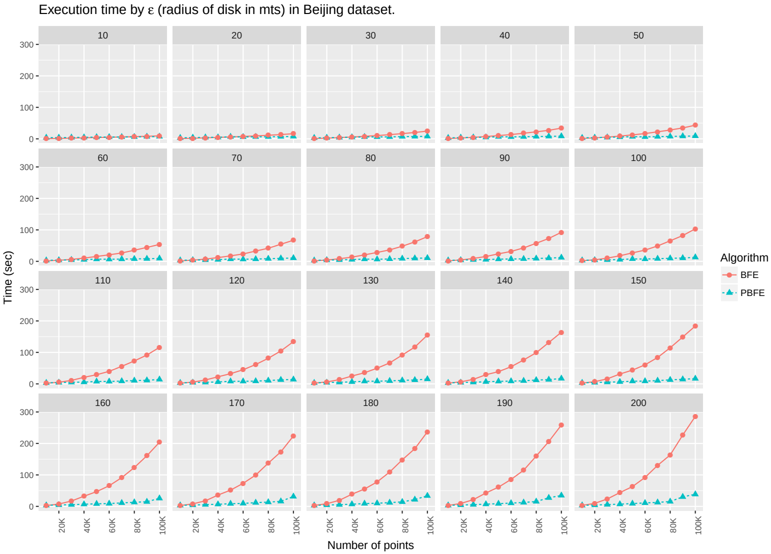
Beijing dataset



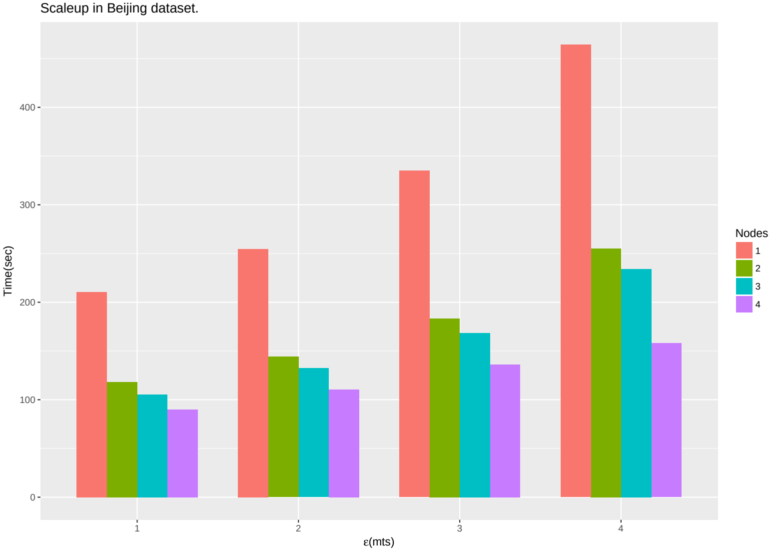
Setup

- Single-node.
- Processor: 4-core Intel(R) Core(TM) i5-2400S CPU @ 2.50GHz
- RAM: 8 GB.
- Ubuntu 16.04 LTS, Simba/Spark 1.6.3.

Execution time

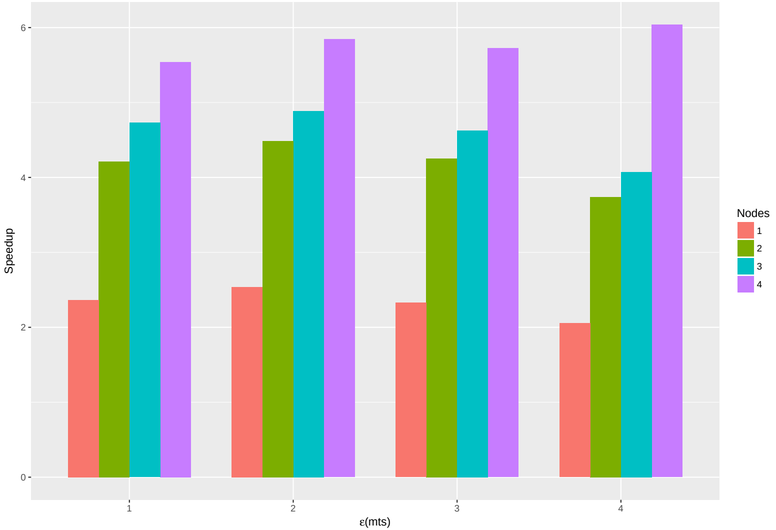


Scaleup


Scaleup in Beijing dataset.
Time(sec)
400
300
200
100
0
1
2
3
4
ε(mts)
Nodes
1
2
3
4

Speedup

Speedup in Beijing dataset.

