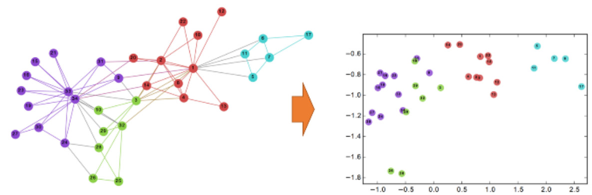
Trajectory Network Mining

Manos Papagelis

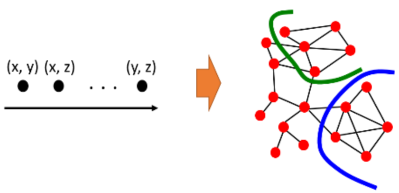
papaggel@eecs.yorku.ca

Current research focus

A. Trajectory Network Mining



B. Network Representation Learning



C. Streaming & Dynamic Graphs

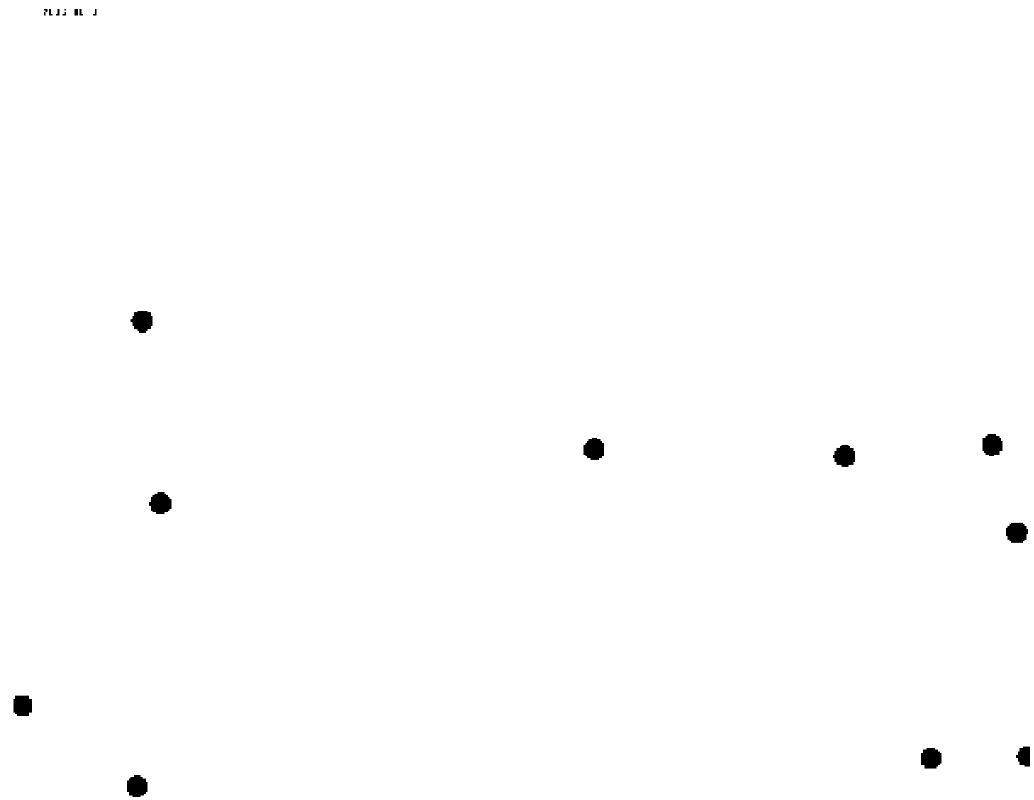
D. Social Media Mining & Analysis

E. City Science / Urban Informatics / IoT

F. Natural Language Processing

Node Importance in Trajectory Networks

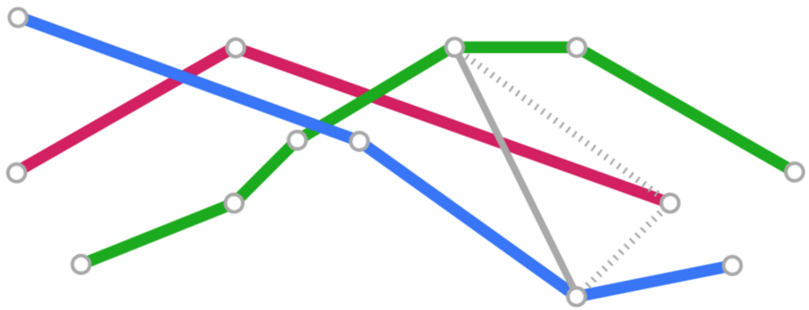
Trajectories of moving objects



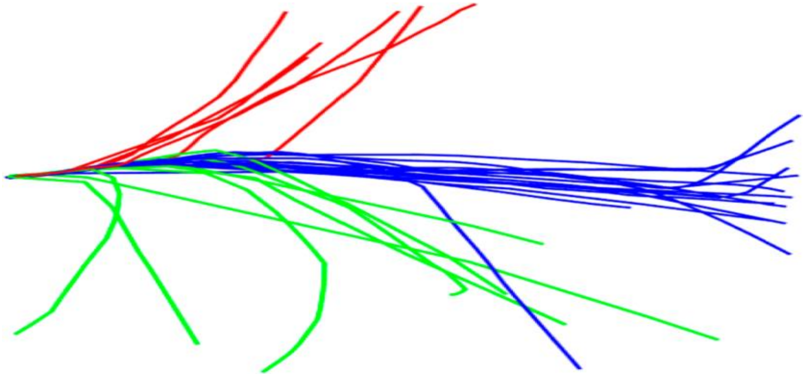
every moving object, forms a **trajectory** – in **2D** it is a sequence of (**x**, **y**, **t**)

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

Trajectory data mining



trajectory similarity

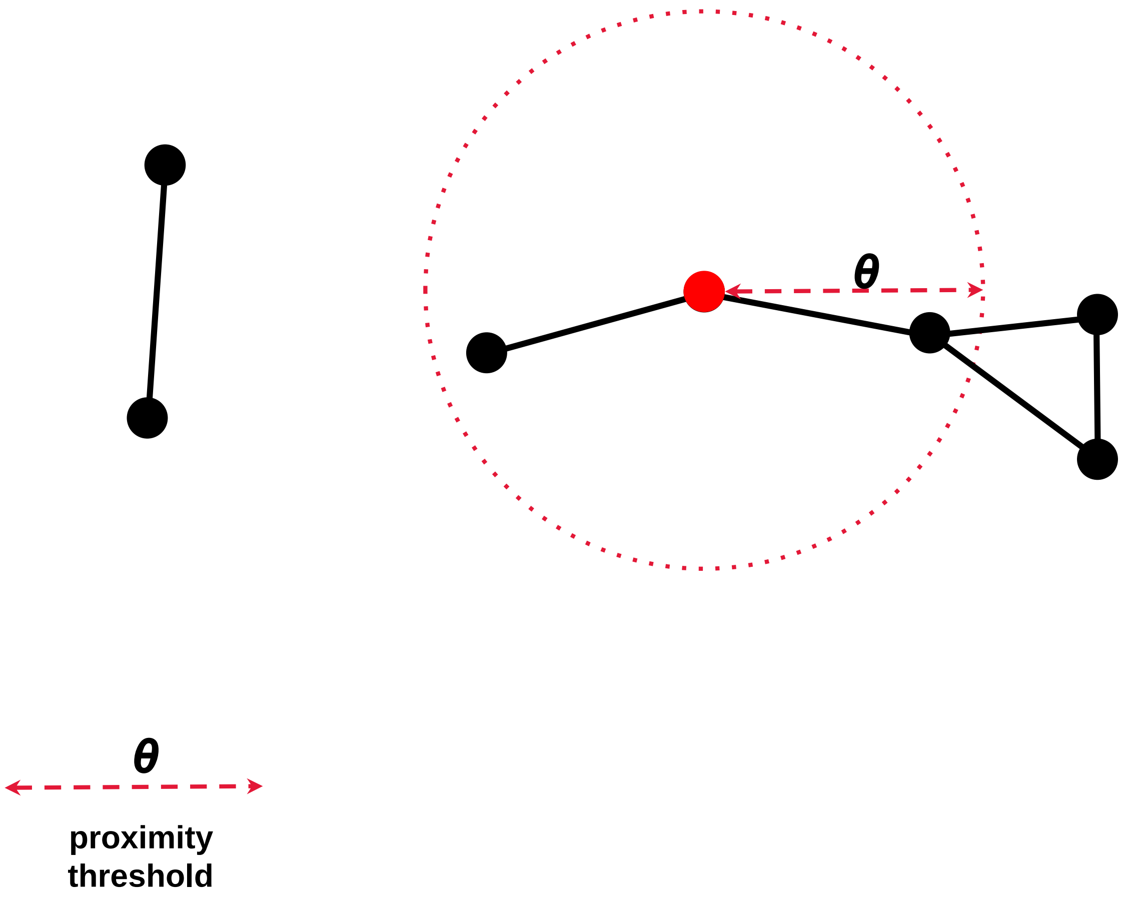


trajectory clustering

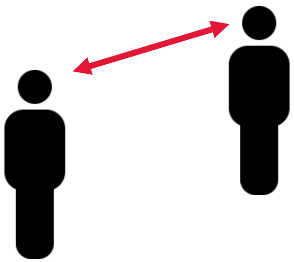
trajectory anomaly detection
trajectory pattern mining
trajectory classification
...more

