Large-scale Mining of Dynamic Networks

Manos Papagelis
York University, Toronto, Canada
Background
Research

Big Data & Knowledge Discovery

Data Mining

Graph Mining

Today
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
Today’s Overview

**Trajectory Network Mining**
- Mining of Node Importance in Trajectory Networks
- Group Pattern Discovery of Pedestrian Trajectories

**Evolving Network Mining**
- Evolving Network Representation Learning Based on Random Walks
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
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

\[ e_1 : (n_1, n_2) \]

\[ e_n \]

edges

time

L

0 \ t_1 \ t_2 \ t_3 \ t_4 \ t_5 \ t_6 \ t_7 \ t_8 \ t_9 \ t_{10} \ t_{11} \ t_{12} \ t_{13} T
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 $(u, v, \text{common})$

- **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 $(u, v, \text{common})$

- **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

The graph shows the running time (in seconds) as a function of the number of objects. The graph compares three methods: Naive, Streaming, and SLOT. The SLOT method shows a significant improvement over the Naive method, with a factor of 1550x increase in efficiency as the number of objects grows. The Streaming method also shows an improvement over the Naive method, but SLOT outperforms both significantly.
Triangle membership / connected components

![Graph showing running time vs number of objects for different methods: Naive-Connectedness, Naive-Triangles, SLOT-Connectedness, SLOT-Triangles. The graph indicates that SLOT-Triangles have the lowest running time for all object counts up to 8K.]
SLOT Scalability

![Graph showing running time vs. number of objects for Degree, Connectedness, and Triangle metrics.](image)
Takeaway

trajectory networks

network importance over time

SLOT properties:
- fast
- exact
- scalable
Seagull migration trajectories

data from Wikelski et al. 2015
Group Pattern Discovery of Pedestrian Trajectories

Joint work with Sawas Abdullah et al.
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

key idea
find pairs of pedestrians $x, y$ where $\text{distance}(x, y) < \theta$
expand pairs to discover groups
Local grouping
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

Pedestrian Monitoring System → Video Streams → Pedestrian Annotation → Raw (Pedestrian) Trajectory Streams

Trajectories Visualization → Trajectory Groups → Trajectory Pattern Mining → Refined Trajectories
Trajectolizer: Interactive Demo

- **C**: Descriptive statistics about the current frame
- **B**: Timeline slider area to navigate video frames
- **D**: Grouping analysis
- **A**: Current frame with pedestrian IDs and trajectories

**Live Demo**
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
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

Learn a vector representation for each node

Feed sentences to a Skip-gram NN model

Obtain a set of random walks

Treat a set of random walks as sentences

StaticNRL (DeepWalk, node2vec, ...)

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
but real-world networks are constantly evolving
Evolving Network Representations Learning
Naive Approach

$t = 0$

$t = 1$

$t = 2$

StaticNRL

StaticNRL

StaticNRL

Impractical (expensive, incomparable representations)
EvoNRL Key Idea

Input network

Obtain a set of random walks

Learn a vector representation for each node

Feed sentences to a Skip-gram NN model

Treat the set of random walks as sentences

dynamically maintain a valid set of random walks for every change in the network
Example: Edge Addition

$t = 0$

addition of edge (1, 4)

$t = 1$

.simulate the rest of the RW

need to update the RW set

.similarly for edge deletion, node addition/deletion
Efficiently Maintaining a Set of Random Walks
EvoNRL Operations

+ edge(n₁, n₂)

Operations on RW
- Search a node
- Delete a RW
- Insert a new RW

need for an efficient indexing data structure
EvoNRL Indexing

Each node is a keyword
Each RW is a document
A set of RWs is a collection of documents

<table>
<thead>
<tr>
<th>Term</th>
<th>Frequency</th>
<th>Postings and Positions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3</td>
<td>&lt;2, 1&gt;, &lt;89, 2&gt;, &lt;90, 4&gt;</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>&lt;89, 1&gt;, &lt;90, 3&gt;</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>&lt;1, 1&gt;, &lt;2, 1&gt;, &lt;87, 3&gt;, &lt;89, 3&gt;, &lt;90, 5&gt;</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>&lt;1, 6&gt;, &lt;87, 3&gt;, &lt;90, 2&gt;</td>
</tr>
<tr>
<td>5</td>
<td>9</td>
<td>&lt;1, 2&gt;, &lt;1, 7&gt;, &lt;2, 3&gt;, &lt;2, 7&gt;, &lt;87, 5&gt;, &lt;88, 2&gt;, &lt;89, 4&gt;, &lt;90, 6&gt;</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>&lt;1, 5&gt;, &lt;2, 6&gt;, &lt;87, 6&gt;, &lt;88, 3&gt;, &lt;89, 3&gt;, &lt;90, 5&gt;</td>
</tr>
<tr>
<td>7</td>
<td>5</td>
<td>&lt;1, 4&gt;, &lt;2, 5&gt;, &lt;87, 7&gt;, &lt;88, 4&gt;, &lt;89, 6&gt;, &lt;90, 7&gt;</td>
</tr>
<tr>
<td>8</td>
<td>5</td>
<td>&lt;1, 3&gt;, &lt;2, 4&gt;, &lt;87, 1&gt;, &lt;88, 6&gt;, &lt;89, 7&gt;</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>&lt;88, 7&gt;</td>
</tr>
</tbody>
</table>
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
Credits

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

code: [https://github.com/farzana0/EvoNRL/](https://github.com/farzana0/EvoNRL/)

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

code: [https://github.com/tipech/trajectory-networks](https://github.com/tipech/trajectory-networks)


[IEEE MDM 2018] **Tensor Methods for Group Pattern Discovery of Pedestrian Trajectories.** A. Sawas, A. Abuolaim, M. Afifi, M. Papagelis. (**best paper award**)

demo: [https://sites.google.com/view/pedestrians-group-pattern/](https://sites.google.com/view/pedestrians-group-pattern/)
Thank you!
Questions?
Data Mining Lab @ YorkU

• **Mandate**
  - Conduct basic research / knowledge transfer
  - Equip students with theoretical knowledge & practical experience
  - Research focus:
    ▪ data mining
    ▪ graph mining
    ▪ machine learning
    ▪ NLP
    ▪ big data analytics

• **Members**
  - Two Faculty Members (Prof. Aijun An, Prof. Manos Papagelis)
  - ~20 High Quality Personnel (HQP)
    ▪ ~5 Postdoc, ~6 PhDs, ~8 MSc, ~3 Undergrads, ~1 staff