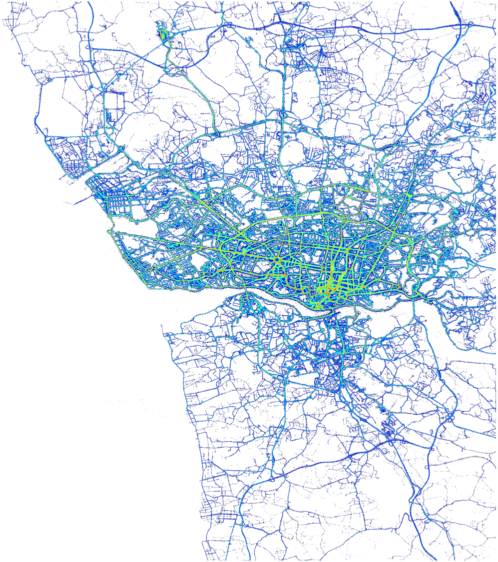


Dataset

- **Porto** from ECML/PKDD'15 Taxi Trajectory Prediction Challenge[2].
  - A complete year (from 01/07/2013 to 30/06/2014).
  - Trajectories for all the 442 taxis running in the city of Porto, in Portugal.
  - ≈17.7 million points (no duplicates).

[2] `http://tinyurl.com/zzbtlt9`

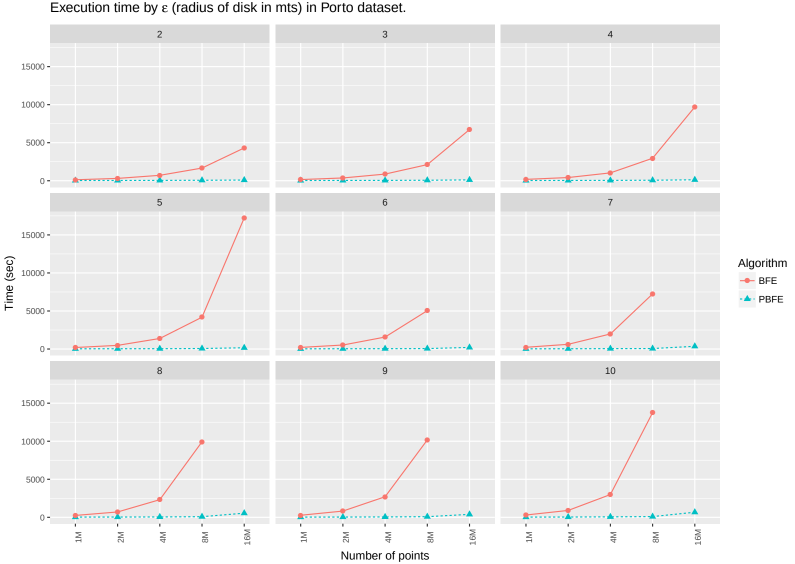
Porto dataset



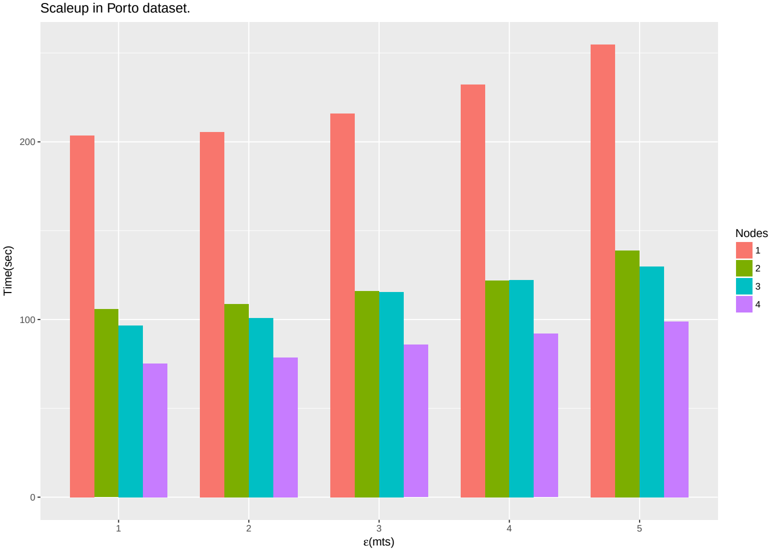
Setup

- 4-node cluster at DBLab.
- Processors: 8-core Intel(R) Xeon(R) CPU E3-1230 V2 @ 3.30GHz
- RAM: 15.5 GB.
- Centos 6.8, Simba/Spark 1.6.3.