we care about network analysis of moving objects

Proximity networks



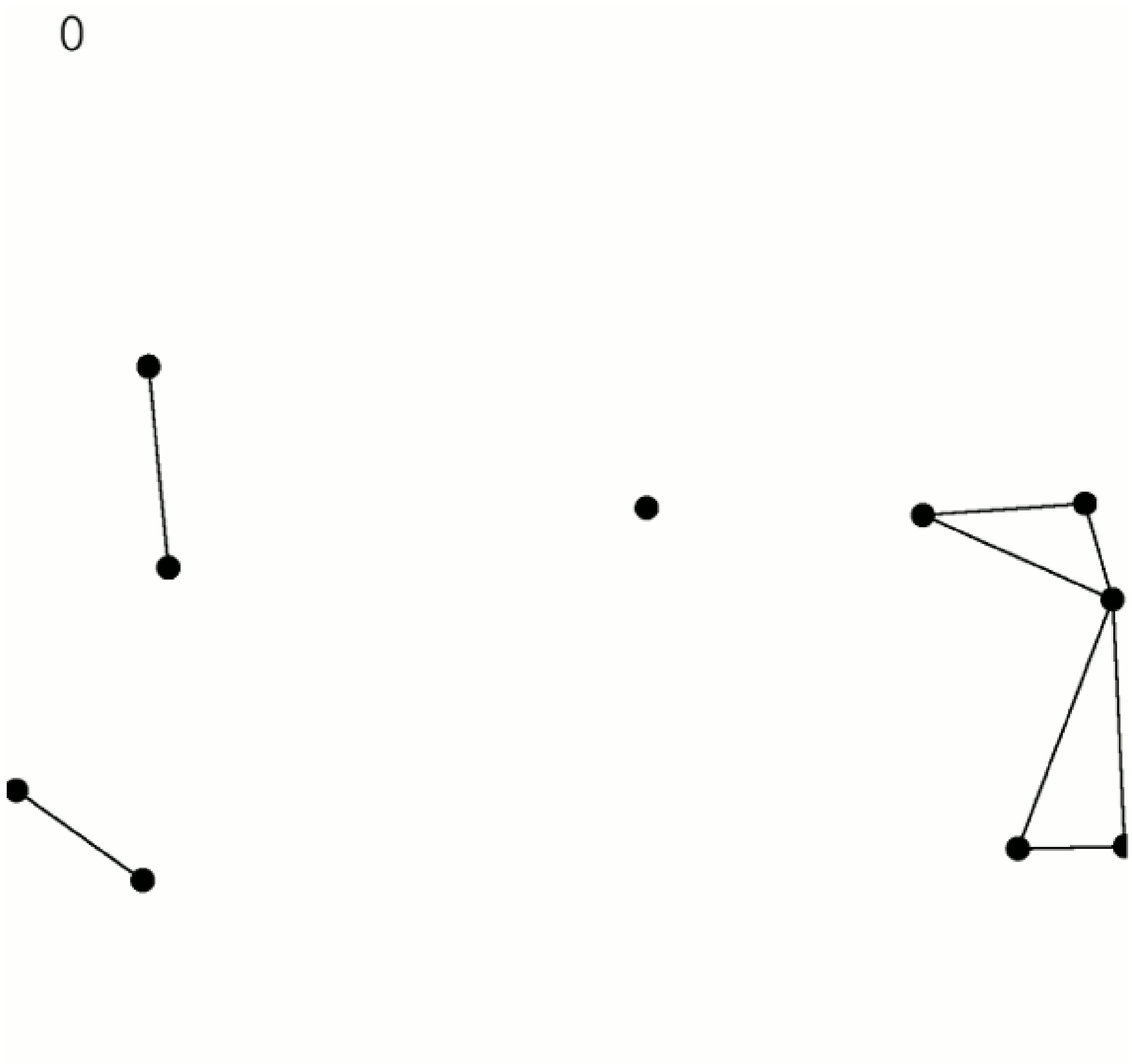
Distance can represent



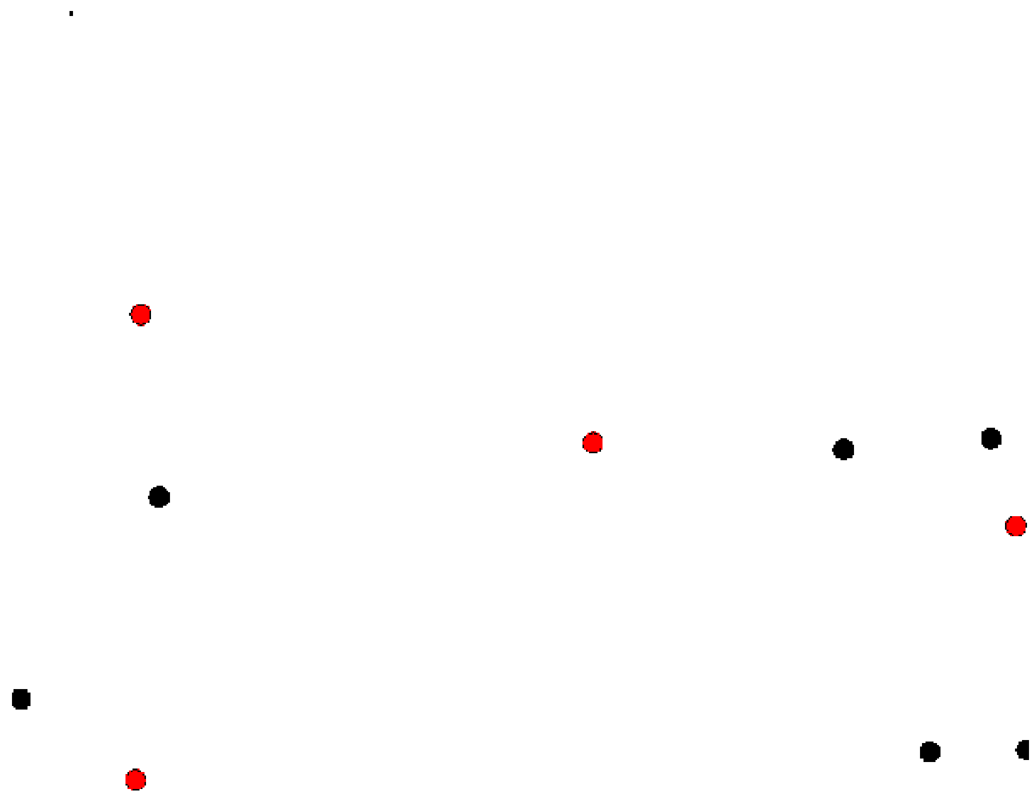
line of sight

wifi/bluetooth signal range

Trajectory networks



The problem

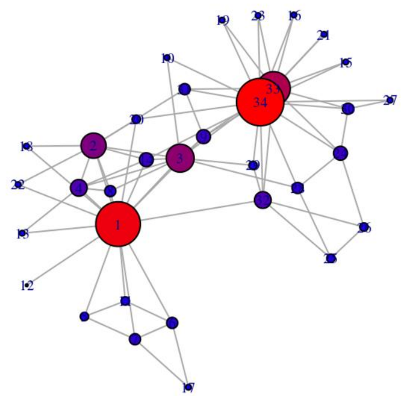


Input: logs of trajectories (**x**, **y**, **t**)

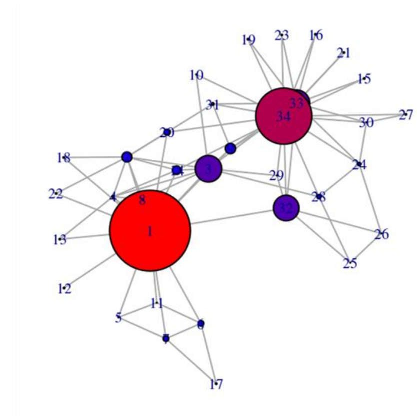
Output: node importance metrics

Node Importance

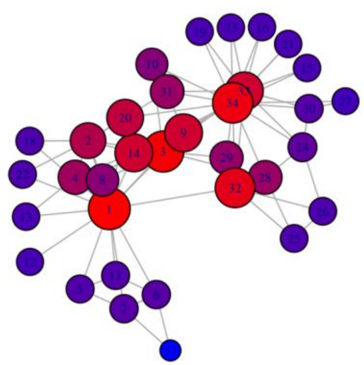
Node importance in static networks



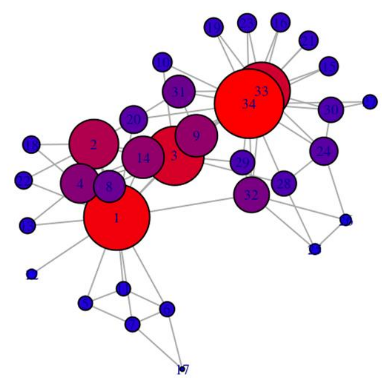
Degree centrality



Betweenness centrality

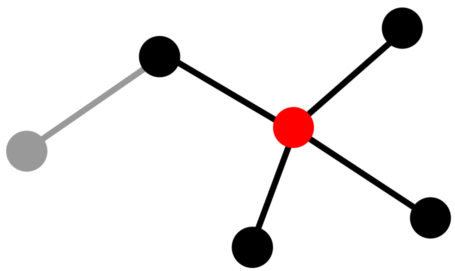


Closeness centrality

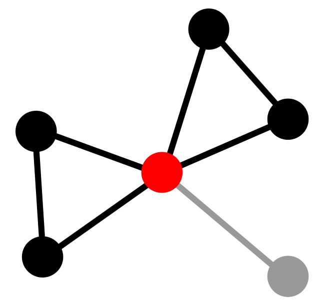


Eigenvector centrality

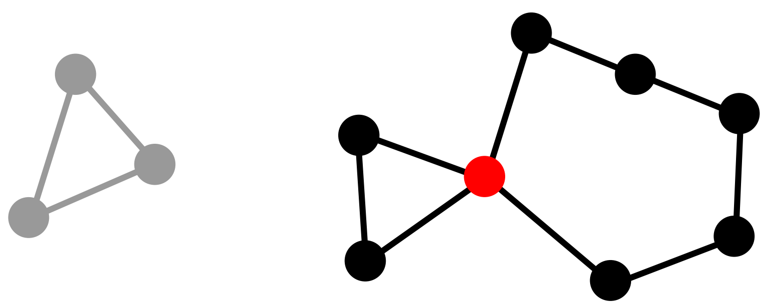
Node importance in TNs



node degree **over time**

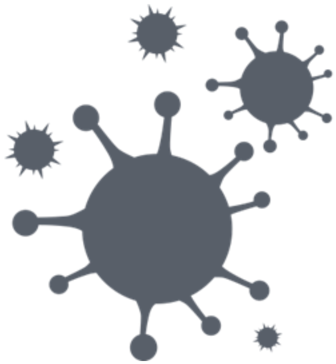


triangles **over time**



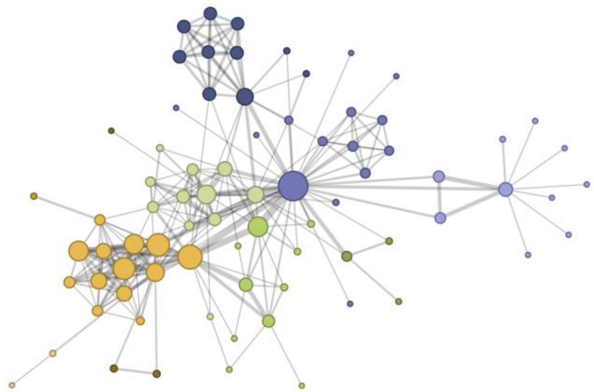
connected components **over time**
(connectedness)

Applications



infection spreading

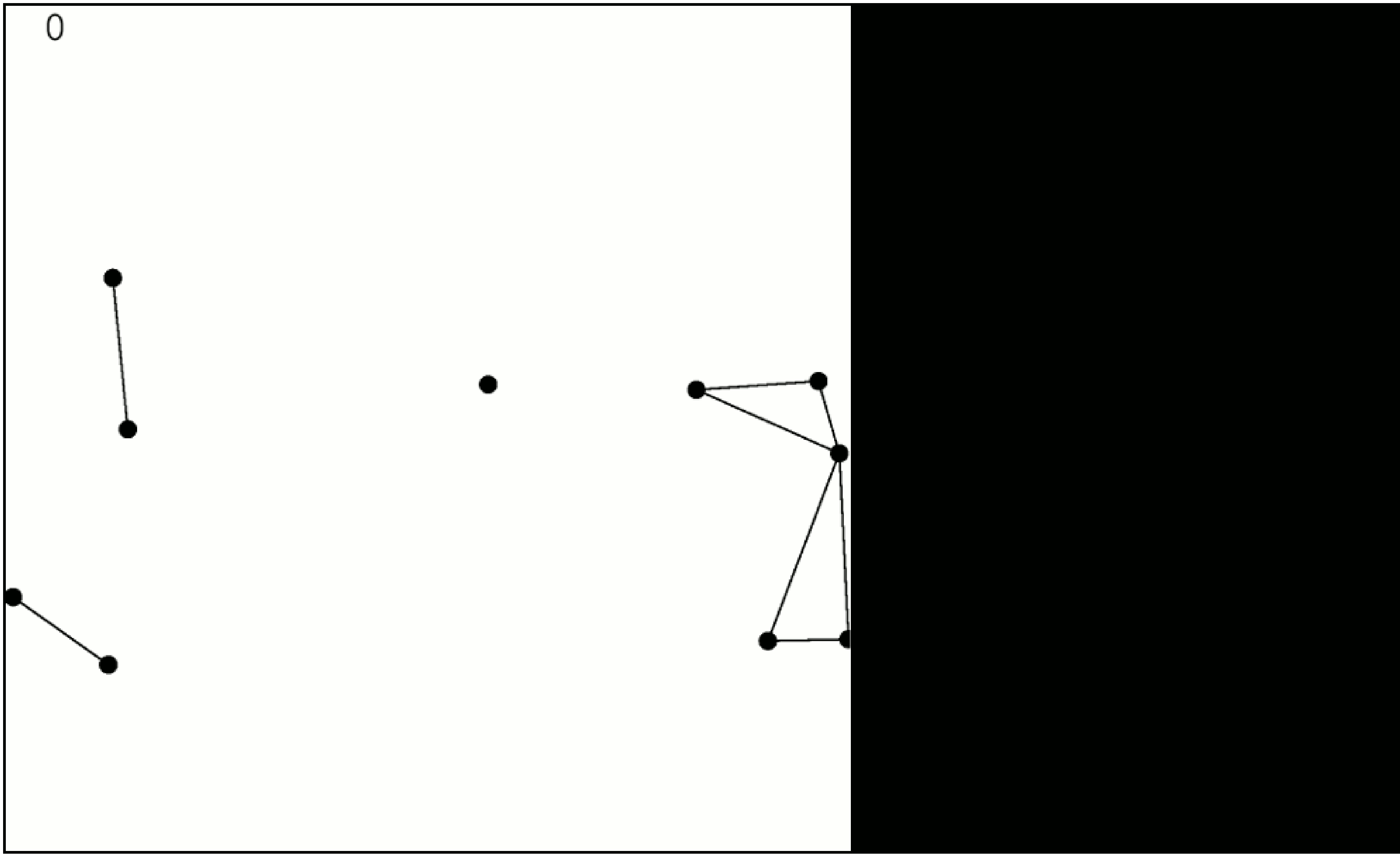
wireless signal security



rich dynamic network analytics

Evaluation of Node Importance in Trajectory Networks

Naive approach



Naive approach

For every discrete time unit:

1. get static snapshot of network
2. run static node importance algorithms on snapshot

