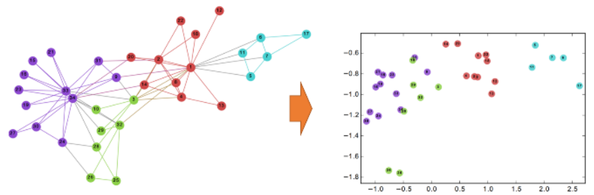
Fast and Accurate Mining of Evolving & Trajectory Networks

Manos Papagelis

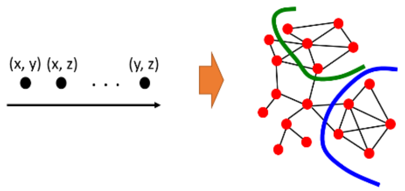
York University, Toronto, Canada

Current Research focus



A. Network Representation Learning

B. Trajectory Network Mining



C. Streaming & Dynamic Graphs

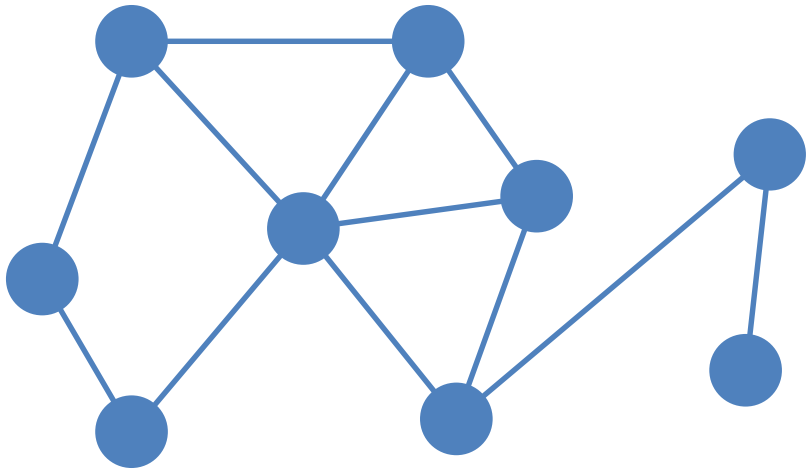
D. Social Media Mining & Analysis

E. City Science / Urban Informatics / IoT

F. Natural Language Processing

EvoNRL: Evolving Network Representation Learning Based on Random Walks

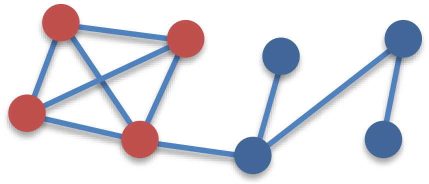
Joint work with Farzaneh Heidari



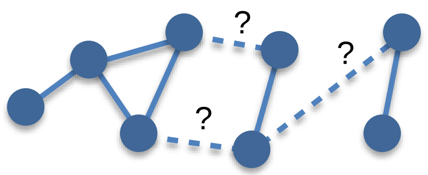
networks

(universal language for describing complex data)

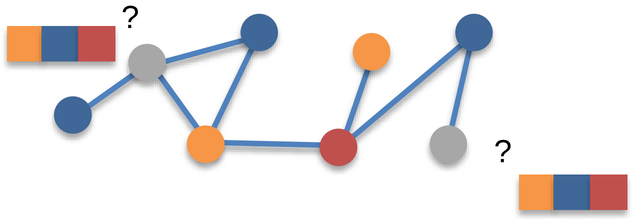
Classical ML Tasks in Networks



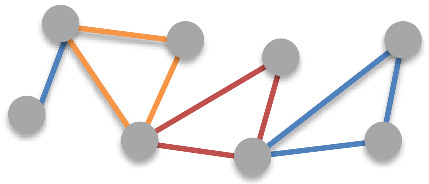
community detection



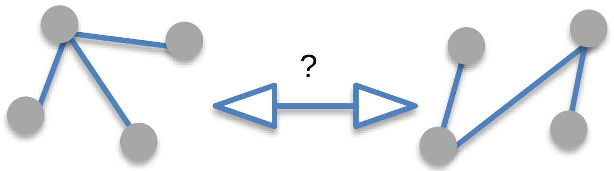
link prediction



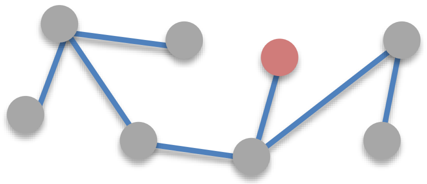
node classification



triangle count



graph similarity

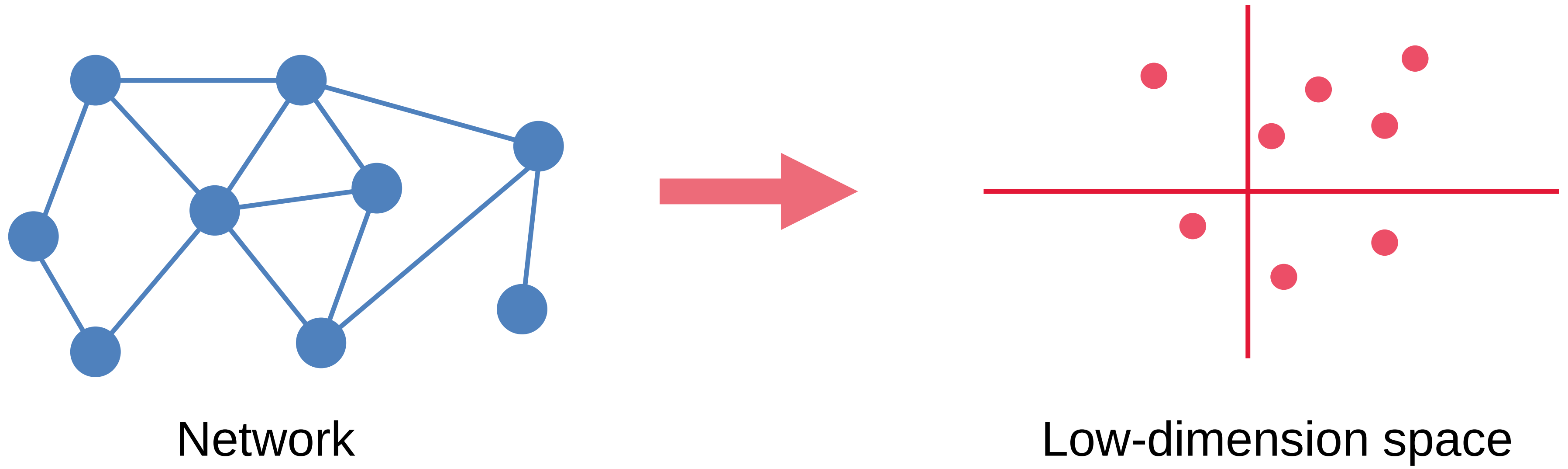


anomaly detection

Limitations of Classical ML:

- expensive computation (high dimension computations)
- extensive domain knowledge (task specific)

Network Representation Learning (NRL)

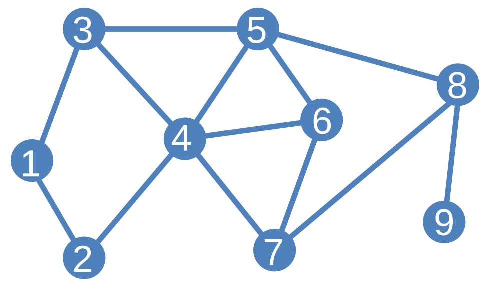


several network structural properties can be learned/embedded
(**nodes, edges, subgraphs, graphs, …**)

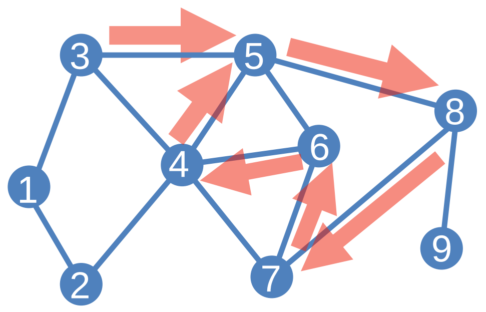
Premise of NRL:

- faster computations (low dimension computations)
- agnostic domain knowledge (task independent)

Random Walk-based NRL



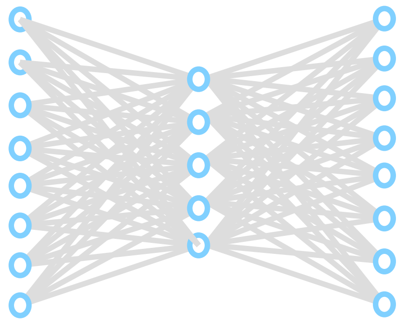
Input network



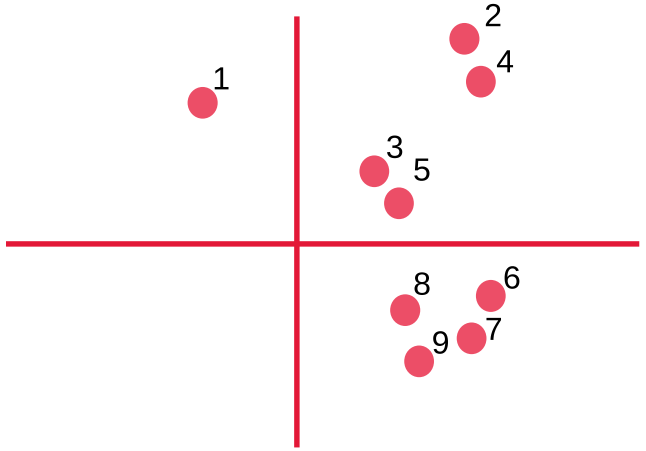
Obtain a set of random walks

1	3 5 8 7 6 4 5
2	1 3 5 8 7 6 5
.	.
.	.
.	.
.	.
87	8 5 4 3 5 6 7
88	4 5 6 7 8 9
89	2 1 3 5 6 7 8
90	7 4 2 1 3 5 6