Execution time

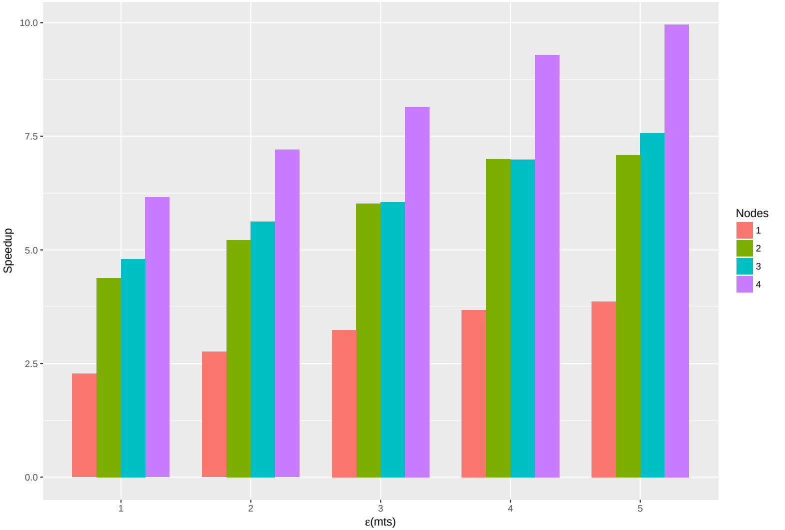


Scaleup


Scaleup in Porto dataset.
Time(sec)
200
100
0
1
2
3
4
5
ε(mts)
Nodes
1
2
3
4

Speedup

Speedup in Porto dataset.

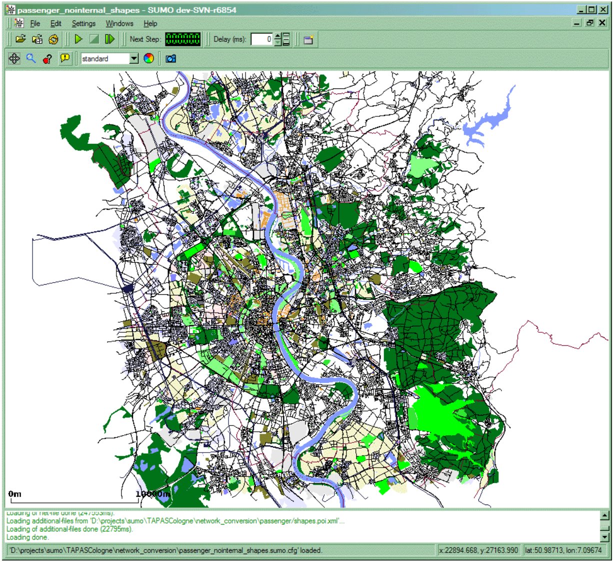


Dataset

- **Cologne** from the SUMO (Simulation of Urbar MObility) project[3].
  - Simulation scenario describing a whole-day traffic in the city of Cologne (Germany).
  - Data demand based on traveling habits and infrastructure (TAPAS system).
  - ≈70 million points (no duplicates).