Aggregate results at the end

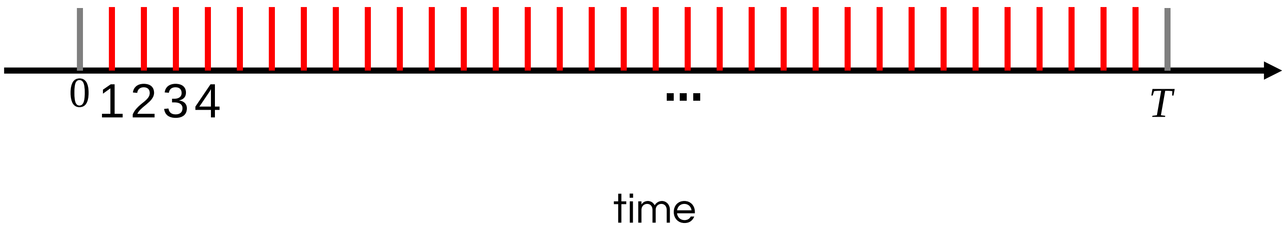
Streaming approach

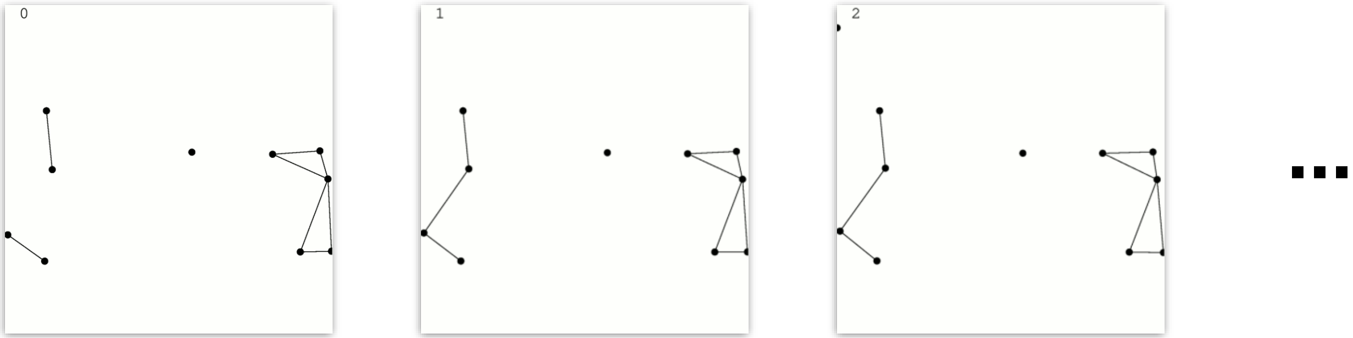
Similar to naive, but:

- no final aggregation
- results calculated incrementally at every step

Still every time unit

Every discrete time unit

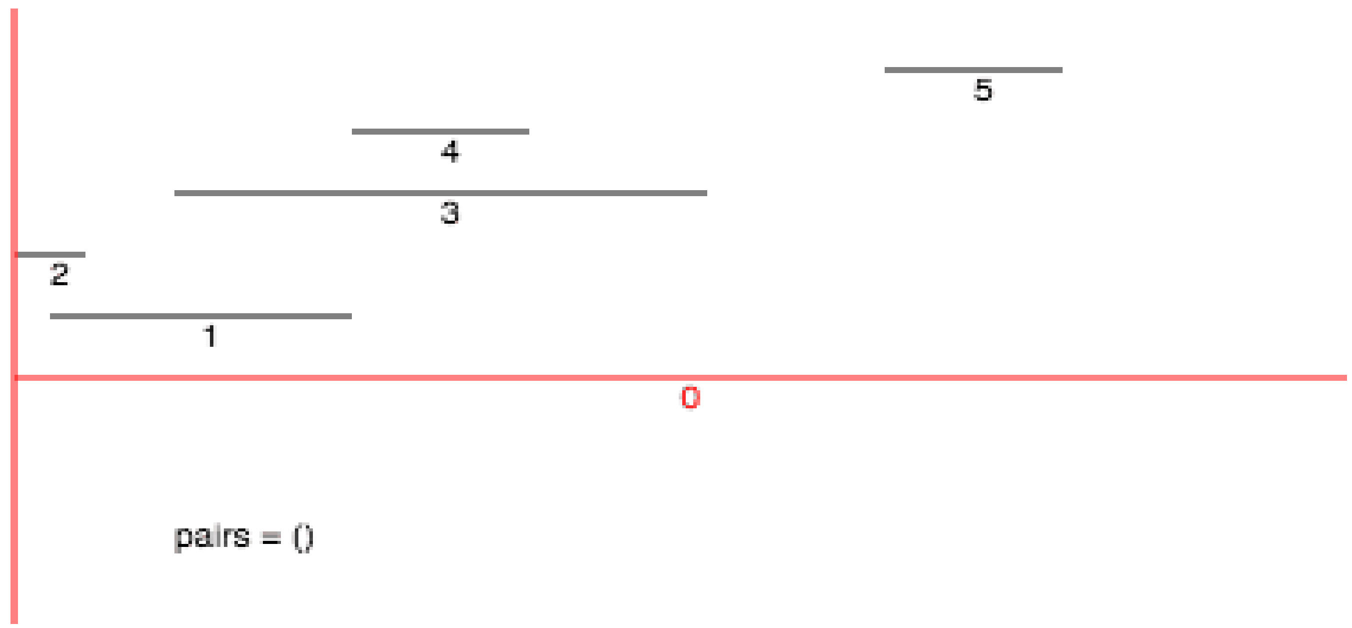




Sweep Line Over Trajectories (SLOT)

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

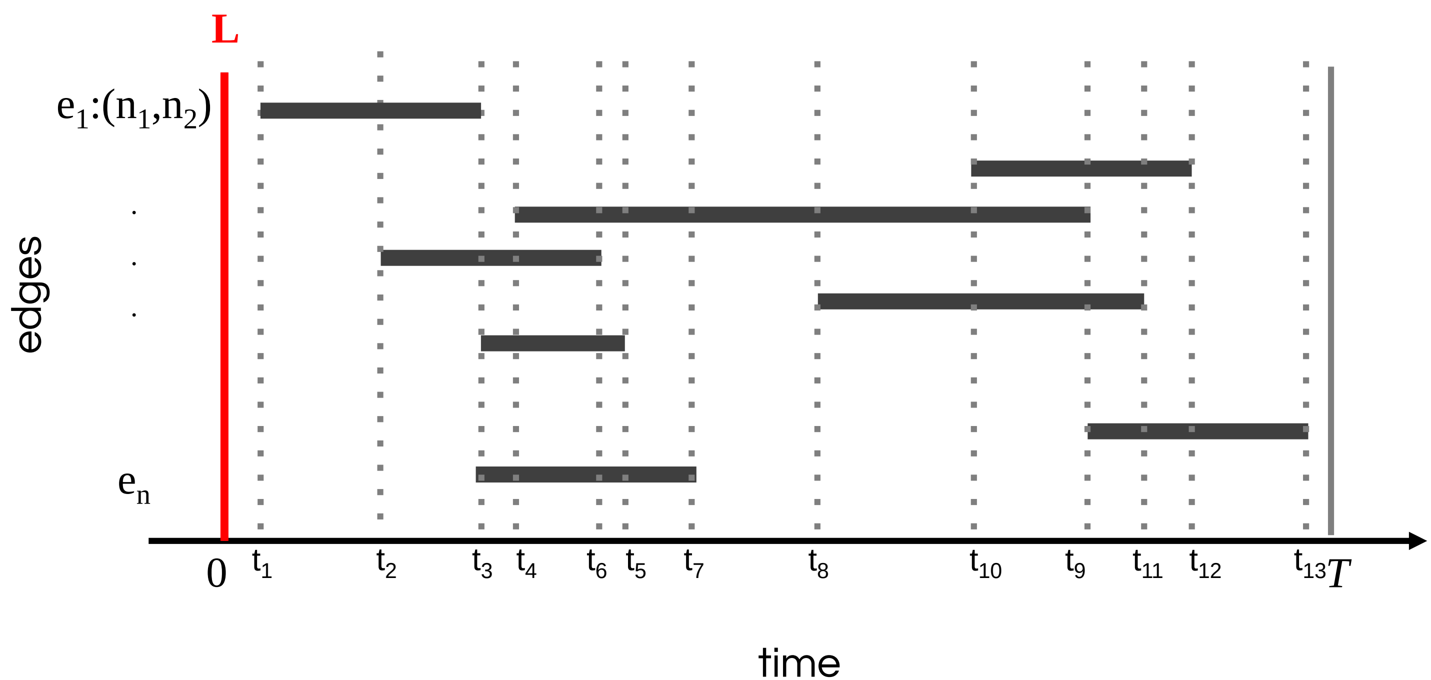
(**algorithm sketch**)

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

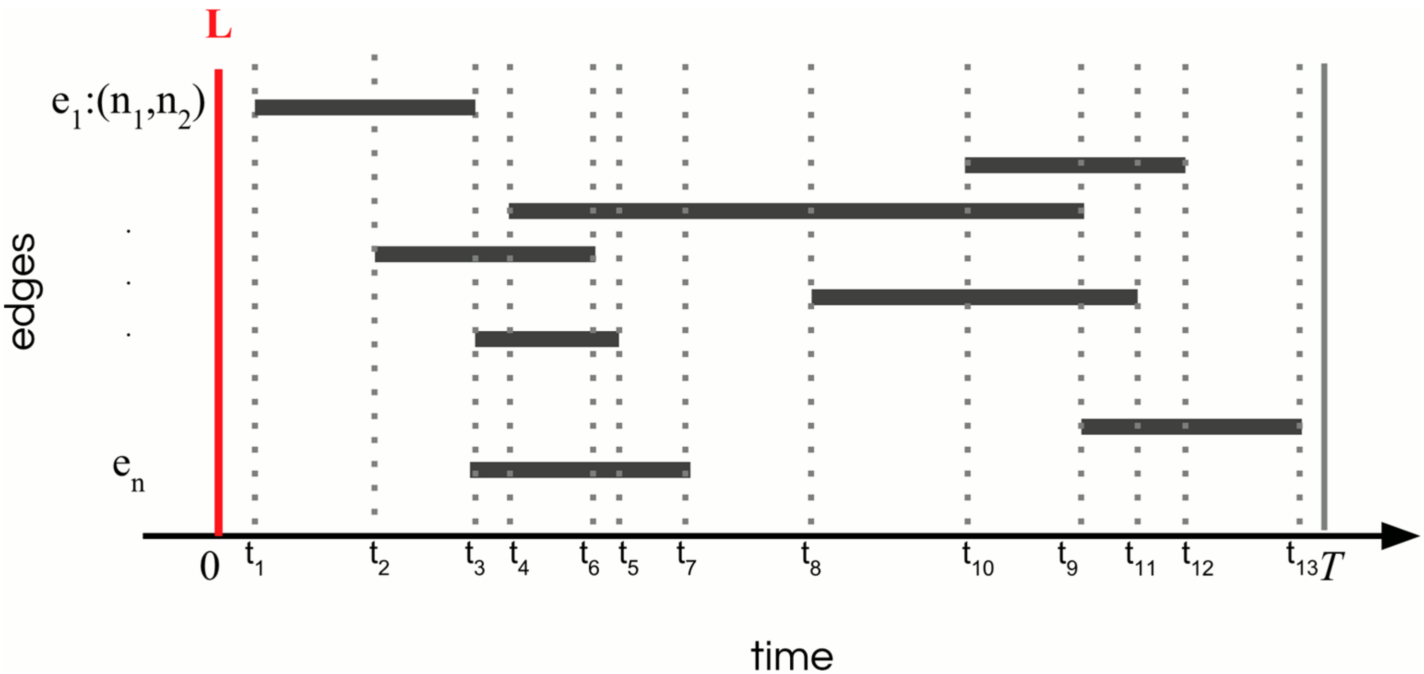
apply **variation** of sweep line algorithm