Treat the set of random walks as sentences



Feed sentences to a Skip-gram NN model



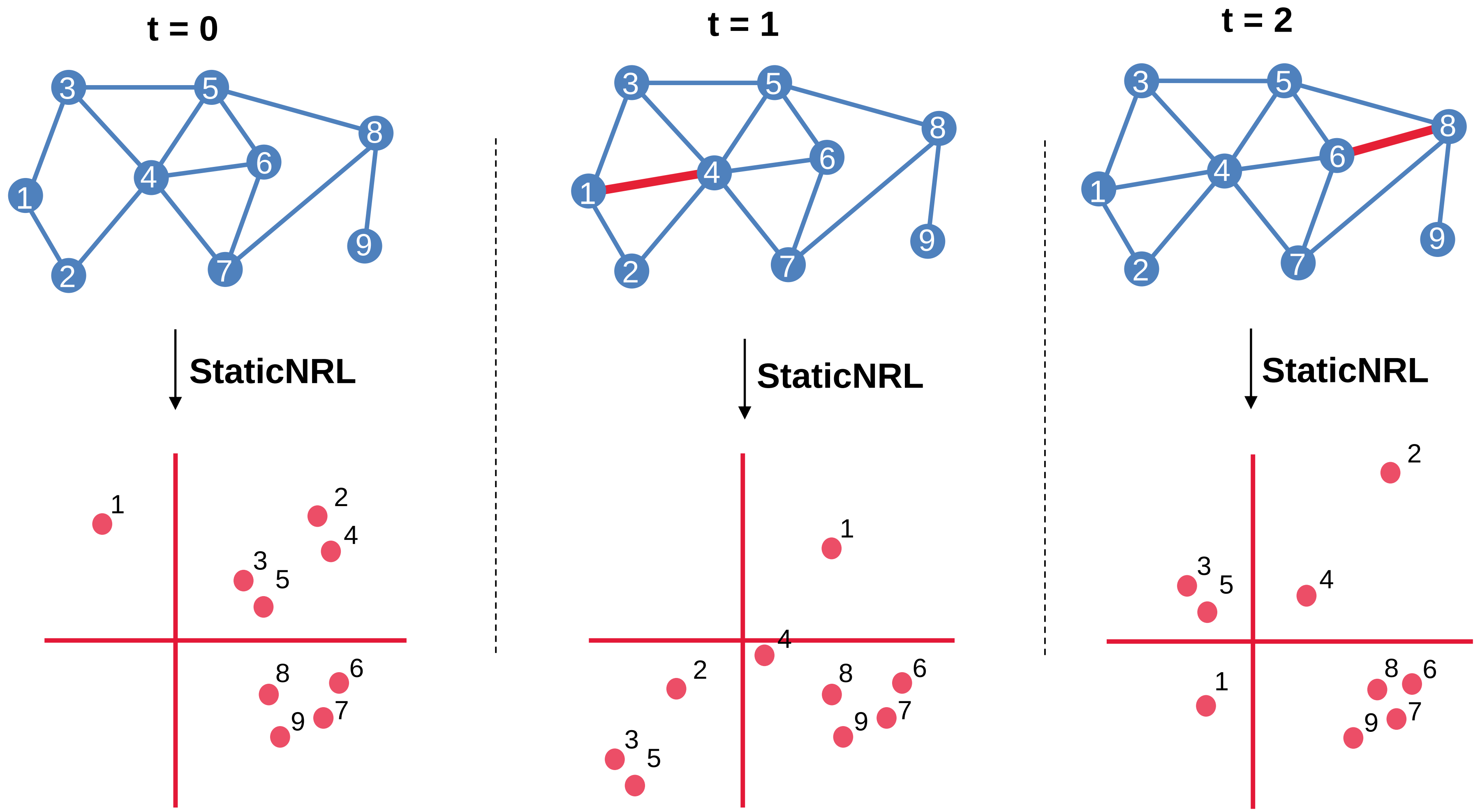
Learn a vector representation for each node

StaticNRL
(DeepWalk, node2vec, …)

but real-world networks are constantly evolving

Evolving Network Representations Learning

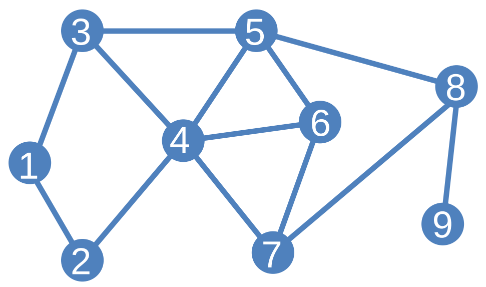
Naive Approach



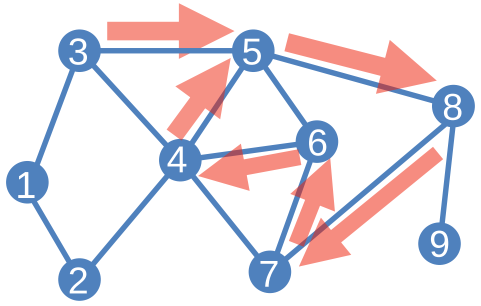
Impractical (expensive, incomparable representations)

EvoNRL Key Idea

dynamically maintain a valid set of random walks for every change in the network



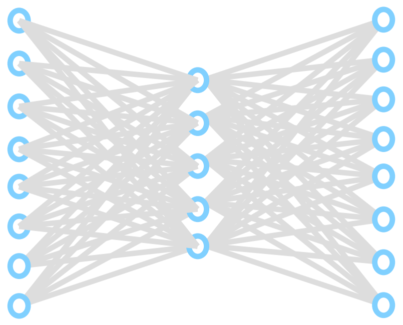
Input network



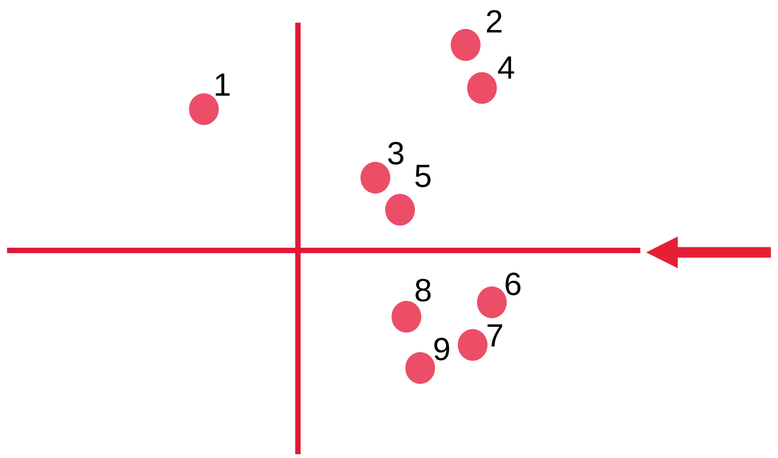
Obtain a set of random walks

1	3 5 8 7 6 4 5
2	1 3 5 8 7 6 5
.	.
.	.
.	.
.	.
87	8 5 4 3 5 6 7
88	4 5 6 7 8 9
89	2 1 3 5 6 7 8
90	7 4 2 1 3 5 6

Treat the set of random walks as sentences



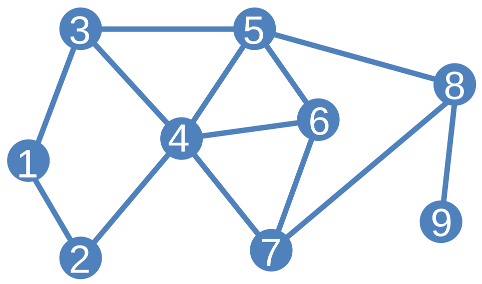
Feed sentences to a Skip-gram NN model



Learn a vector representation for each node

Example: Edge Addition

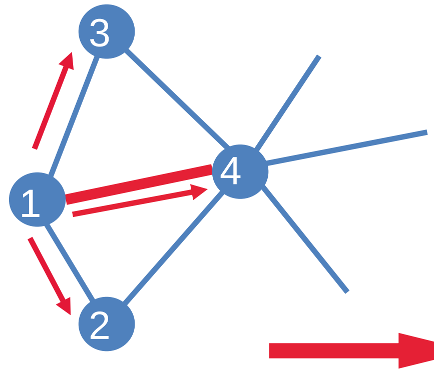
t = 0



1	3 5 8 7 6 4 5
2	1 3 5 8 7 6 5
.	.
.	.
.	.
.	.
87	8 5 4 3 5 6 7
88	4 5 6 7 8 9 8
89	2 1 3 5 6 7 8
90	7 4 2 1 3 5 6

addition of edge (1, 4)

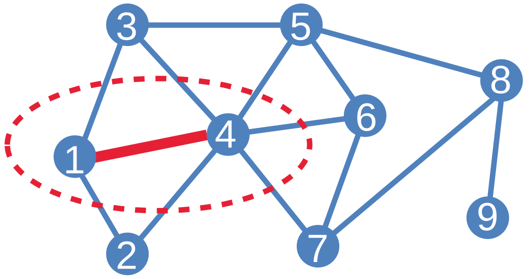
$\longrightarrow$



need to update the RW set

$\longrightarrow$

t = 1



simulate the rest of the RW

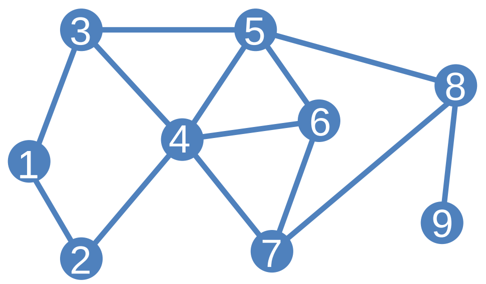
2	1 4 $\overbrace{3\ 5\ 6\ 7\ 8}$

1	3 5 8 7 6 4 5
2	1 3 5 8 7 6 5
.	.
.	.
.	.
.	.
87	8 5 4 3 5 6 7
88	4 5 6 7 8 9 8
89	2 1 3 5 6 7 8
90	7 4 2 1 3 5 6

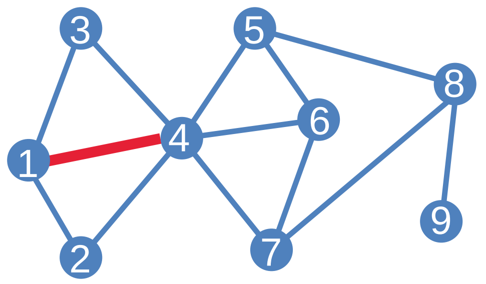
similarly for edge deletion, node addition/deletion

Efficiently Maintaining a Set of Random Walks

EvoNRL Operations



+ edge(n_1, n_2)



2	1 **4** 3 5 6 7 8

1	3 5 8 7 6 4 5
2	1 3 5 8 7 6 5
.	.
.	.
.	.
.	.
87	8 5 4 3 5 6 7
88	4 5 6 7 8 9
89	2 1 3 5 6 7 8
90	7 4 2 1 3 5 6

1	3 5 8 7 6 4 5
2	1 3 5 8 7 6 5
.	.
.	.
.	.
.	.
87	8 5 4 3 5 6 7
88	4 5 6 7 8 9
89	2 1 3 5 6 7 8
90	7 4 2 1 3 5 6