[3] https://tinyurl.com/l9lbw6b

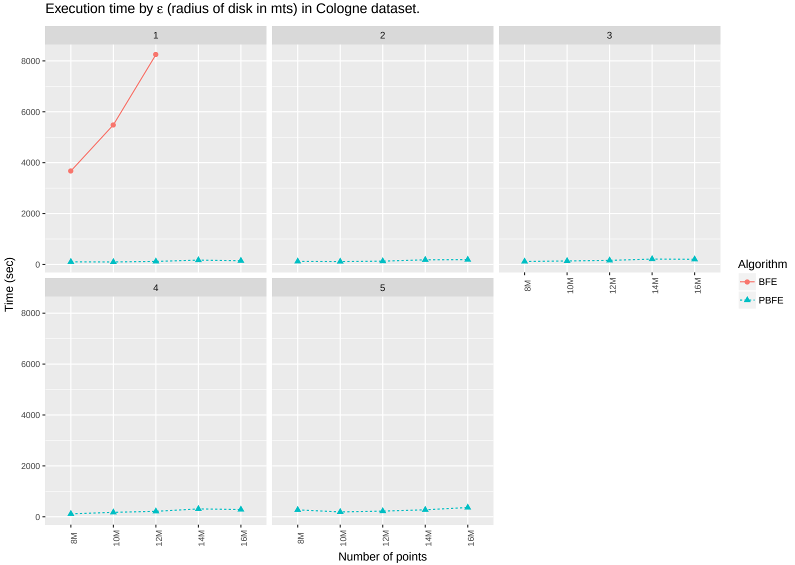
Cologne dataset



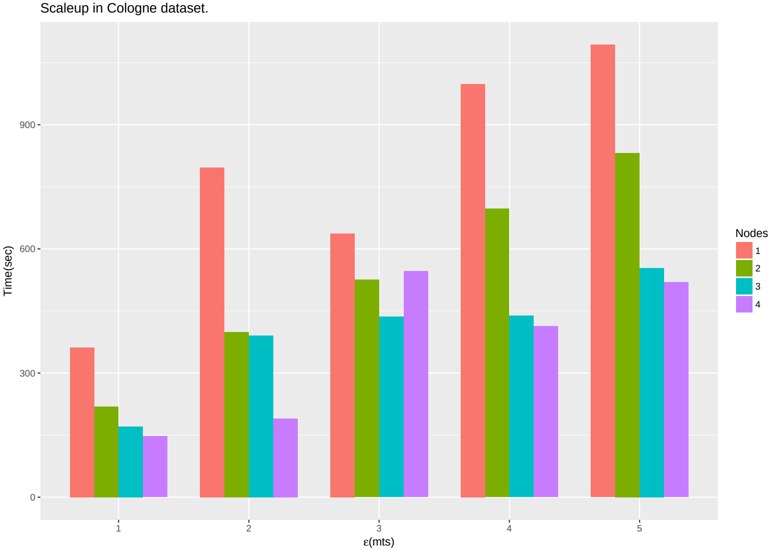
Setup

- 4-node cluster at DBLab.
- Processors: 8-core Intel(R) Xeon(R) CPU E3-1230 V2 @ 3.30GHz
- RAM: 15.5 GB.
- Centos 6.8, Simba/Spark 1.6.3.

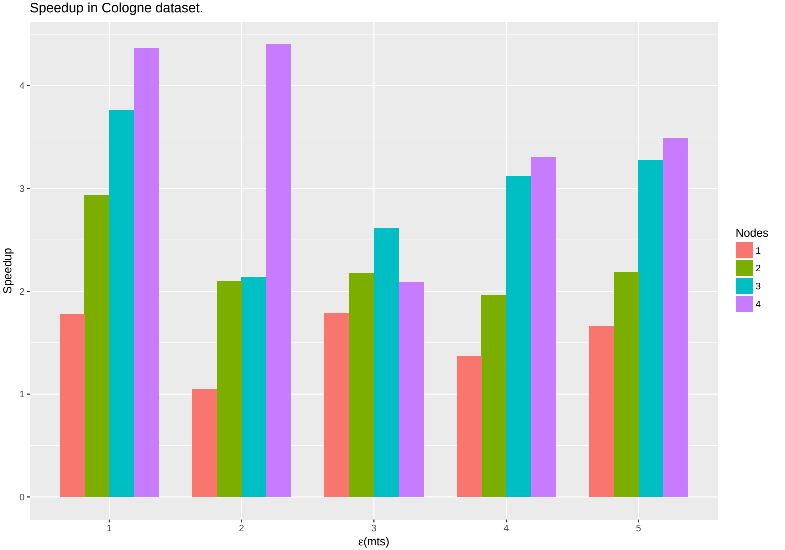
Execution time



Scaleup


Scaleup in Cologne dataset.
Time(sec)
900
600
300
0
1
2
3
4
5
ε(mts)
Nodes
1
2
3
4