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

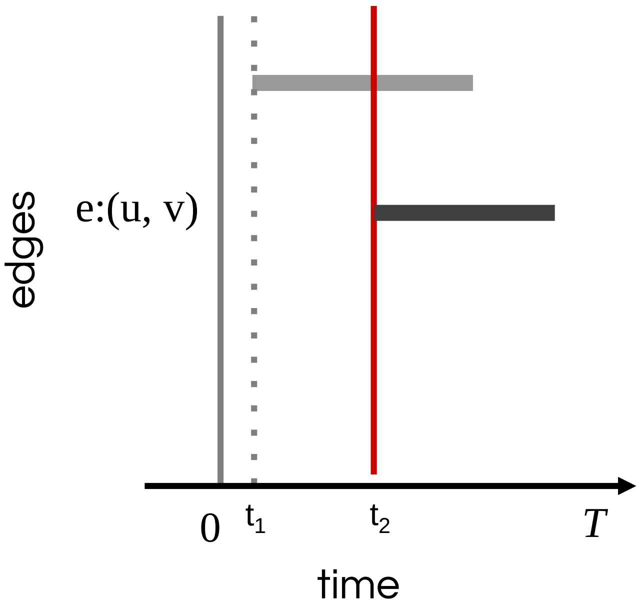
Represent edges as time intervals

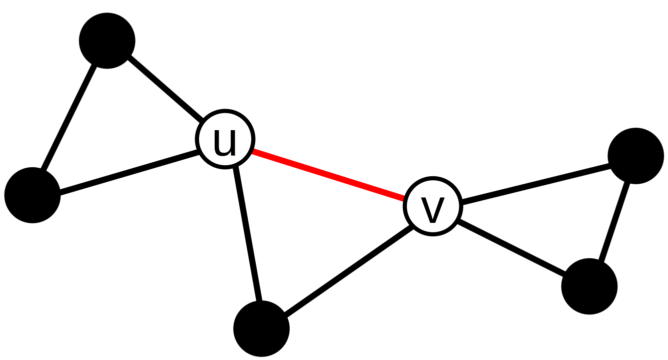


SLOT: Sweep Line Over Trajectories



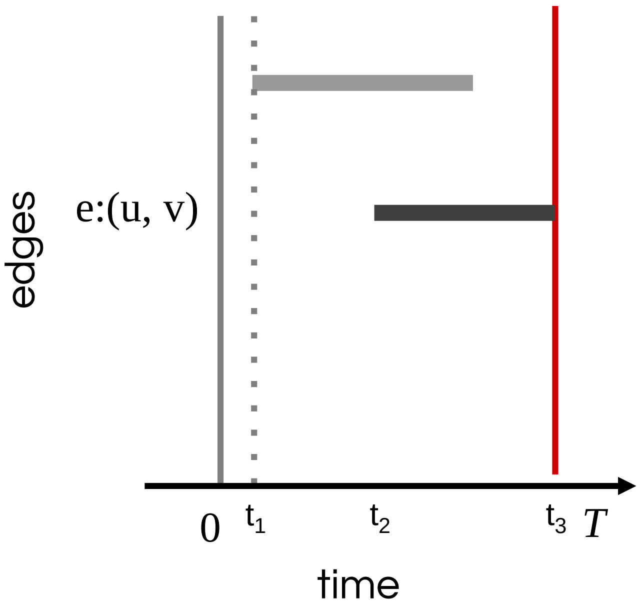
At every edge **start**

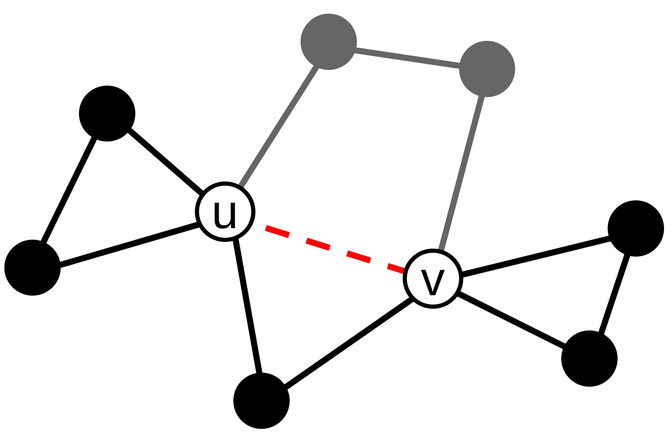




- node degree
  - nodes u, v now connected
  - increment u, v node degrees
- triangle membership
  - did a triangle just form?
  - look for u, v common neighbors
  - increment triangle (u, v, common)
- connected components
  - did two previously unconnected components connect?
  - compare old components 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 component separate?
  - BFS to see if u, v still connected
  - if not, split component to two

SLOT: At the end of the algorithm ...

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

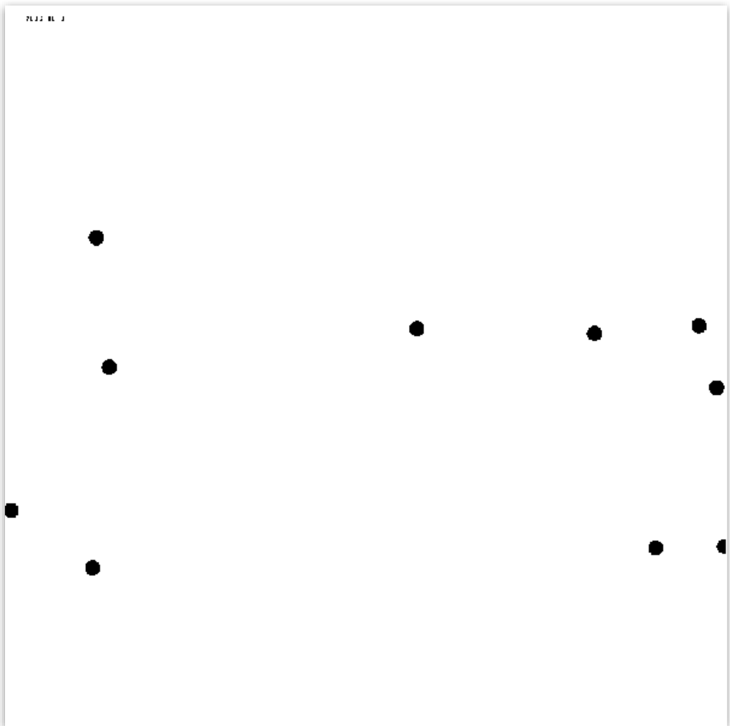
Exact results (not approximations)

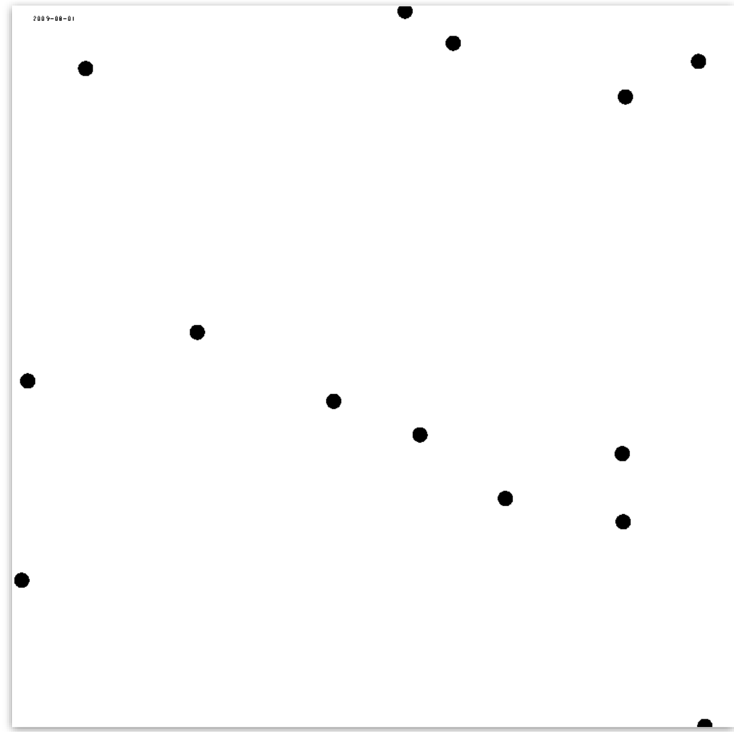
e.g. node degree of u *d(u)* is:

d(u) = 5,	**from**	t=0	**until**	t=10	**duration**=10
d(u) = 6,	**from**	t=10	**until**	t=50	**duration**=40
d(u) = 4,	**from**	t=50	**until**	t=100	**duration**=50