Operations on RW

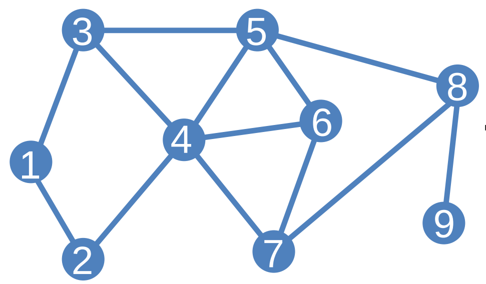
Search a node

Delete a RW

Insert a new RW

need for an efficient indexing data structure

EvoNRL Indexing



1	3 5 8 7 6 4 5
2	1 3 5 8 7 6 5
.	.
.	.
.	.
.	.
87	8 5 4 3 5 6 7
88	4 5 6 7 8 9
89	2 1 3 5 6 7 8
90	7 4 2 1 3 5 6

elastic

each node is **a keyword**
each RW is **a document**
a set of RWs is **a collection of documents**

Term	**Frequency**	**Postings and Positions**
1	3	< 2, 1 >, < 89, 2 >, < 90, 4 >
2	2	<89, 1>, <90, 3>
3	5	<1, 1>, <2, 1>, <87, 3>, <89, 3>, <90, 5>
4	4	<1, 6>, <87, 3>, <90, 2>
5	9	<1, 2>, <1, 7>, <2, 3>, <2, 7>, <87, 5>, <88, 2>, <89, 4>, <90, 6>
6	6	<1, 5>, <2, 6>, <87, 6>, <88, 3>, <89, 3>, <90, 5>
7	5	<1, 4>, <2, 5>, <87, 7>, <88, 4>, <89, 6>, <90, 7>
8	5	<1, 3>, <2, 4>, <87, 1>, <88, 6>, <89, 7>
9	1	<88, 7>

Evaluation of EvoNRL

Evaluation: EvoNRL vs StaticNRL

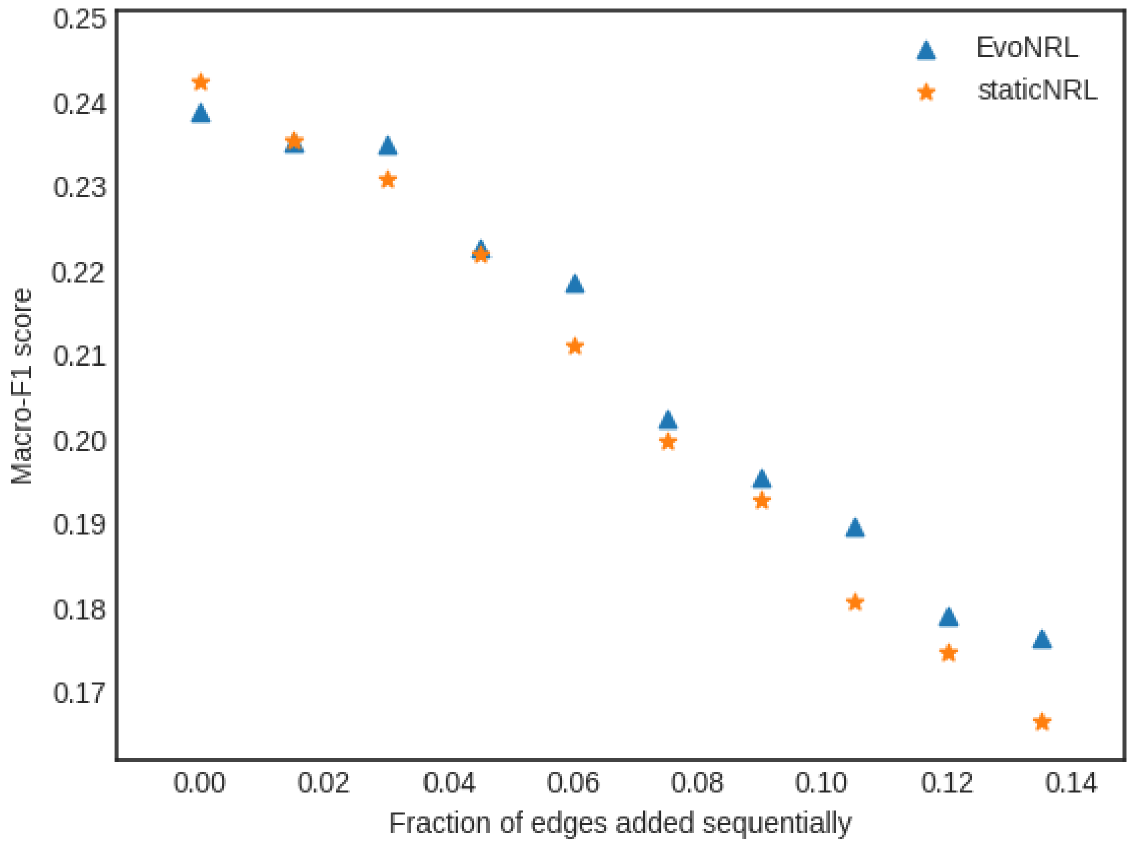
Accuracy

- EvoNRL **≈** StaticNRL

Running Time

- EvoNRL **<<** StaticNRL

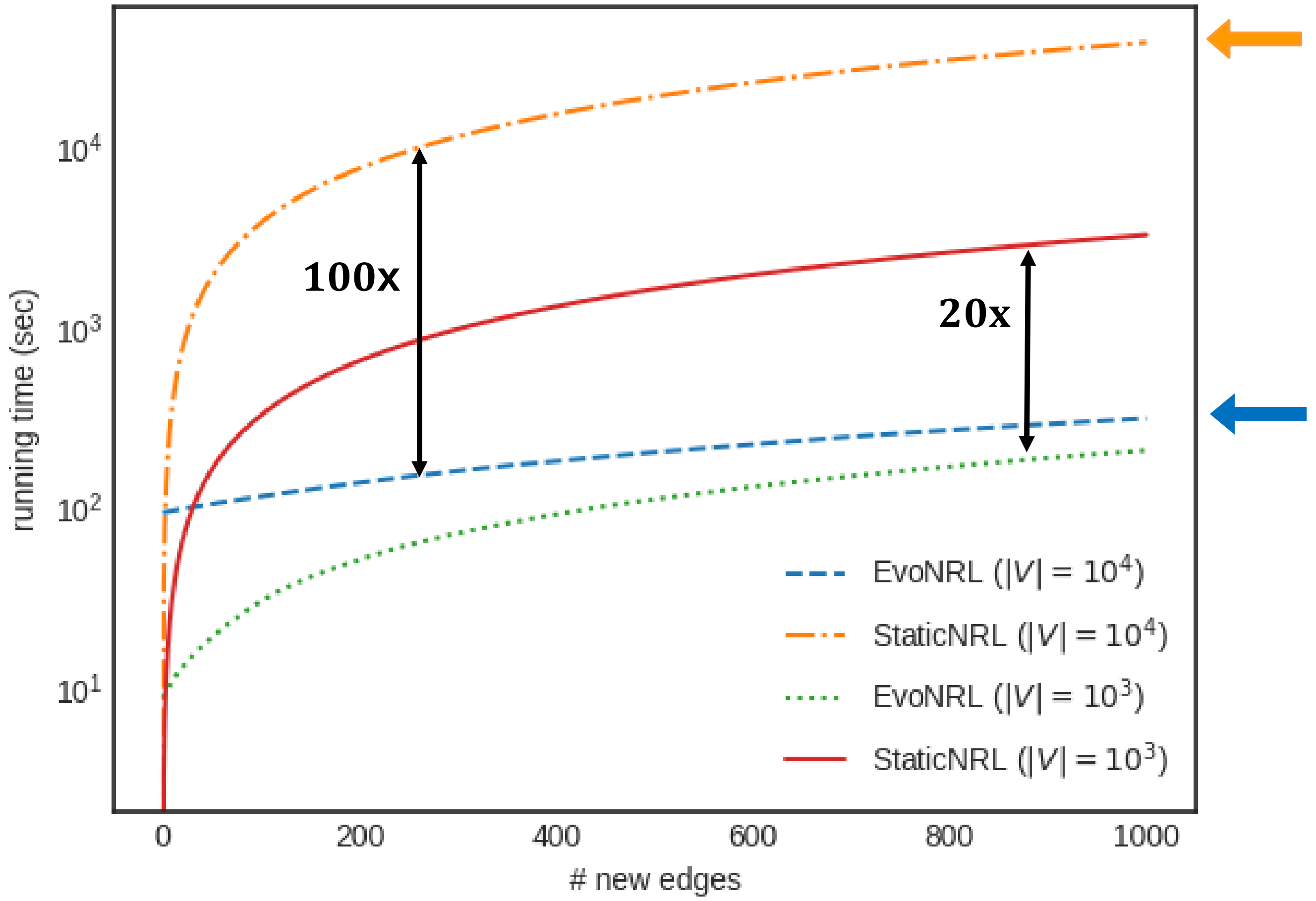
Accuracy: edge addition



EvoNRL has similar accuracy to StaticNRL

(similar results for edge deletion, node addition/deletion)

Time Performance



EvoNRL performs orders of time faster than StaticNRL

Takeaway

how can we learn representations of an evolving network?