Speedup


Speedup in Cologne dataset.
Speedup
0
1
2
3
4
1
2
3
4
5
ε(mts)
Nodes
1
2
3
4

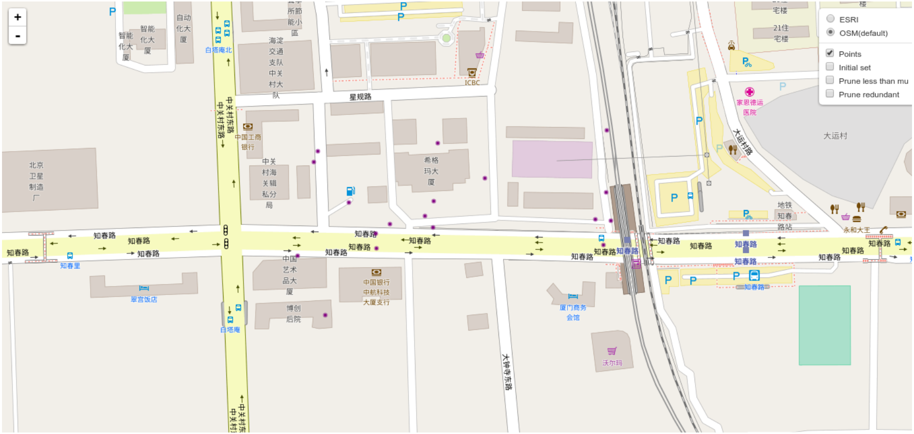
Outline

Outline

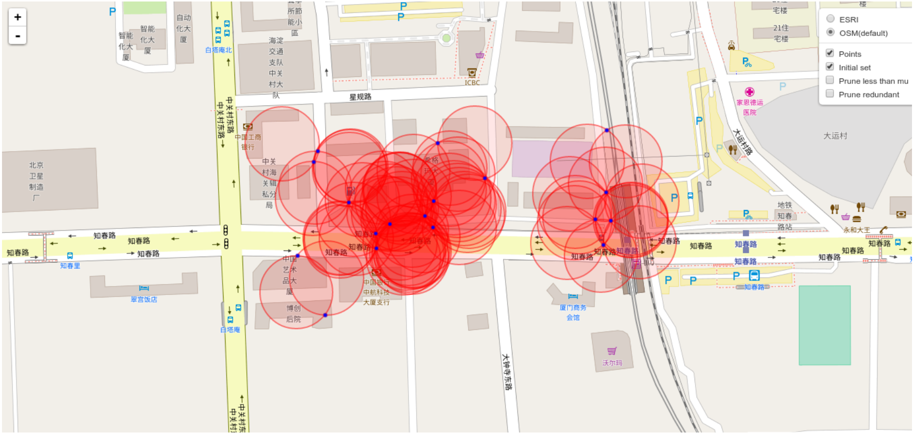
Finding maximal disks

- The first stage of BFE finds all possible disk locations for each pair of "close enough" points...
- This candidate set still have redundant and under-the-threshold (μ) disks...
- BFE uses a simple approach to filter the disks but it can be costly ($O(n^2)$)