Evaluation of SLOT

Simulating trajectories

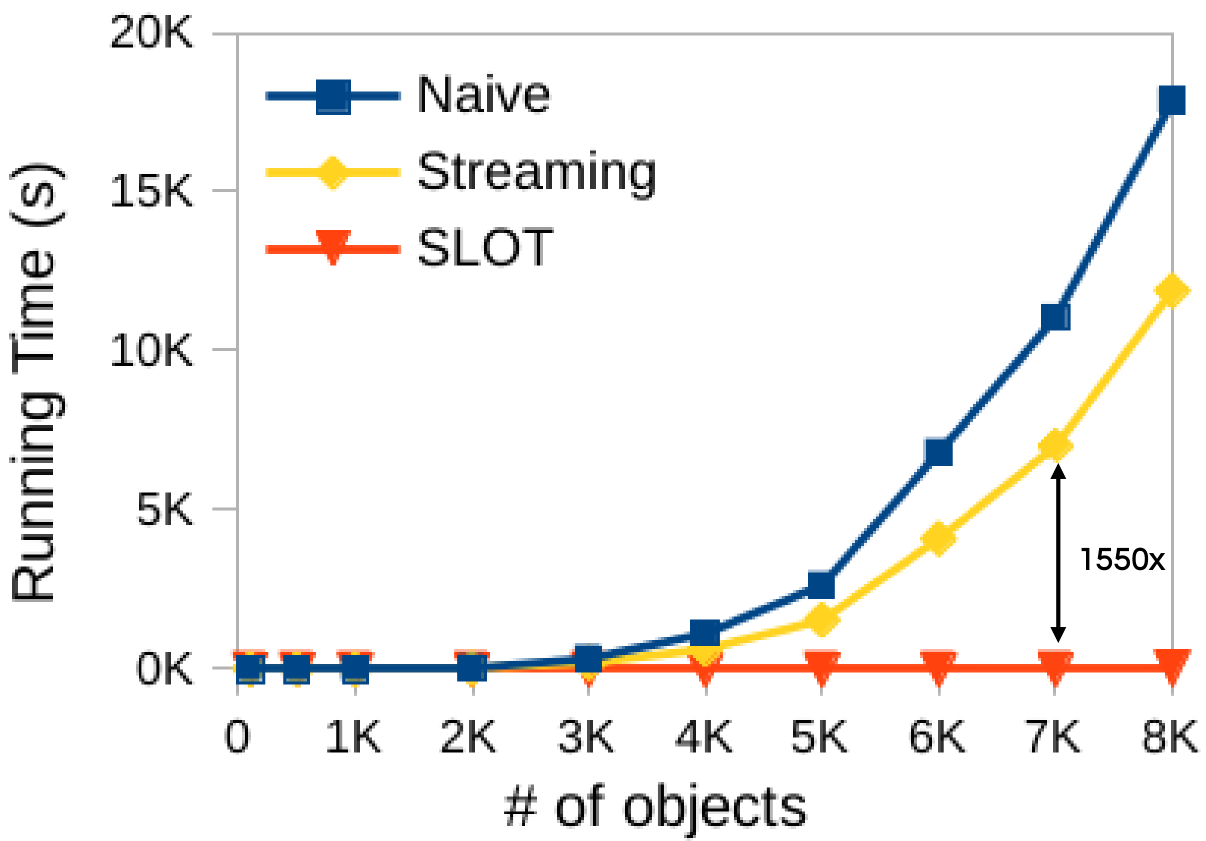




constant velocity

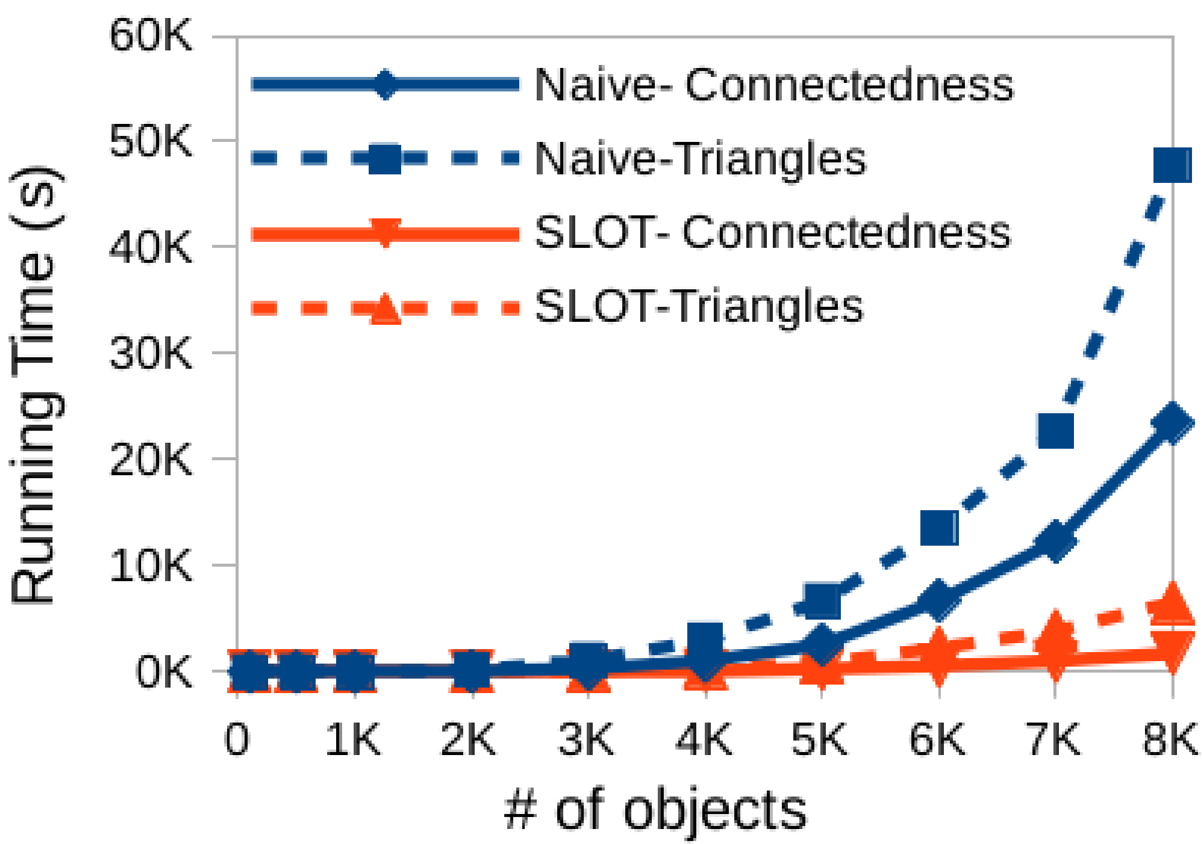
random velocity

Node degree

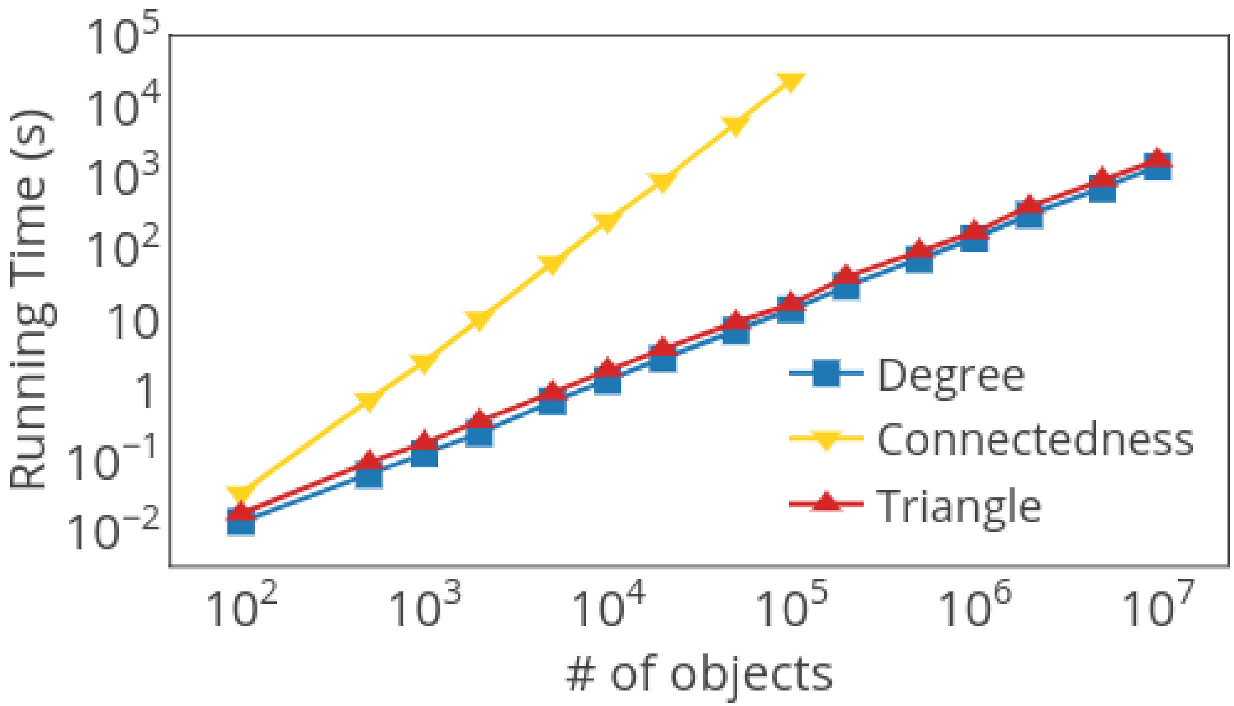


20K
15K
10K
5K
0K
Running Time (s)
Naive
Streaming
SLOT
1550x
0
1K
2K
3K
4K
5K
6K
7K
8K
# of objects

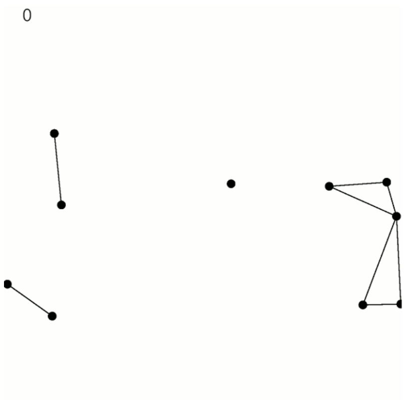
Triangle membership / connected components



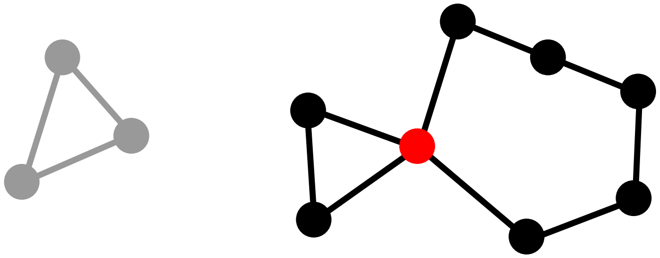
SLOT Scalability



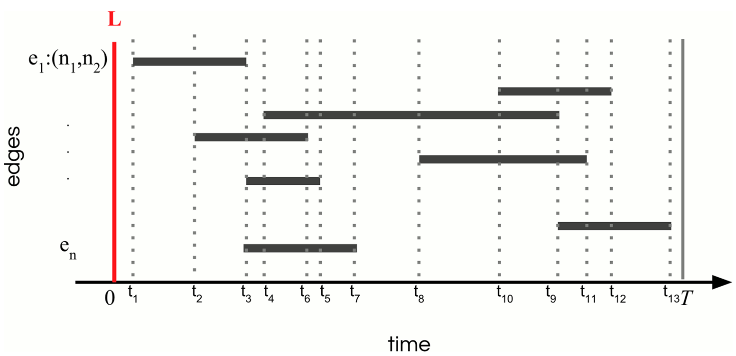
Takeaway



trajectory networks



network importance **over time**



SLOT algorithm

SLOT properties:

- fast
- exact
- scalable

Seagull migration trajectories

data from Wikelski et al. 2015

Group Pattern Discovery of Pedestrian Trajectories

Pedestrian trajectories

what is a
group?

many definitions, many algorithms

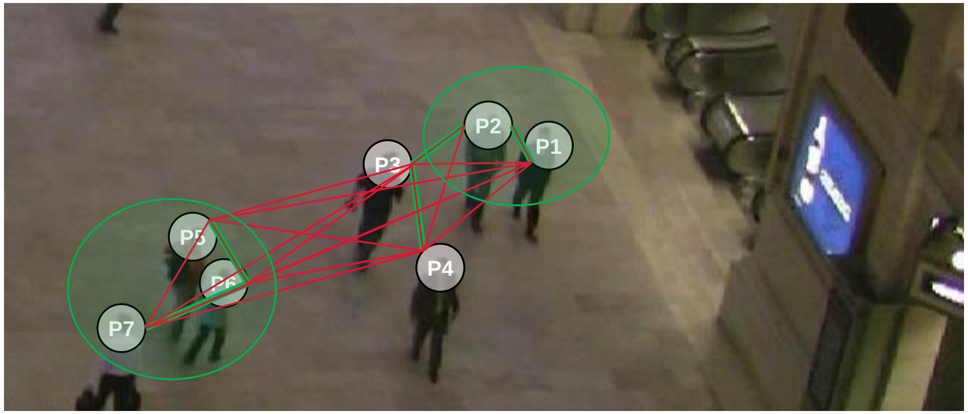
e.g., *flock*, *convoy*, *evolving-clusters*, *gathering-pattern*, ... [ACM TIST Tutorial 2015]

Finding pedestrian groups

Local Grouping

Intuitive method

Spatial-only



proximity threshold θ

key idea

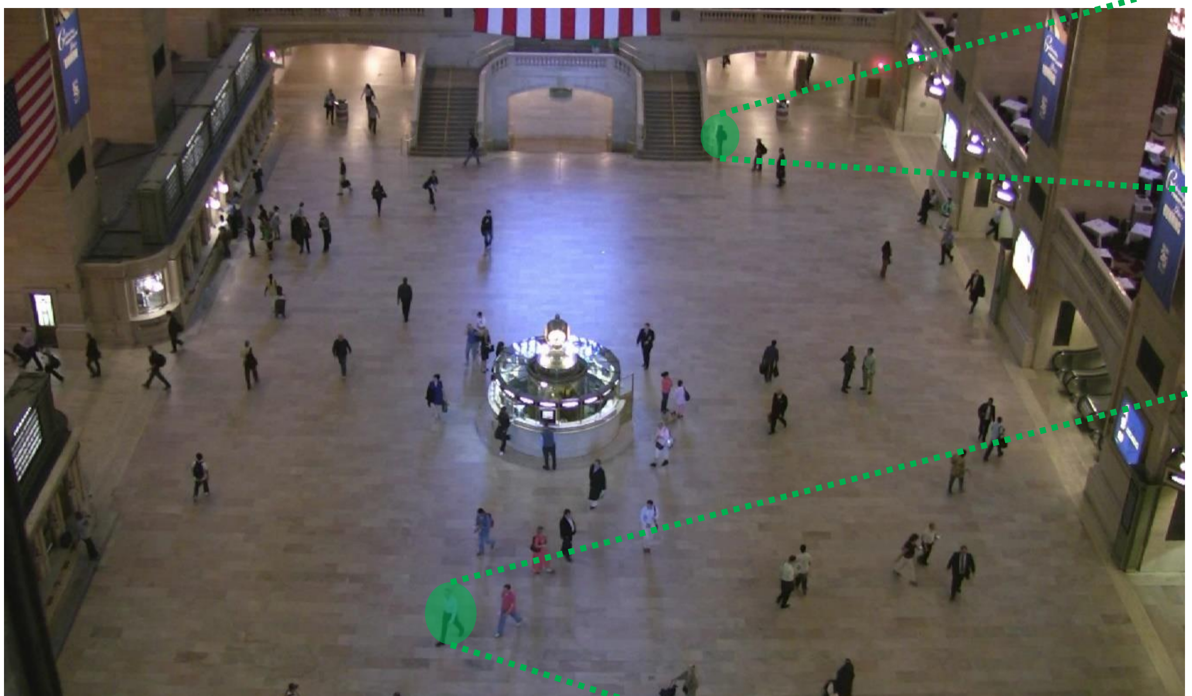
find **pairs** of pedestrians **x, y** where **distance(x, y) < θ**

expand **pairs** to discover **groups**

Local grouping

Challenge: Projection into ground plane

High perspective distortion - pedestrians closer to the camera appear larger than the ones farther away