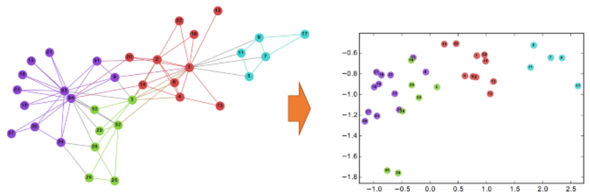
EvoNRL

time efficient

accurate

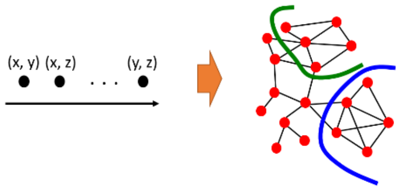
generic method

Current Research focus



A. Network Representation Learning

B. Trajectory Network Mining



C. Streaming & Dynamic Graphs

D. Social Media Mining & Analysis

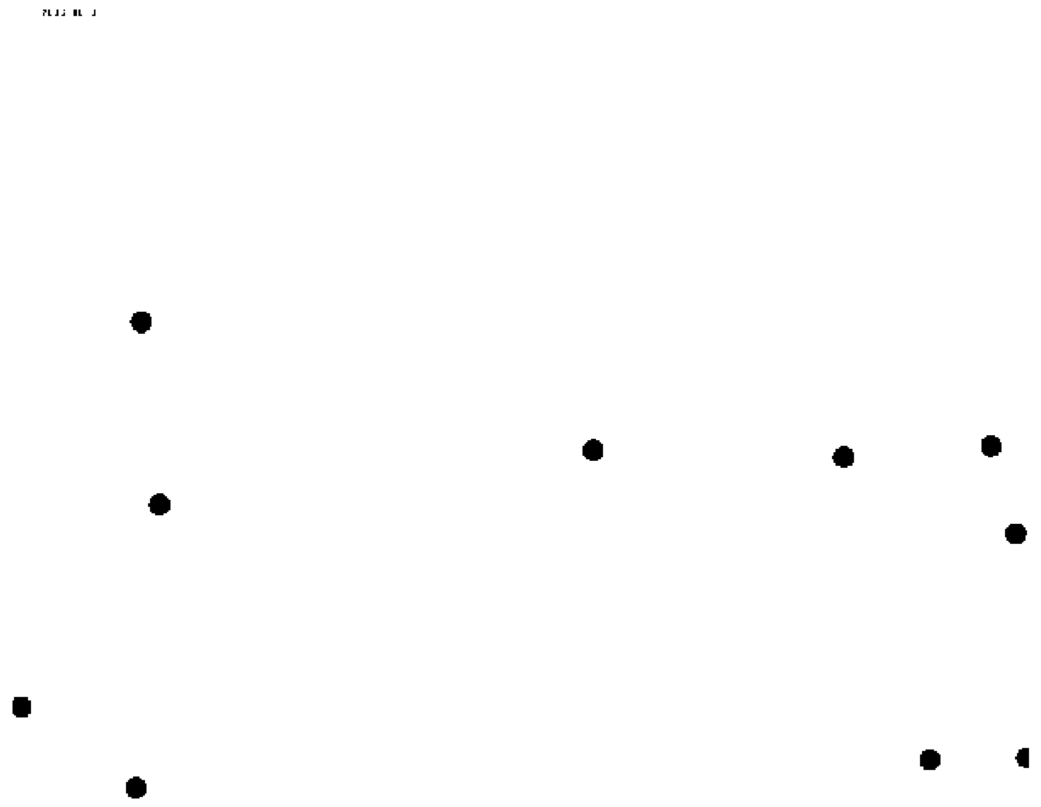
E. City Science / Urban Informatics / IoT

F. Natural Language Processing

Node Importance in Trajectory Networks

Joint work with Tilemachos Pechlivanoglou

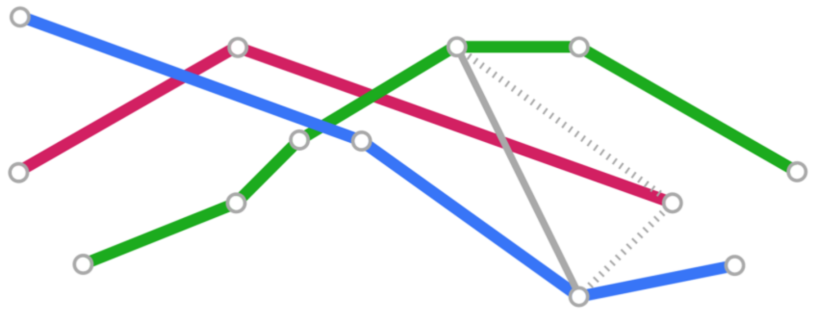
Trajectories of moving objects



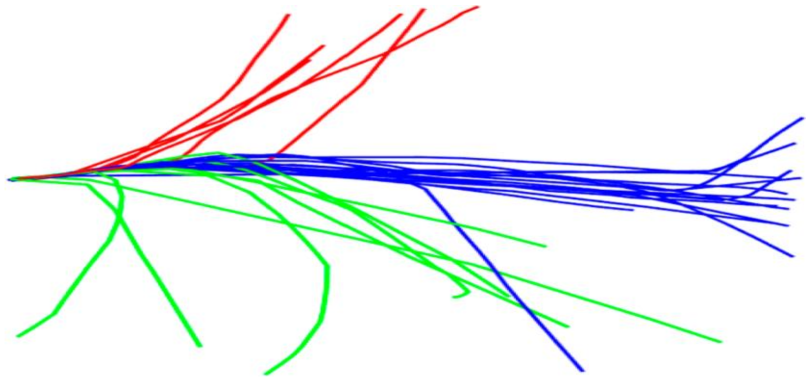
every moving object, forms a **trajectory** – in **2D** it is a sequence of (**x**, **y**, **t**)

there are trajectories of moving **cars**, **people**, **birds**, ...

Trajectory data mining



trajectory similarity

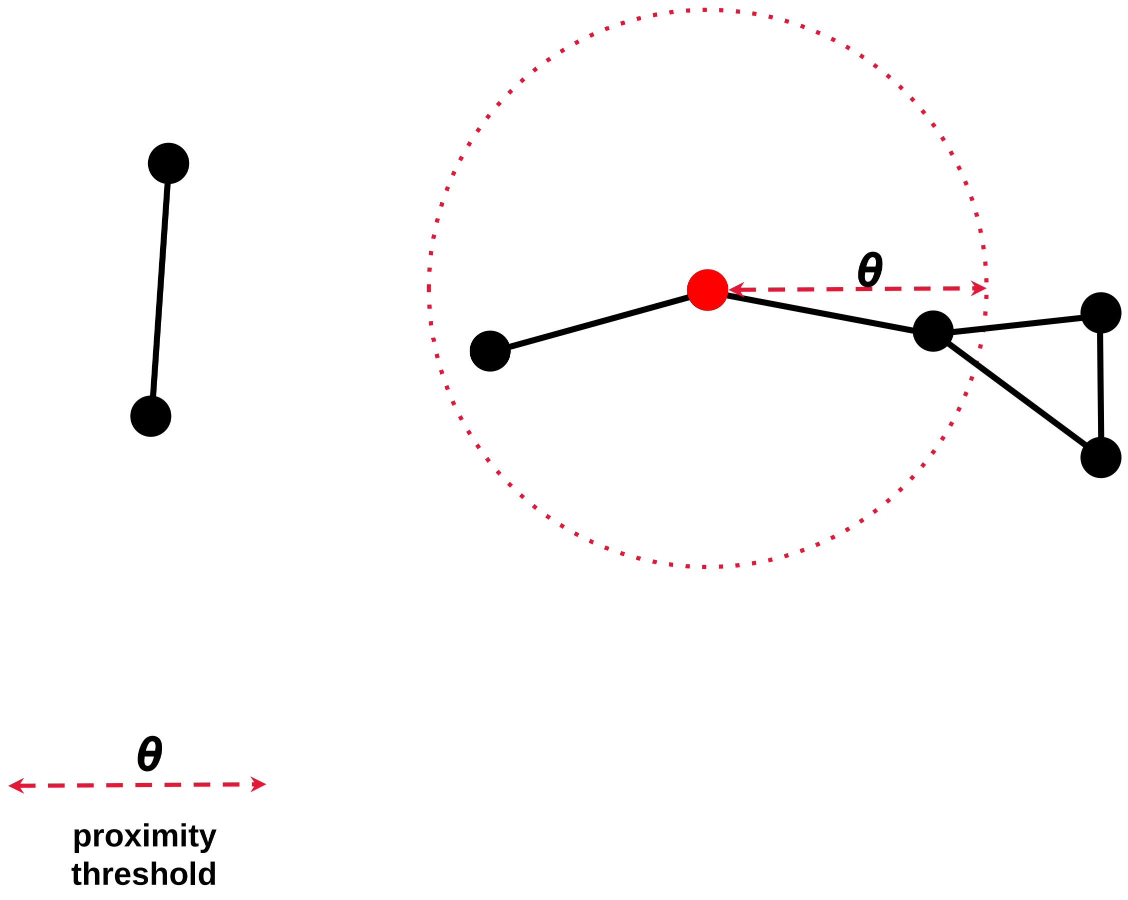


trajectory clustering

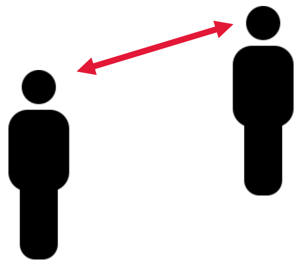
trajectory anomaly detection
trajectory pattern mining
trajectory classification
...more

we care about network analysis of moving objects

Proximity networks



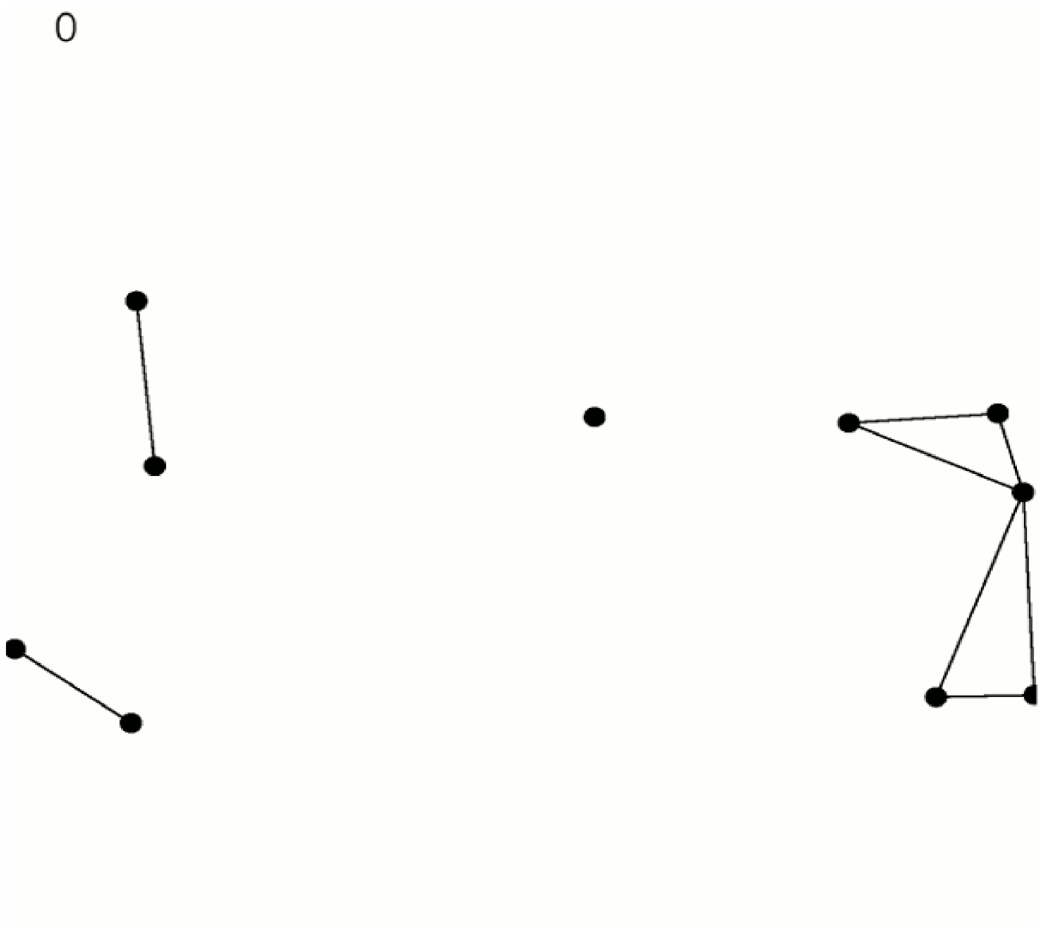
Distance can represent



line of sight

wifi/bluetooth signal range

Trajectory networks



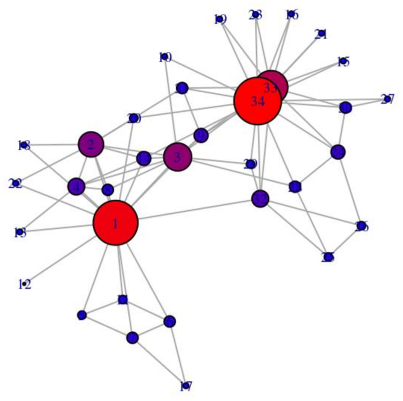
The Problem

Input: logs of trajectories (**x**, **y**, **t**) in time period [0, T]