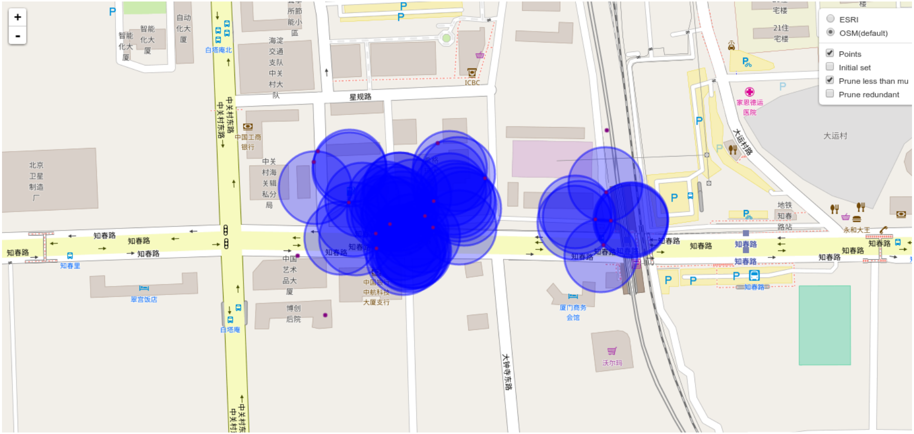
Finding maximal disks



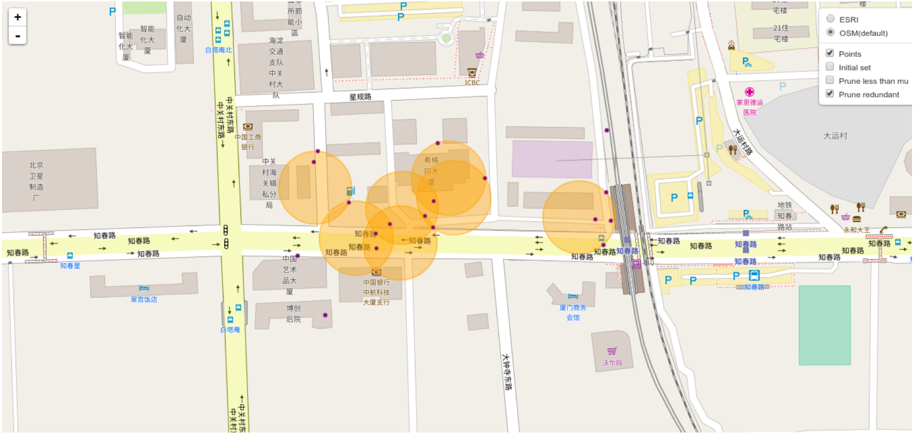
Finding maximal disks



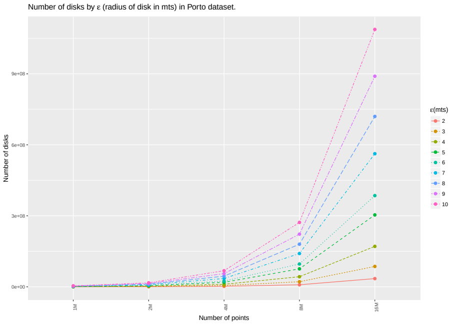
Finding maximal disks

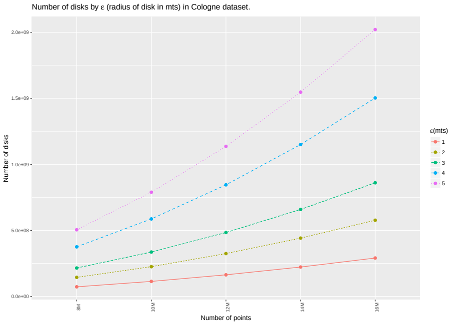


Finding maximal disks



Number of disks can be huge...





Finding maximal disks using Maximal Pattern algorithms

- Points enclosed by disks become transactions...
- Apply well-know algorithms (LCM in this case)...
- Filter patterns with length less than μ.

A Frequent/Closed/Maximal pattern example...

Suppose a database D contains 4 transactions:

$$D = \{\langle a_1, a_2, ..., a_{100}\rangle; \langle a_1, a_2, ..., a_{100}\rangle; \langle a_{20}, a_{21}, ..., a_{80}\rangle; \langle a_{40}, a_{41}, ..., a_{60}\rangle\}$$

With $min\_sup = 2$:

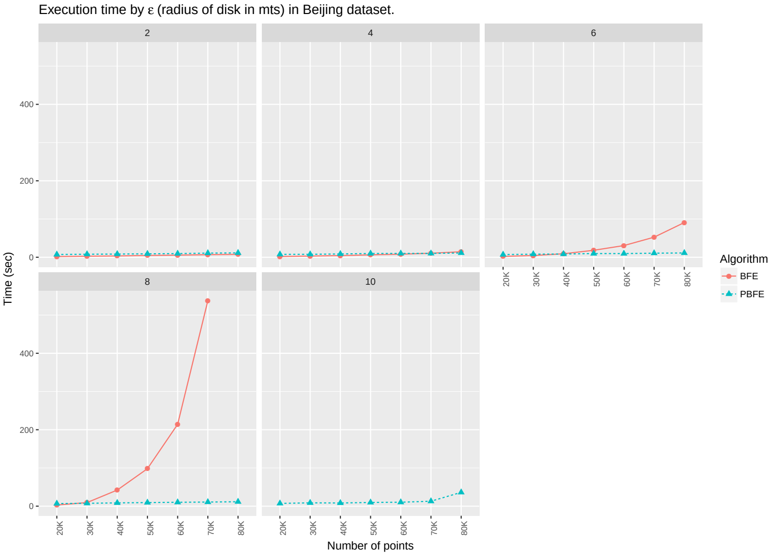
- $\text{F} = \approx 1.27 * 10^{30}$
- $\text{C} = \{\{a_1, a_2, ..., a_{100} : 2\}; \{a_{20}, a_{21}, ..., a_{80} : 3\}; \{a_{40}, a_{41}, ..., a_{60} : 4\}\}$
- $\text{M} = \{\{a_1, a_2, ..., a_{100} : 2\}\}$