Estimated Homography to overcome this distortion

expand the key idea
to include the
time dimension

Global groups vs. Time-window groups

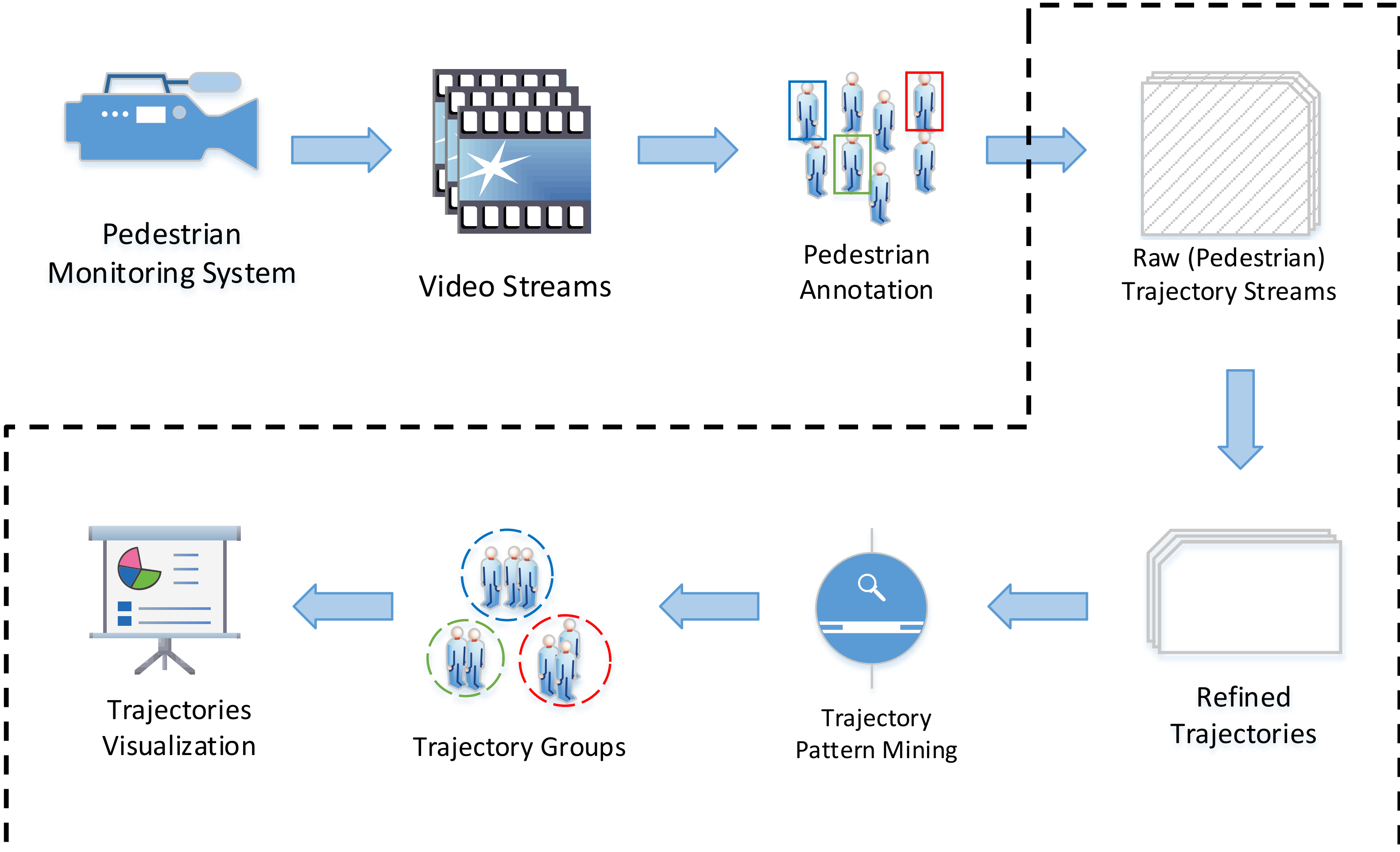
global grouping

time-window grouping

Trajectolizer

Demo

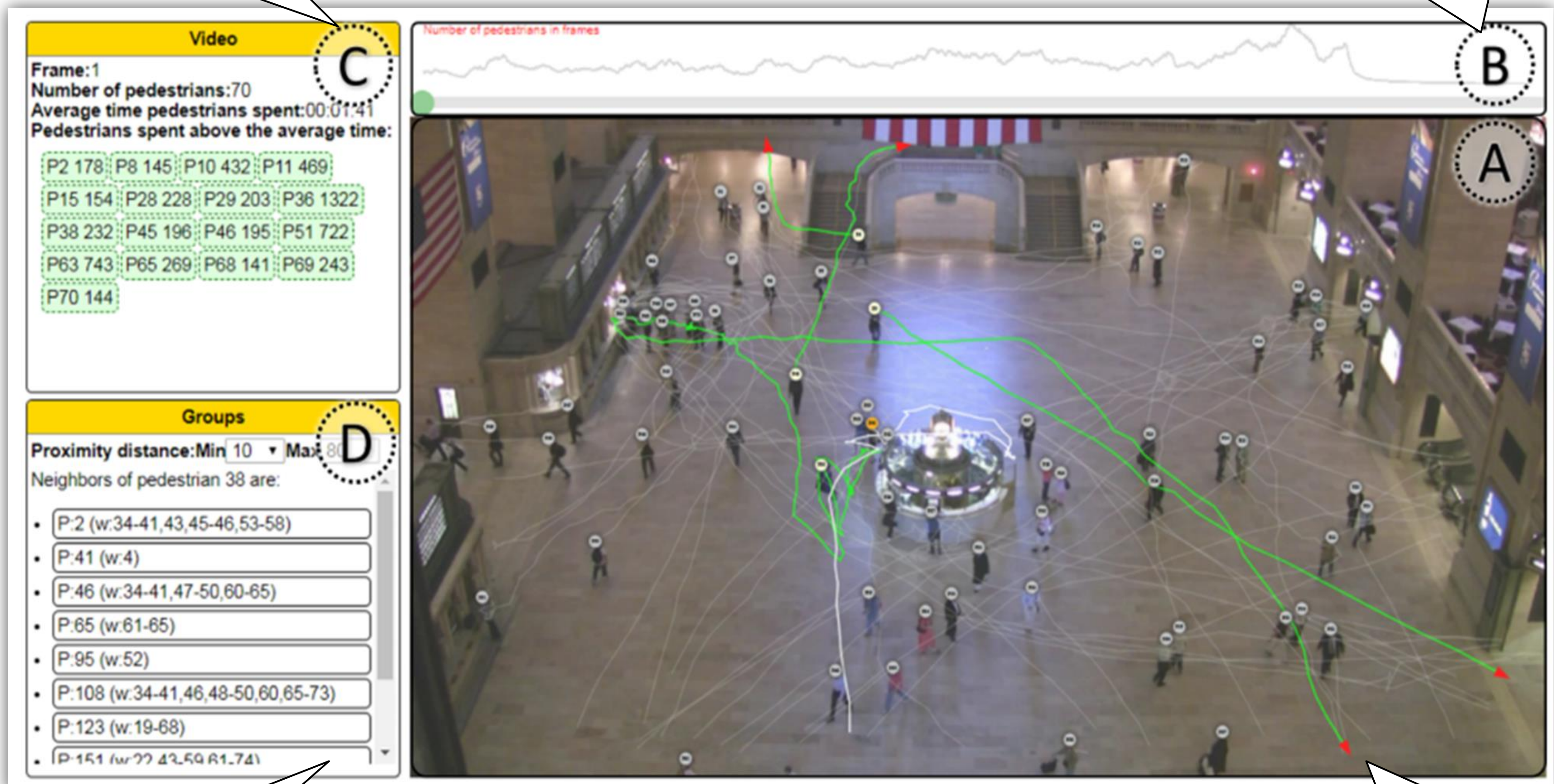
Trajectolizer: System Overview



Trajectolizer: Interactive Demo

descriptive statistics about the current frame

timeline slider area to navigate video frames

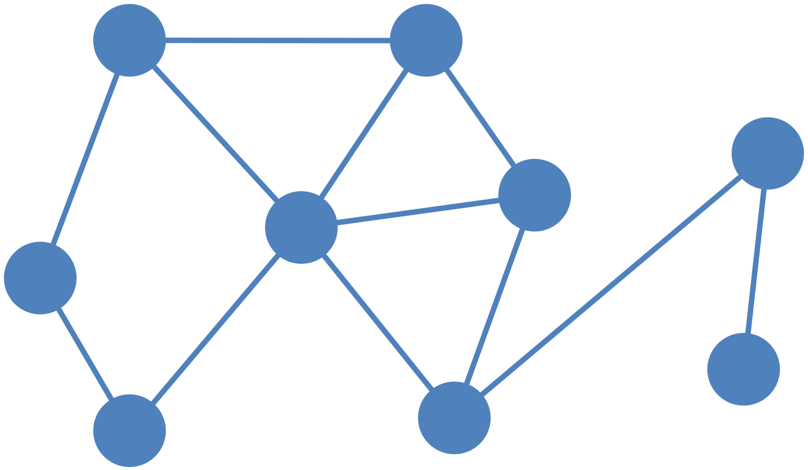


grouping analysis

[Live Demo](#)

current frame with pedestrian IDs and trajectories

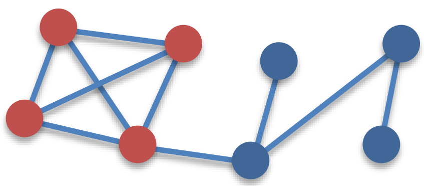
EvoNRL: Evolving Network Representation Learning Based on Random Walks



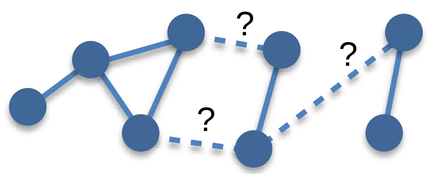
networks

(universal language for describing complex data)

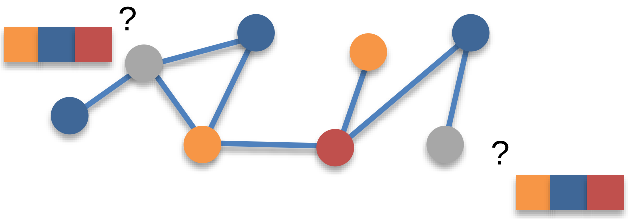
Classical ML Tasks in Networks



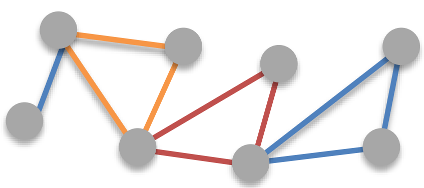
community detection



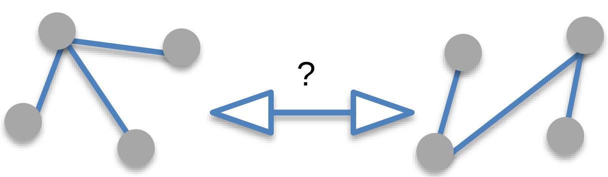
link prediction



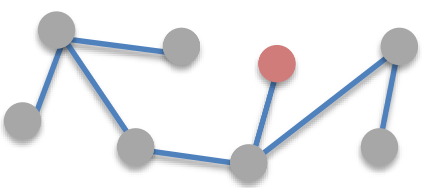
node classification



triangle count



graph similarity



anomaly detection

Limitations of Classical ML Tasks

expensive computation

(high dimension computations)

extensive domain knowledge

(task specific)

Network Representation Learning (NRL)

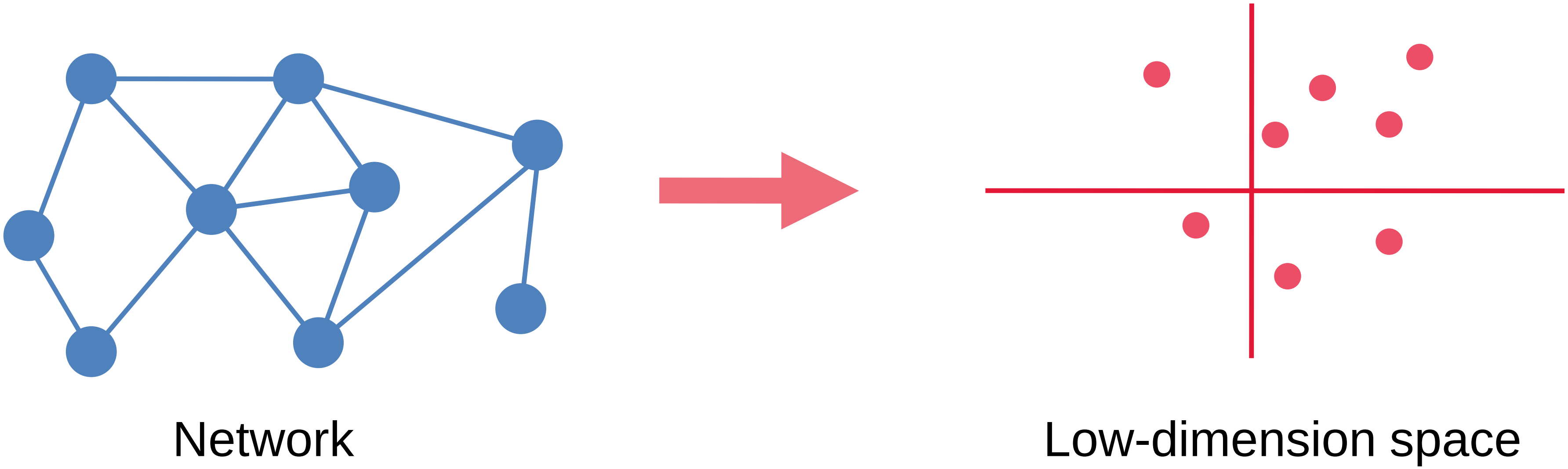
faster computations

(low dimension computations)

agnostic domain knowledge

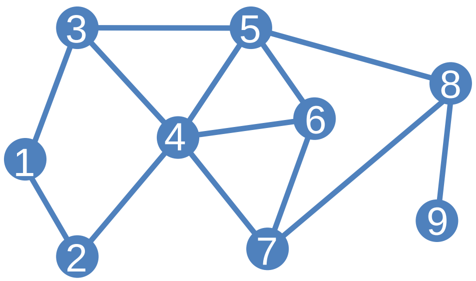
(task independent)

Network Representation Learning (NRL)

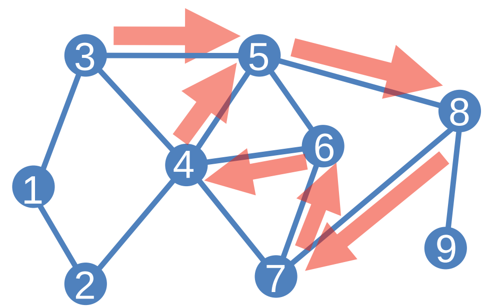


several network structural properties can be learned/embedded
(**nodes, edges, subgraphs, graphs, …**)