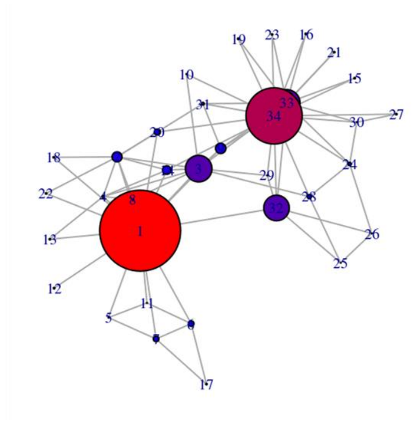
Output: node importance metrics

Node Importance

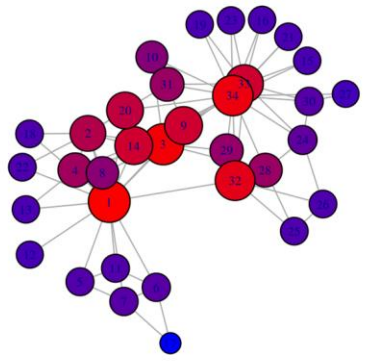
Node importance in static networks



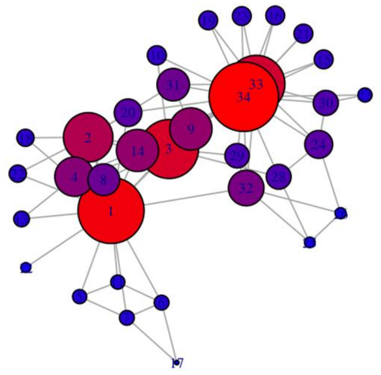
Degree centrality



Betweenness centrality

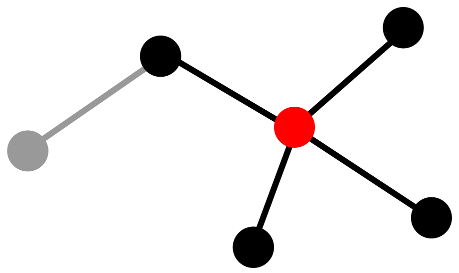


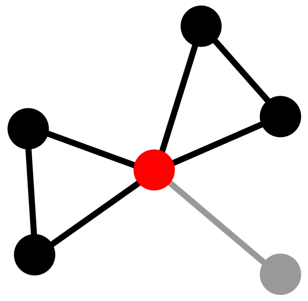
Closeness centrality

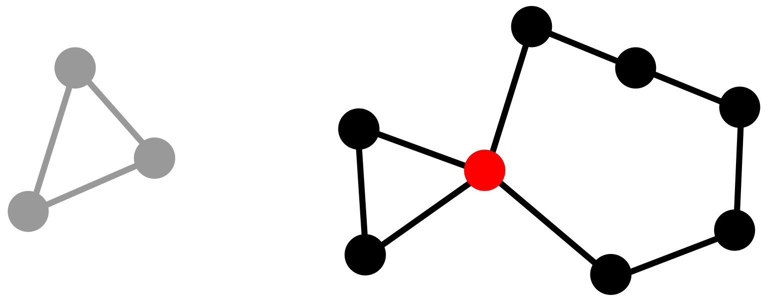


Eigenvector centrality

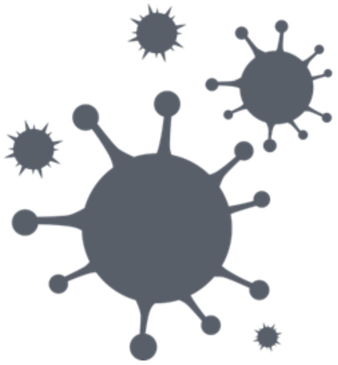
Node importance in TNs


node degree **over time**


triangles **over time**

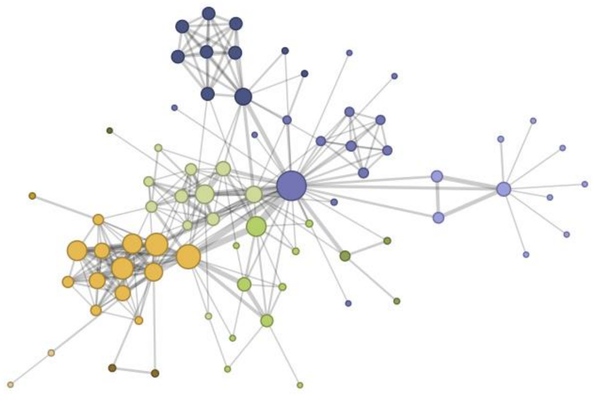

connected components **over time**
(connectedness)

Applications



infection spreading

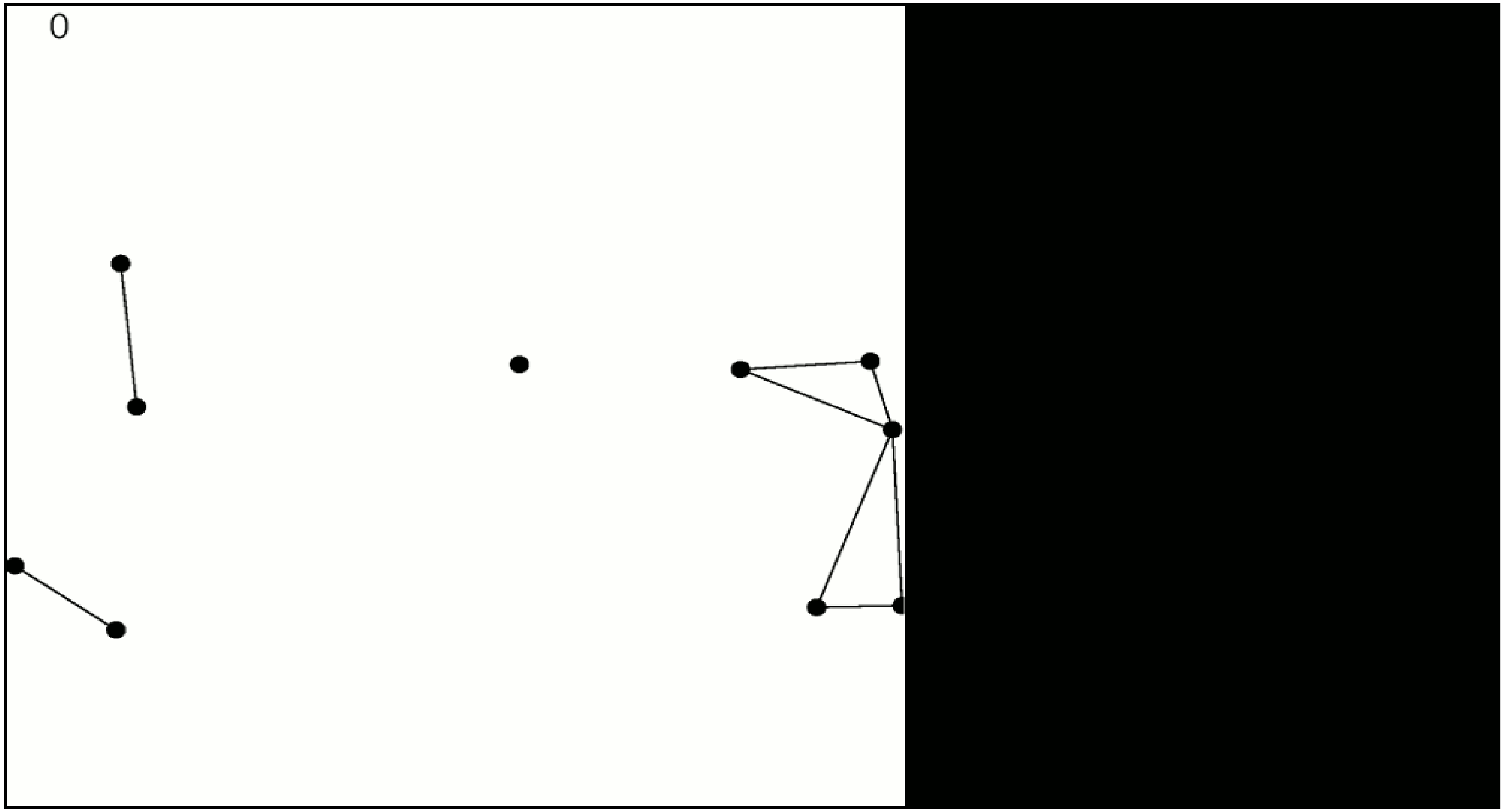
security in autonomous vehicles



rich dynamic network analytics

Evaluation of Node Importance in Trajectory Networks

Naive approach



For every discrete time unit t:

1. obtain static snapshot of the proximity network
2. run static node importance algorithms on snapshot

Aggregate results at the end

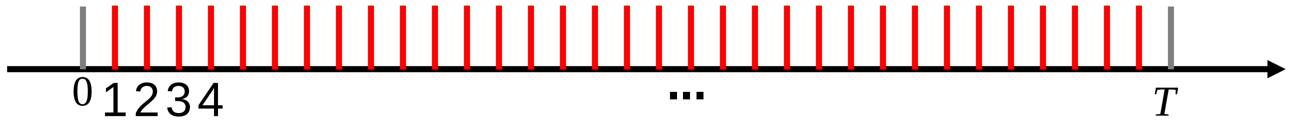
Streaming approach

Similar to naive, but:

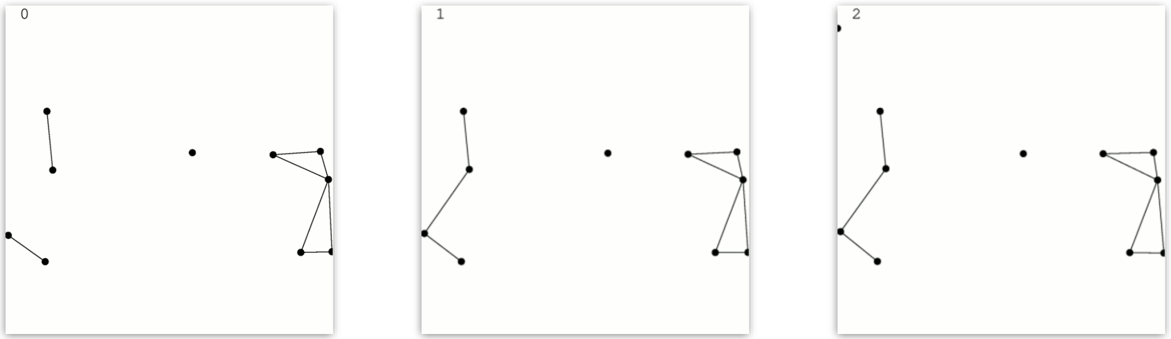
- **no final aggregation**
- results calculated **incrementally** at every step

Still **every time** unit

Every discrete time unit



time

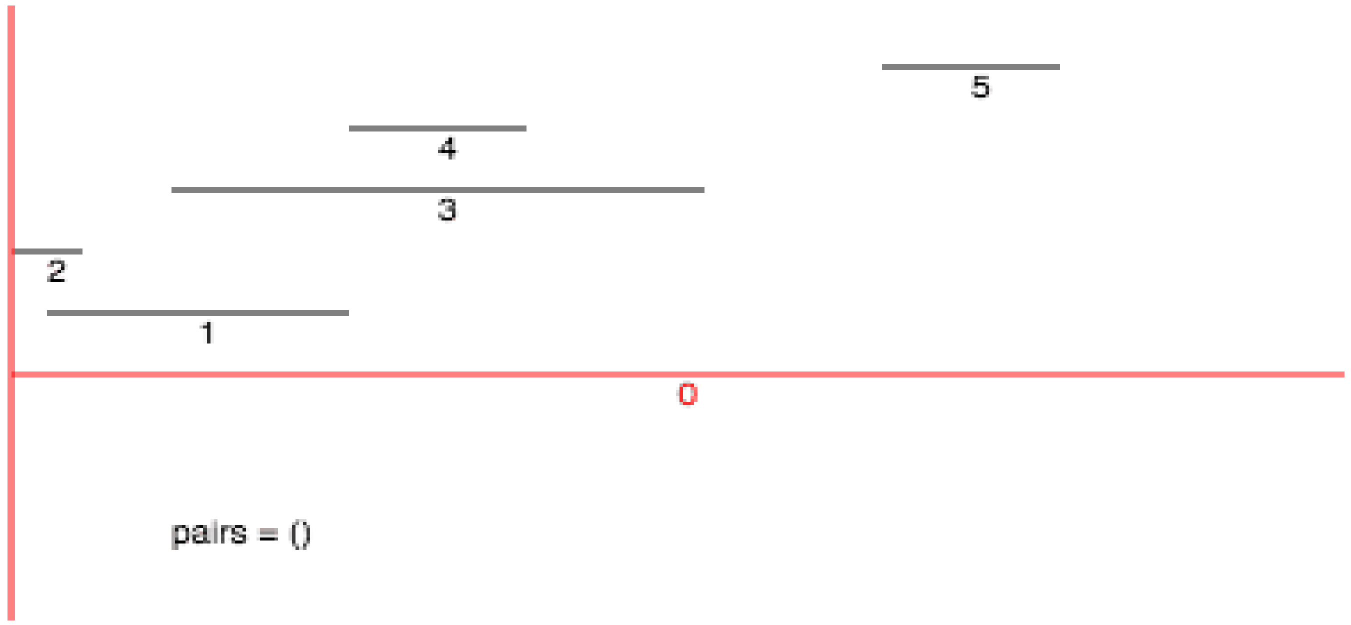


...

Sweep Line Over Trajectories (SLOT)

Sweep line algorithm

A computational geometry algorithm that given line segments computes line segment overlaps



Efficient one pass algorithm that only processes line segments at the beginning and ending points

SLOT: Sweep Line Over Trajectories

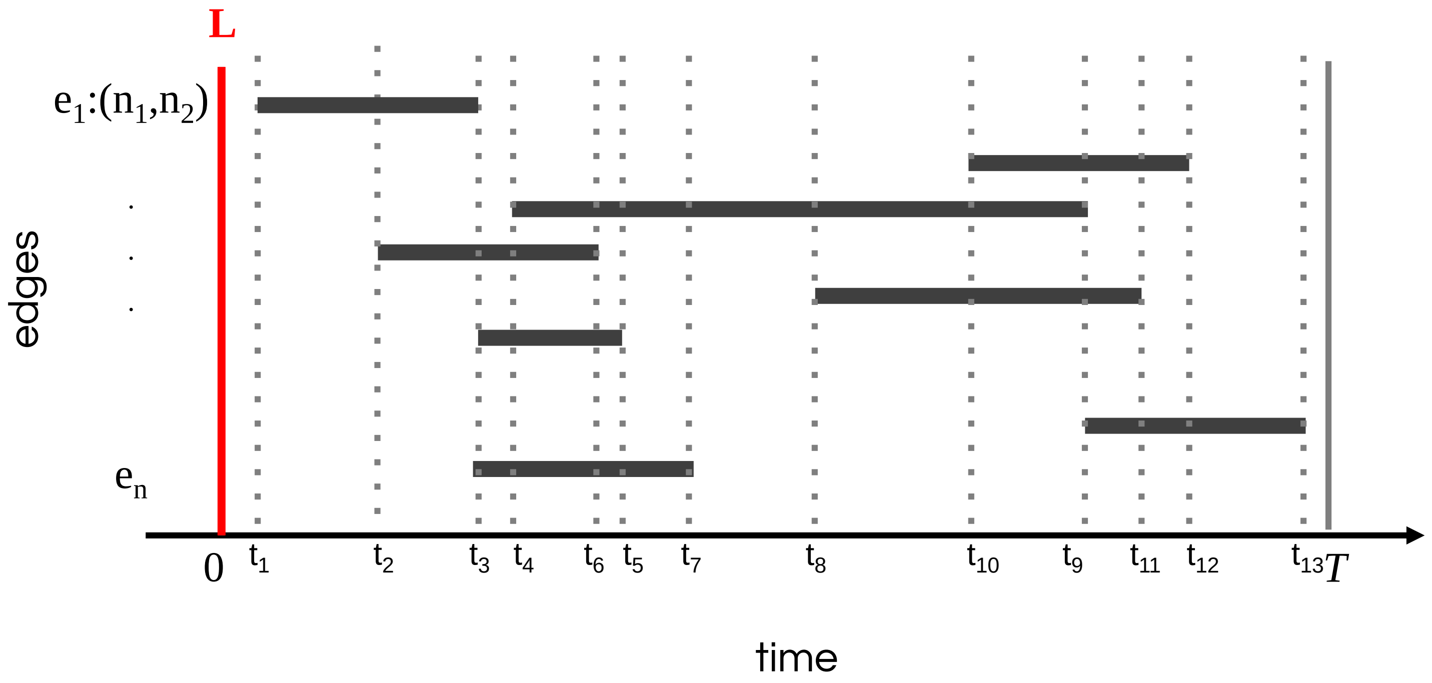
(algorithm sketch)

represent TN **edges** as **time intervals**

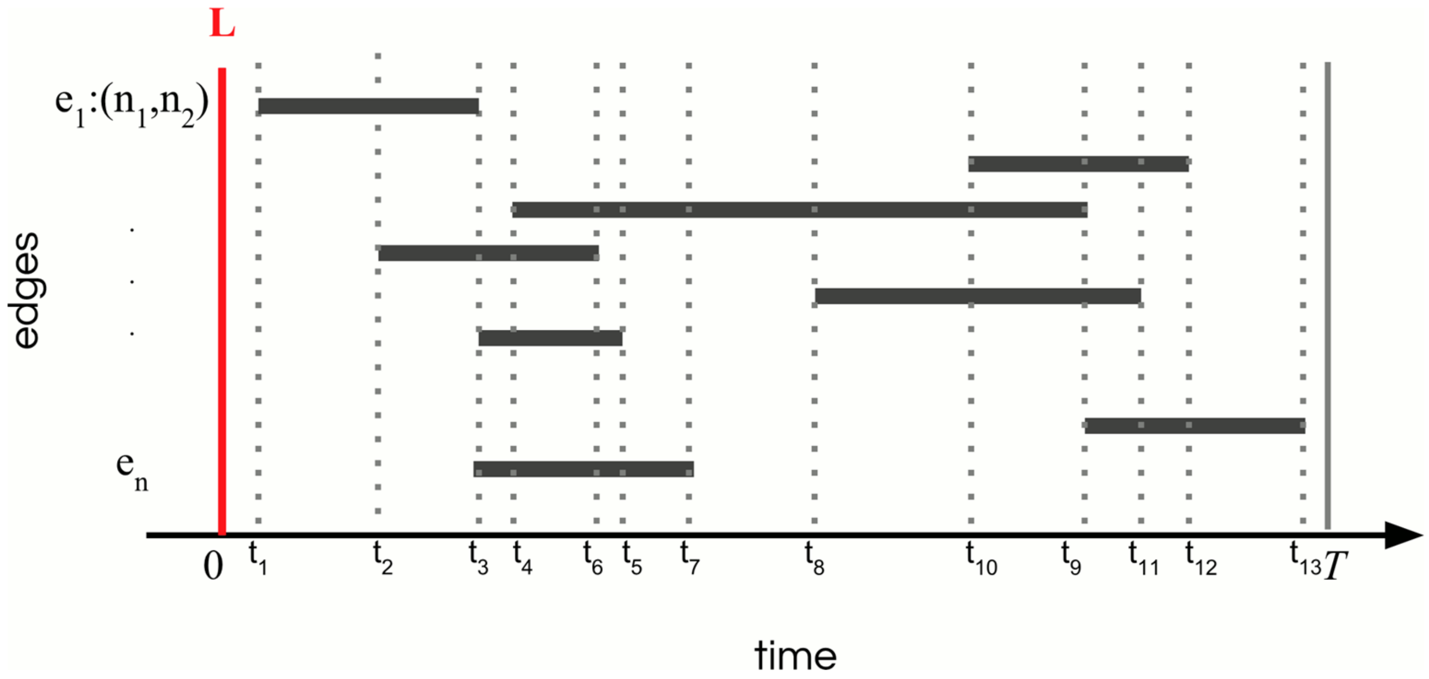
apply **variation** of sweep line algorithm

simultaneously compute *node degree, triangle membership, connected components* in **one pass**