Outline

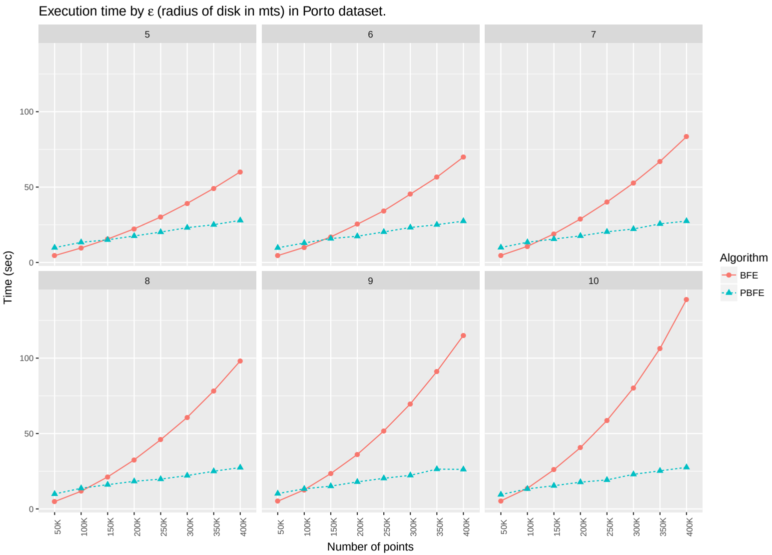
Setup

- 4-node cluster at DBLab.
- Processors: 8-core Intel(R) Xeon(R) CPU E3-1230 V2 @ 3.30GHz
- RAM: 15.5 GB.
- Centos 6.8, Simba/Spark 1.6.3.

Beijing - Execution time



Porto - Execution time



Outline

Conclusions

- An implementation of a parallel method to detect a maximal set of disks for the BFE algorithm has been presented.
- The method proves to be scalable and reliable.
- Execution time improves up to 3 orders of magnitude compared to the sequential code.
- Good behavior of Scaleup and Speedup metrics.

Conclusions

- An implementation of a parallel method to detect a maximal set of disks for the BFE algorithm has been presented.
- The method proves to be scalable and reliable.
- Execution time improves up to 3 orders of magnitude compared to the sequential code.
- Good behavior of Scaleup and Speedup metrics.

Conclusions

- An implementation of a parallel method to detect a maximal set of disks for the BFE algorithm has been presented.
- The method proves to be scalable and reliable.
- Execution time improves up to 3 orders of magnitude compared to the sequential code.
- Good behavior of Scaleup and Speedup metrics.

Conclusions

- An implementation of a parallel method to detect a maximal set of disks for the BFE algorithm has been presented.
- The method proves to be scalable and reliable.
- Execution time improves up to 3 orders of magnitude compared to the sequential code.
- Good behavior of Scaleup and Speedup metrics.

Conclusions

- An implementation of a parallel method to detect a maximal set of disks for the BFE algorithm has been presented.
- The method proves to be scalable and reliable.
- Execution time improves up to 3 orders of magnitude compared to the sequential code.
- Good behavior of Scaleup and Speedup metrics.

Conclusions

- An implementation of a parallel method to detect a maximal set of disks for the BFE algorithm has been presented.
- The method proves to be scalable and reliable.
- Execution time improves up to 3 orders of magnitude compared to the sequential code.
- Good behavior of Scaleup and Speedup metrics.

Future work

- Work on a parallel strategy to join the sets of valid disks between time intervals.
- Test the approach with other spatio-temporal patterns.

Future work

- Work on a parallel strategy to join the sets of valid disks between time intervals.
- Test the approach with other spatio-temporal patterns.

Future work

- Work on a parallel strategy to join the sets of valid disks between time intervals.
- Test the approach with other spatio-temporal patterns.

Future work

- Work on a parallel strategy to join the sets of valid disks between time intervals.
- Test the approach with other spatio-temporal patterns.

Thank you!!!

Do you have any question?