Random Walk-based NRL



Input network

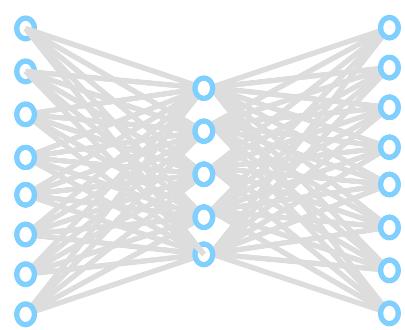


Obtain a set of random walks

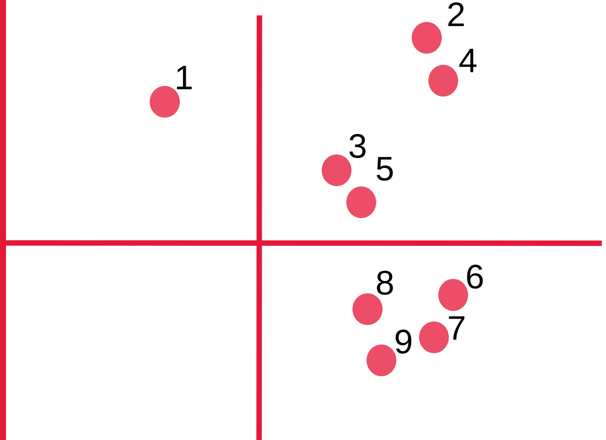
3 5 8 7 6 4 5

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 Skip-gram NN model



Learn a vector representation for each node

Random Walk-based NRL

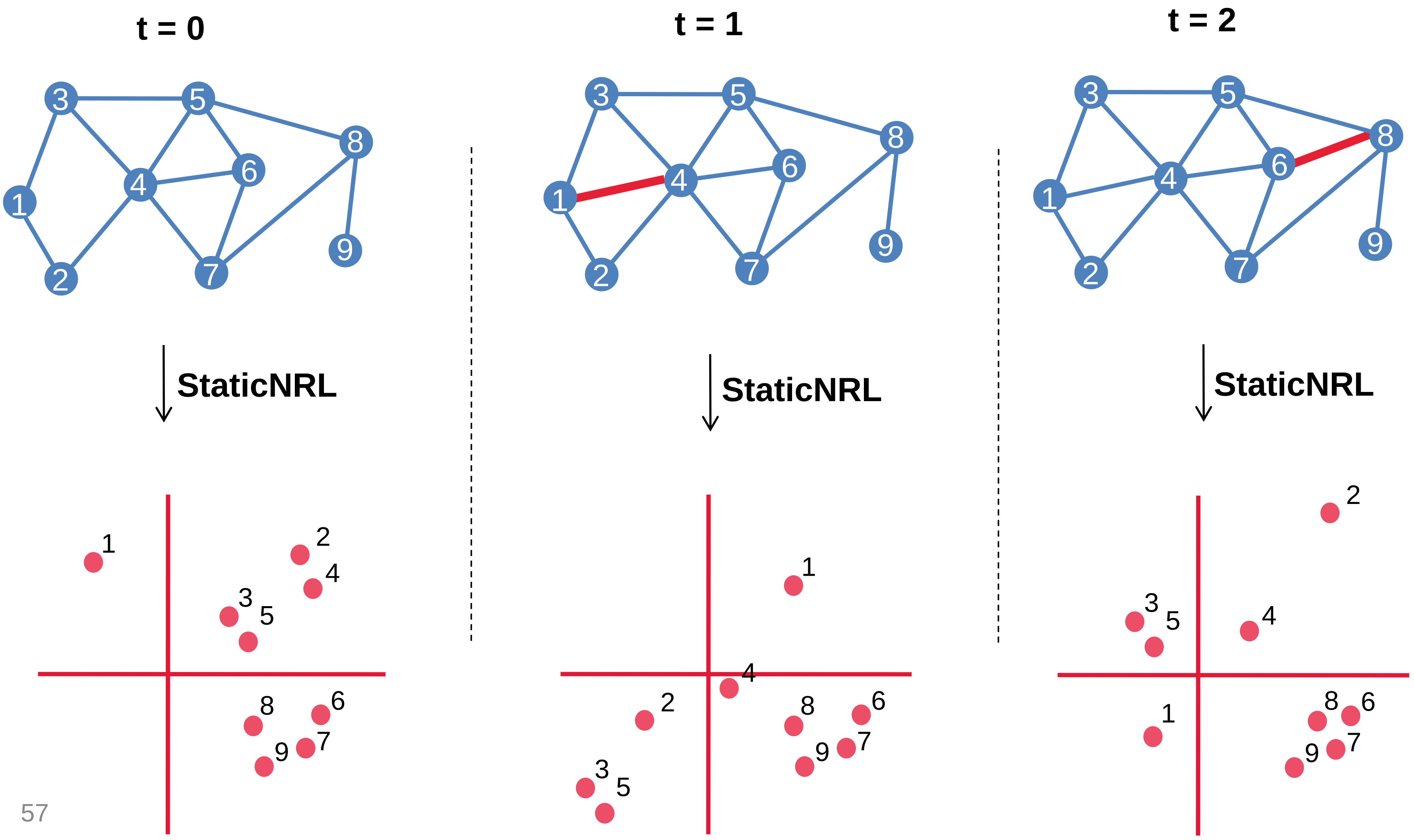
StaticNRL

DeepWalk
node2vec
…

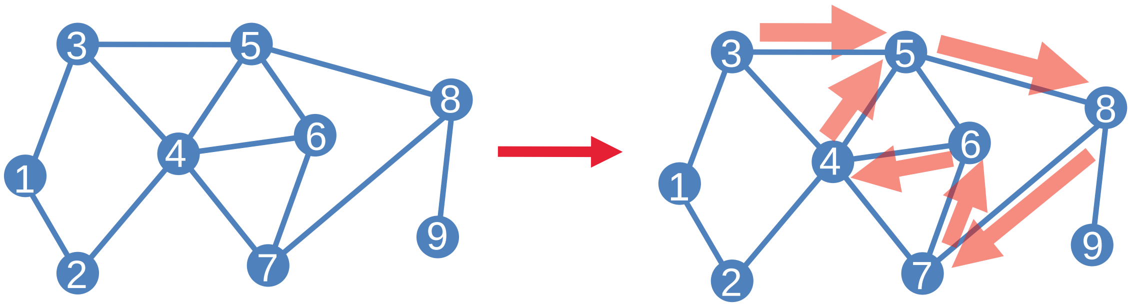
But…

real-world networks are constantly changing how can we learn representations of an evolving network?

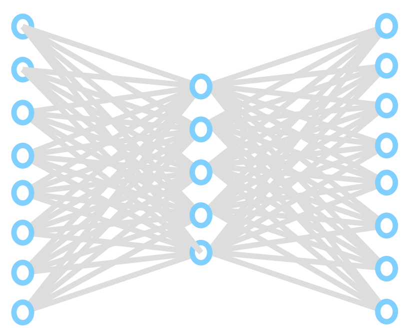
Naive Approach



Limitation #1

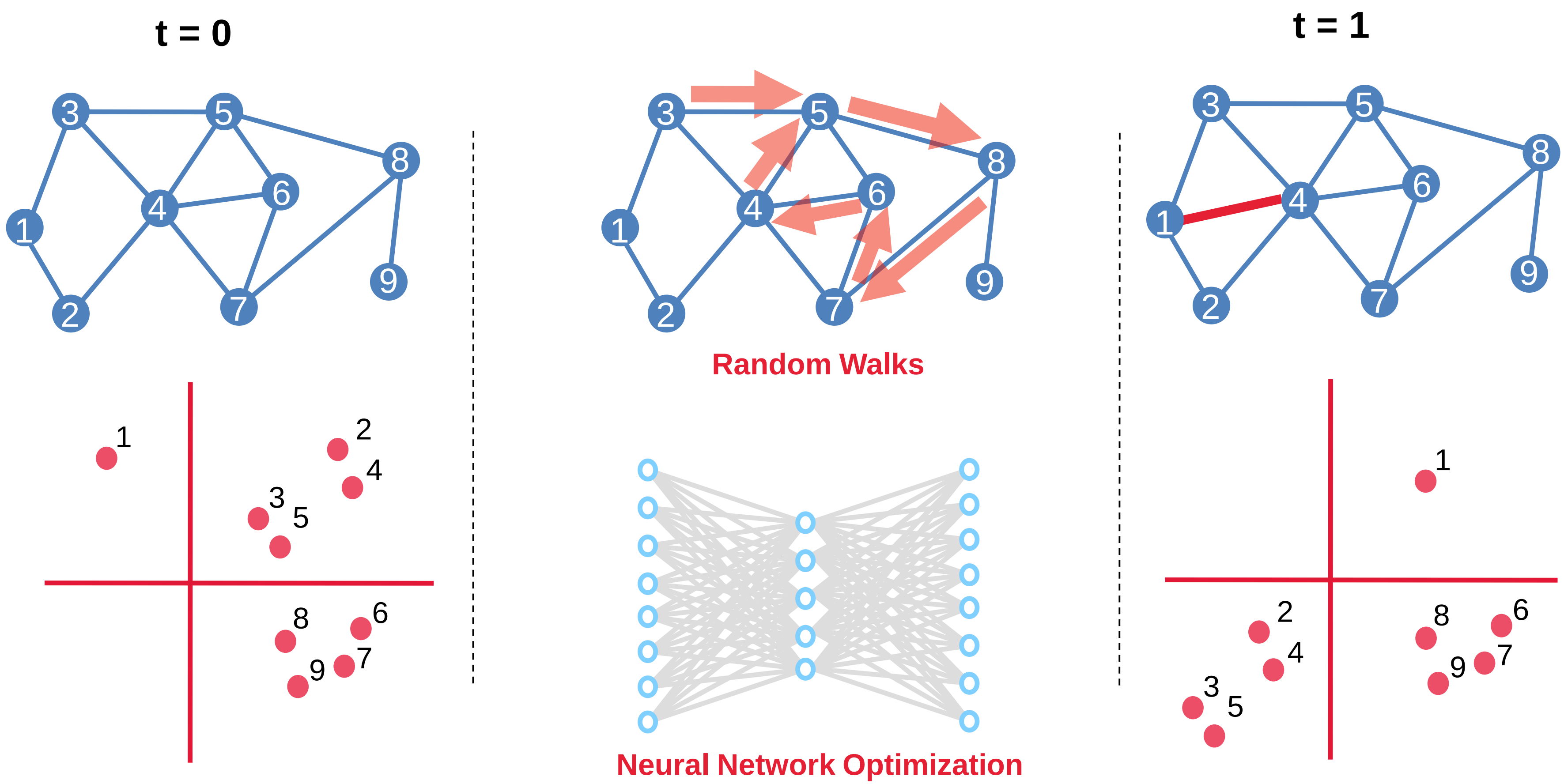


1	3 5 8 7 6 4 5
2	1 3 5 8 7 6 5
.	.
.	.
.	.
.	.
87	8 5 4 3 5 6 7
88	4 5 6 7 8 9
89	2 1 3 5 6 7 8
90	7 4 2 1 3 5 6



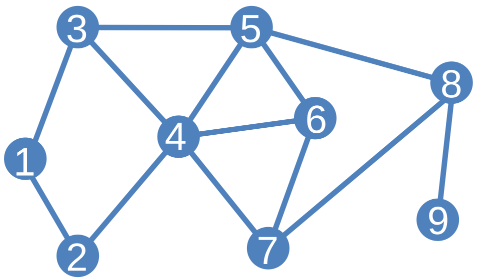
time expensive

Limitation #2

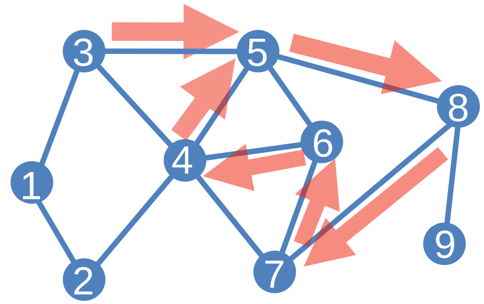


incomparable representations

EvoNRL Key Idea



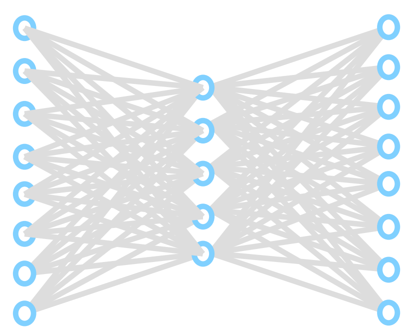
Input network



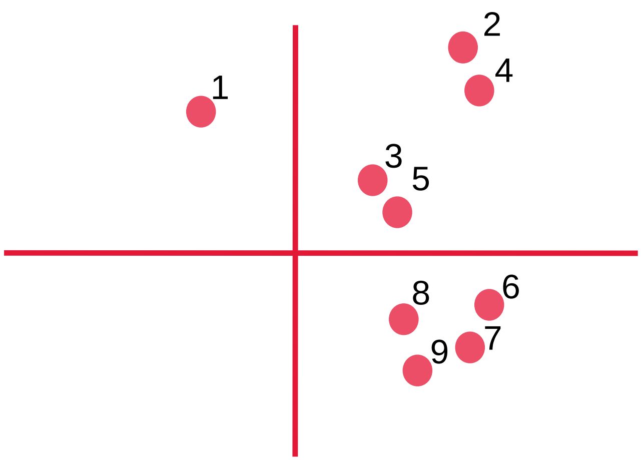
Obtain a set of
random walks

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
Skip-gram NN model

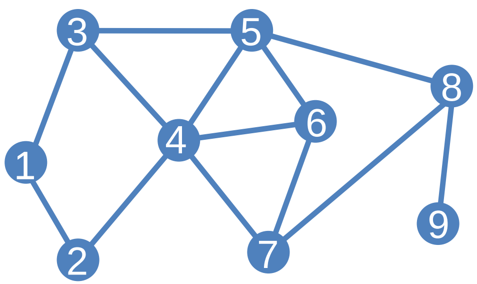


Learn a vector representation
for each node

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

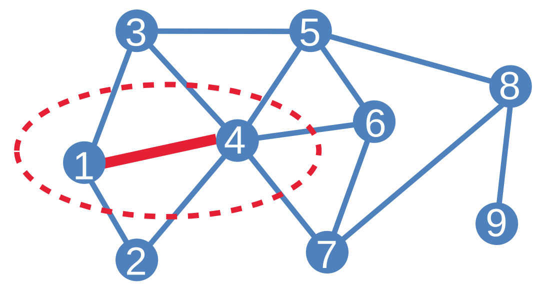
Example

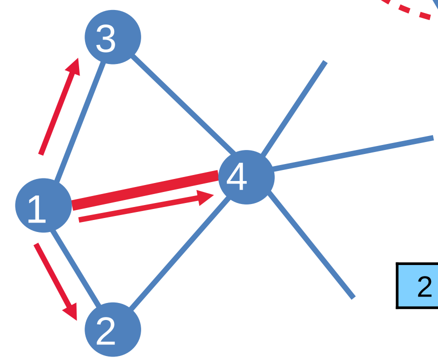
t = 0



addition of edge (1, 4)