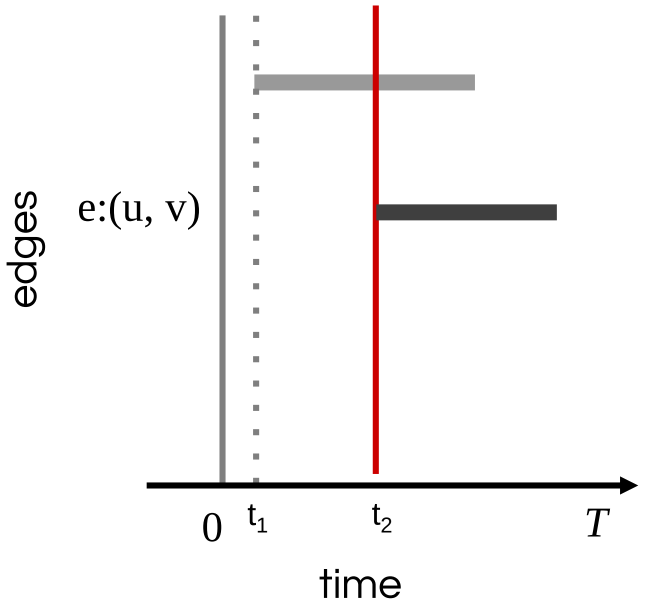
Represent edges as time intervals

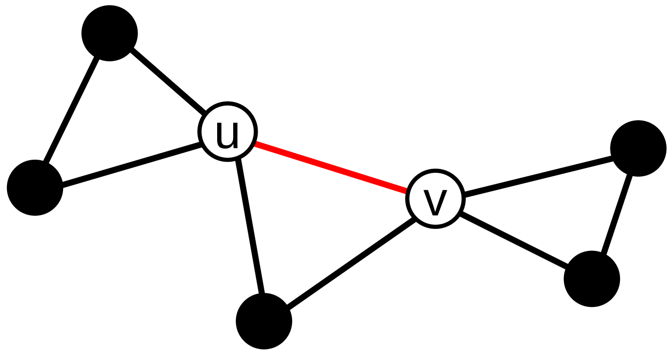


SLOT: Sweep Line Over Trajectories



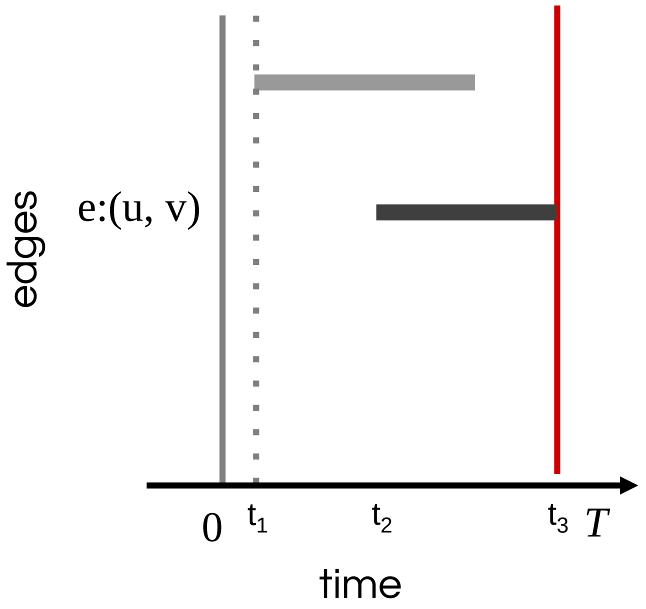
At every edge start

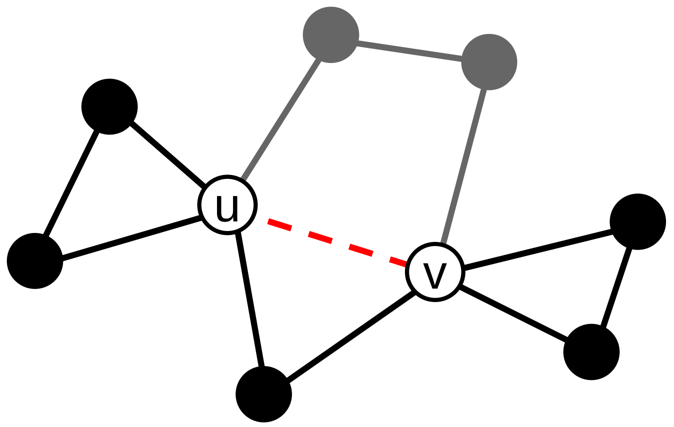




- node degree
  - nodes u, v now connected
  - increment u, v node degrees
- triangle membership
  - did a triangle just form?
  - look for u, v common neighbors
  - increment triangle (u, v, common)
- connected components
  - did two previously disconnected components connect?
  - compare old components of u, v
  - if no overlap, merge them

At every edge stop





- node degree
  - nodes u, v now disconnected
  - decrement u, v degree
- triangle membership
  - did a triangle just break?
  - look for u, v common neighbors
  - decrement triangle (u, v, common)
- connected components
  - did a conn. compon. separate?
  - BFS to see if u, v still connected
  - if not, split component to two

SLOT: At the end of the algorithm …

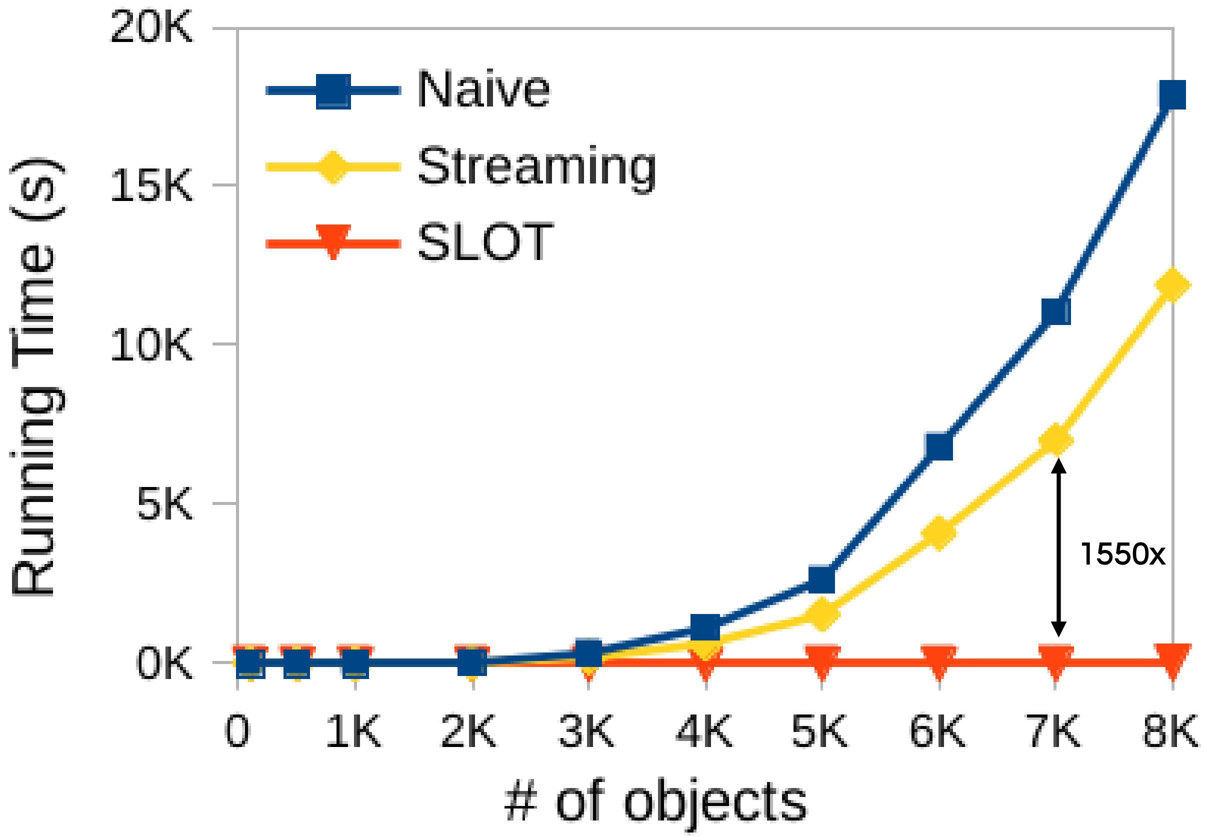
Rich Analytics

- **node degrees:** start/end time, duration
- **triangles:** start/end time, duration
- **connected components:** start/end time, duration

Exact results (not approximations)

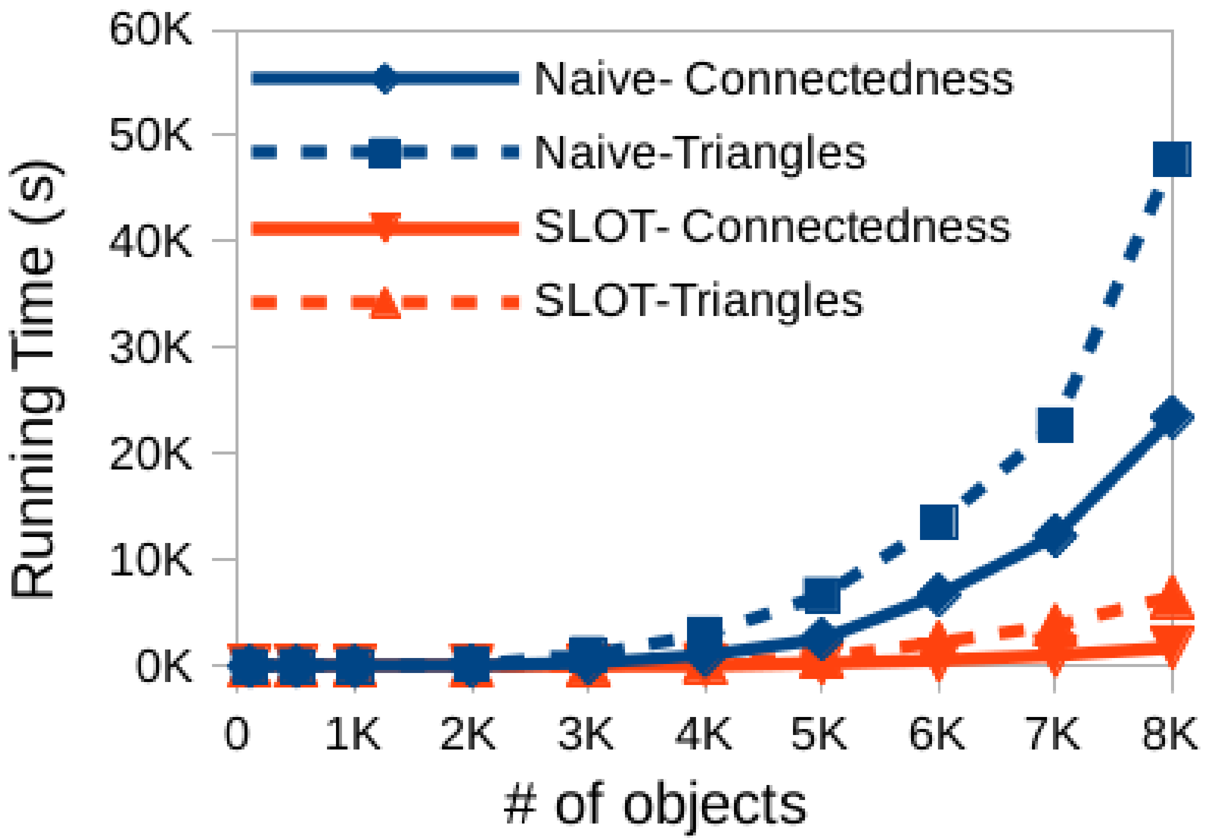
Evaluation of SLOT

Node degree

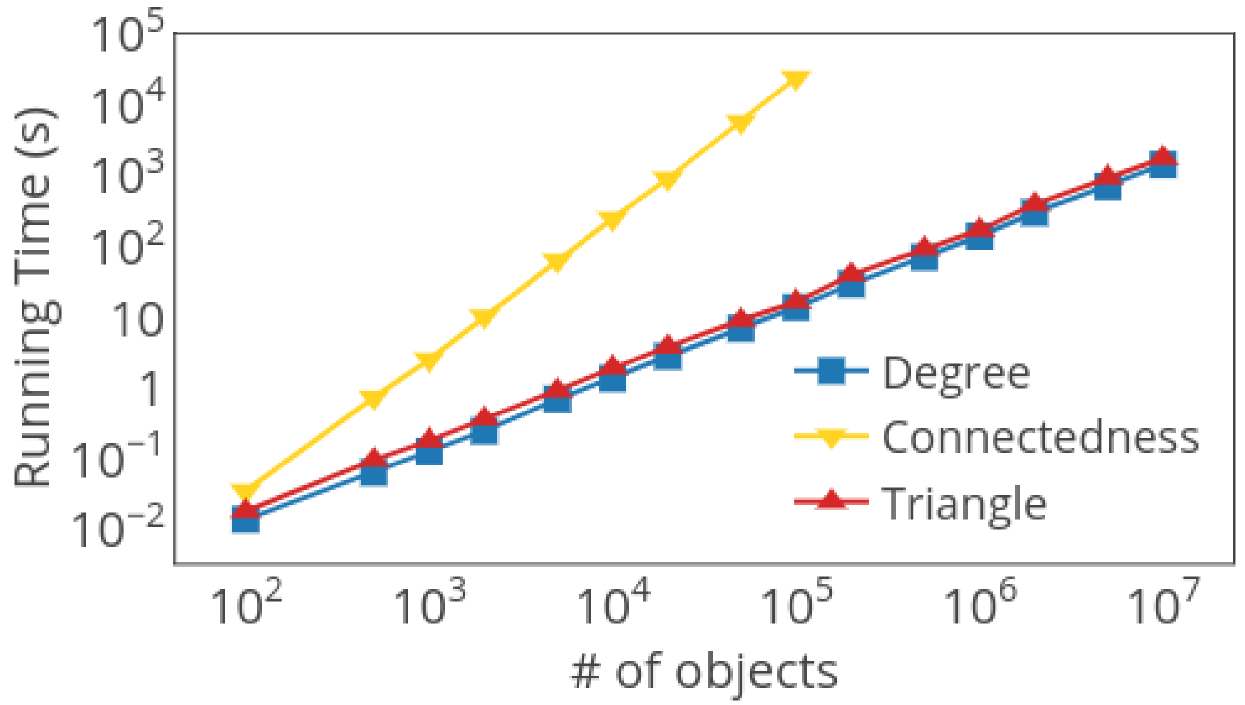


20K
15K
10K
5K
0K
Running Time (s)
Naive
Streaming
SLOT
1550x
0
1K
2K
3K
4K
5K
6K
7K
8K
# of objects

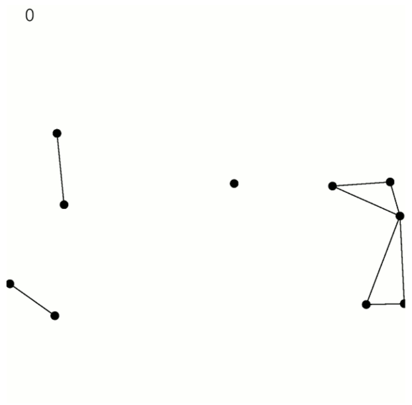
Triangle membership / connected components



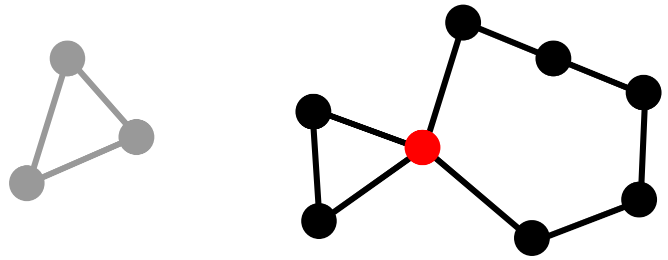
SLOT Scalability



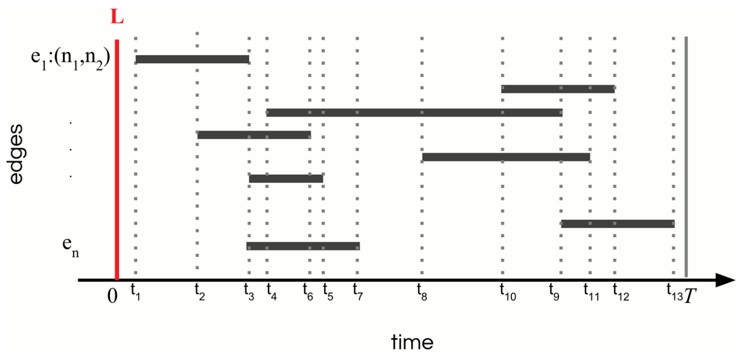
Takeaway



trajectory networks



network importance **over time**

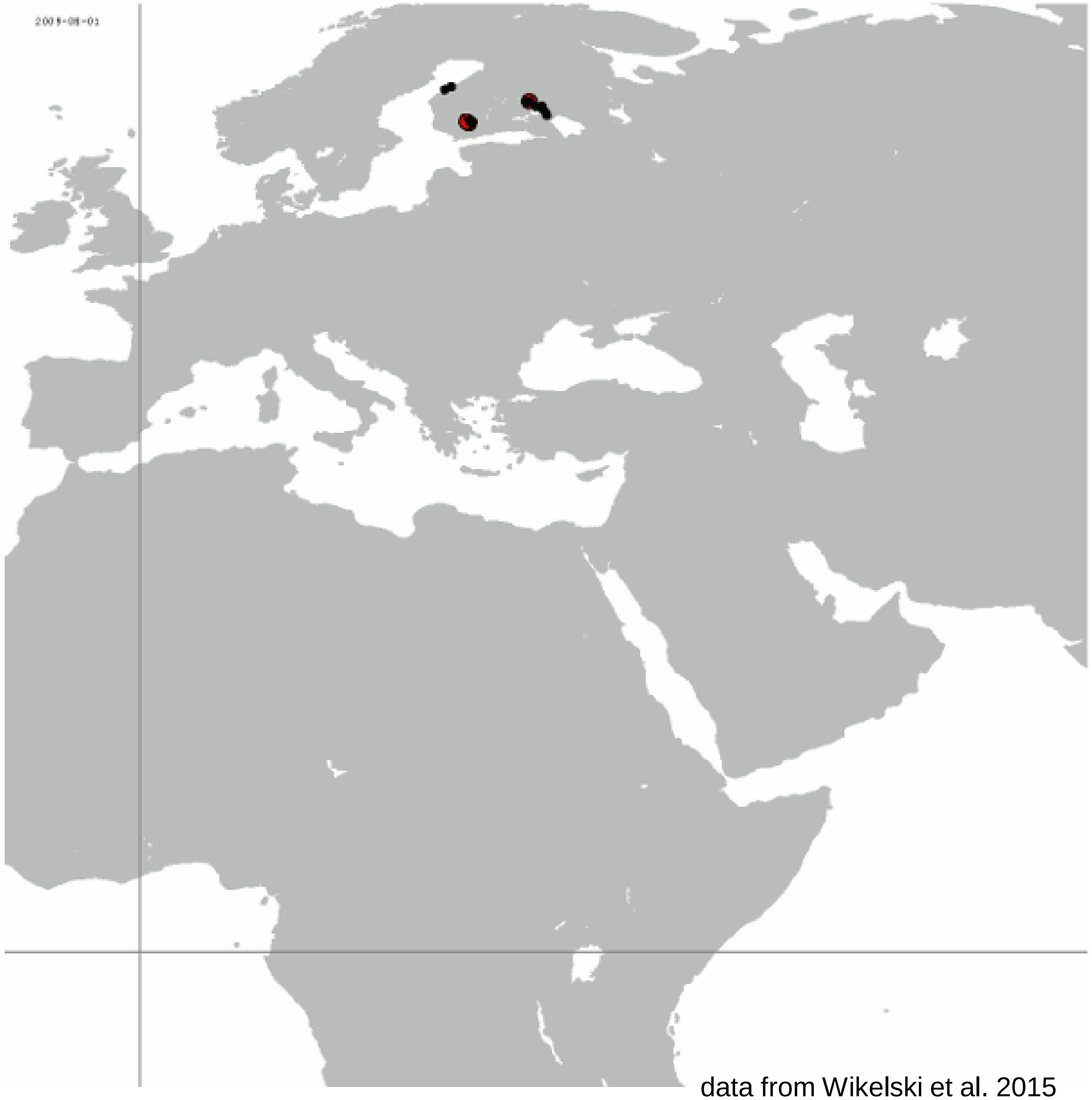


SLOT algorithm

SLOT properties:

- fast
- exact
- scalable

Seagull migration trajectories



data from Wikelski et al. 2015

Credits

Farzaneh Heidari

[Complex Networks 2018] **EvoNRL: Evolving Network Representation Learning Based on Random Walks**. Farzaneh Heidari and Manos Papagelis.

Source code: https://github.com/farzana0/EvoNRL/

Tilemachos Pechlivanoglou

[IEEE Big Data 2018] **Fast and Accurate Mining of Node Importance in Trajectory Networks**. Tilemachos Pechlivanoglou and Manos Papagelis.

Source code: https://github.com/tipech/trajectory-networks

For more info visit: Data Mining Lab @ YorkU

Thank you!

Questions?