t = 1





simulate the rest of the RW

2	1 4 3 5 6 7 8

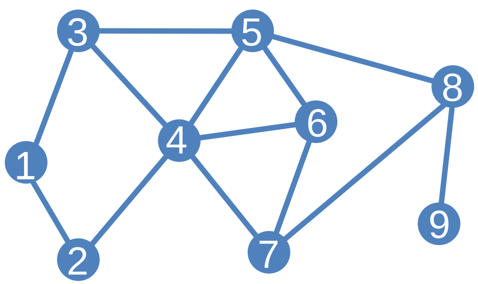
1	3 5 8 7 6 4 5
2	1 3 5 8 7 6 5
.	.
.	.
.	.
.	.
87	8 5 4 3 5 6 7
88	4 5 6 7 8 9
89	2 1 3 5 6 7 8
90	7 4 2 1 3 5 6

need to update the RW set

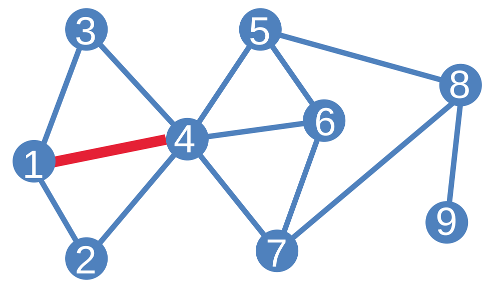
1	3 5 8 7 6 4 5
2	1 3 5 8 7 6 5
.	.
.	.
.	.
.	.
87	8 5 4 3 5 6 7
88	4 5 6 7 8 9
89	2 1 3 5 6 7 8
90	7 4 2 1 3 5 6

how can we efficiently maintain 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

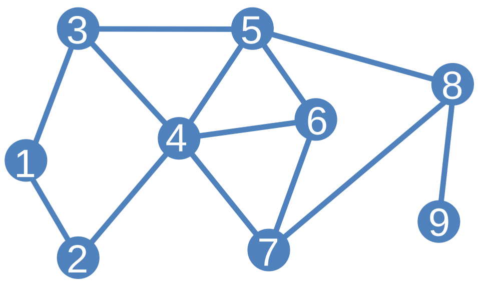
Operations on RW

Search a node

Delete a RW

Insert a new RW

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

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

elastic

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: EvoNRL vs StaticNRL

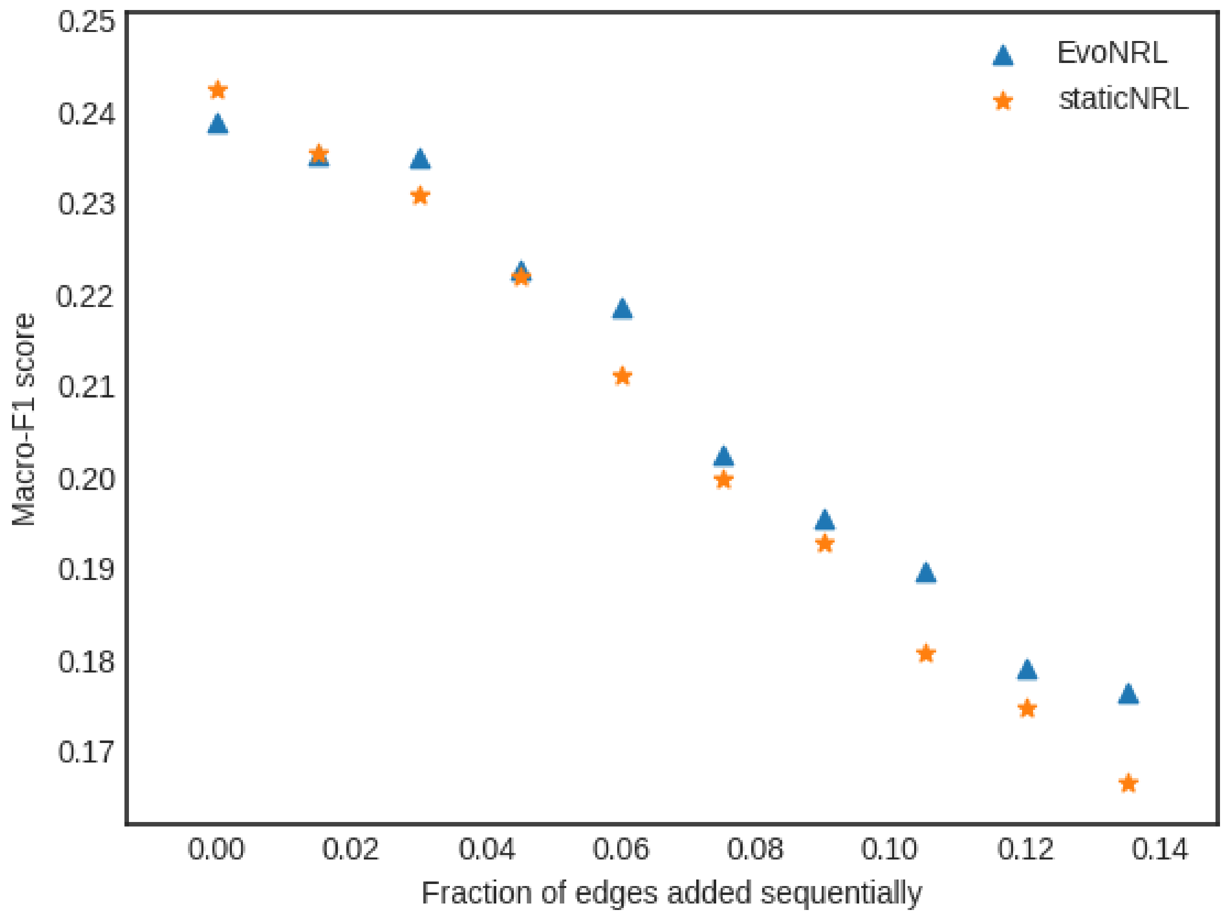
Accuracy

- EvoNRL ≈ StaticNRL

Running Time

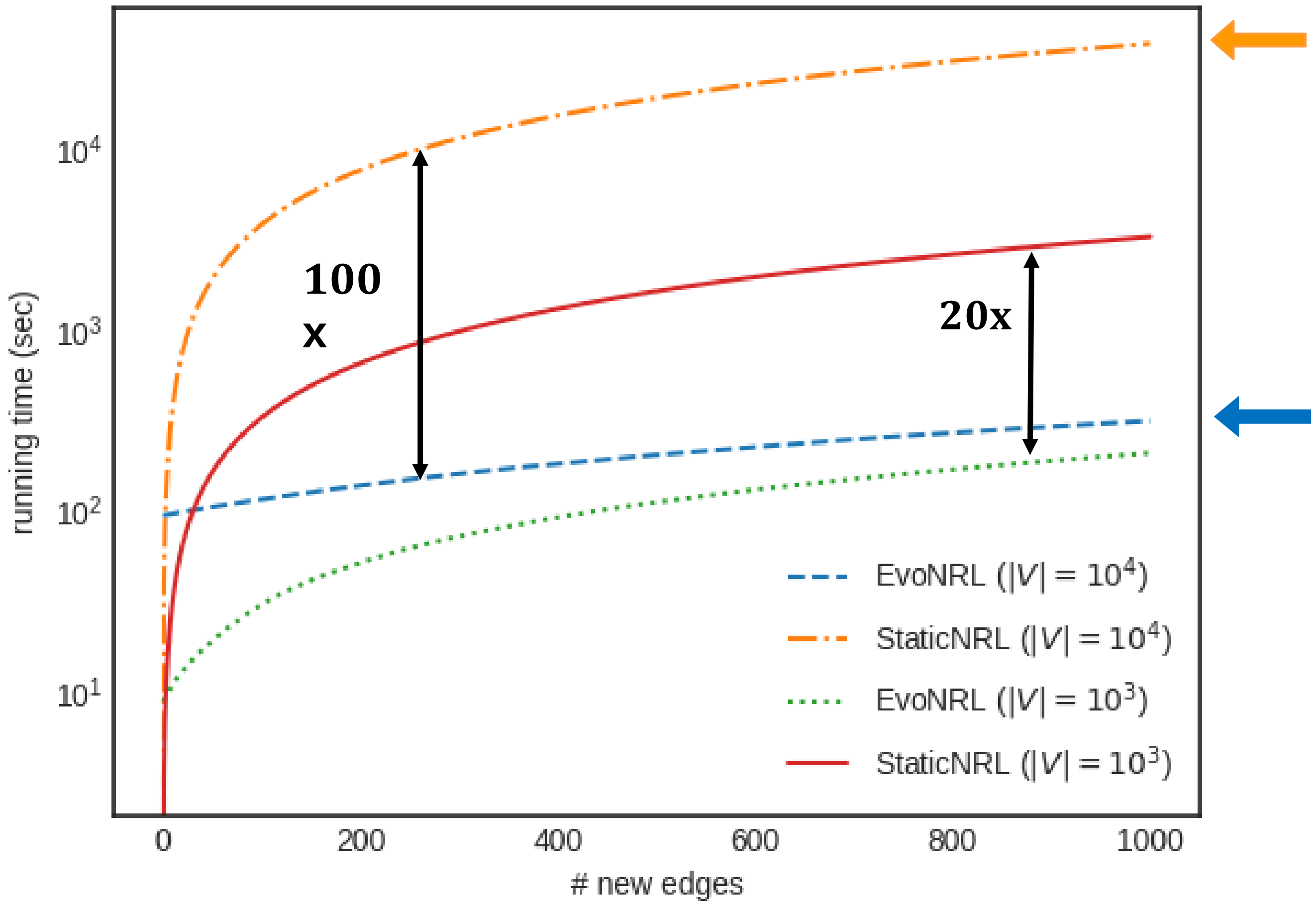
- EvoNRL << StaticNRL

Accuracy



EvoNRL has the similar accuracy as StaticNRL

Time Performance



EvoNRL performs orders of time faster than StaticNRL

Summary

how can we learn representations of an evolving network?

EvoNRL

time efficient

accurate

generic method

Thank you!

Questions?

Credits

Farzaneh Heidari

Tilemachos Pechlivanoglou

Abdullah Sawas

Data Mining Lab @ YorkU

Mahmoud Afifi

Abdullah Abuolaim

References

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

[IEEE MDM 2018] Tensor Methods for Group Pattern Discovery of Pedestrian Trajectories. Abdullah Sawas, Abdullah Abuolaim, Mahmoud Afifi, Manos Papagelis. Proceedings of the 19th IEEE International Conference on Mobile Data Management (IEEE MDM 2018, **best paper award**)

[IEEE MDM 2018] Trajectolizer: Interactive Analysis and Exploration of Trajectory Group Dynamics. Abdullah Sawas, Abdullah Abuolaim, Mahmoud Afifi, Manos Papagelis. Proceedings of the 19th IEEE International Conference on Mobile Data Management (IEEE MDM 2018, demo)

[Complex Networks 2018] EvoNRL: Evolving Network Representation Learning Based on Random Walks. Farzaneh Heidari, Manos Papagelis. Proceedings of the 7th International Conference on Complex Networks and Their Applications.