Mining Large Dynamic Graphs and Tensors

Kijung Shin
Ph.D. Student
Carnegie Mellon University
(kijungs@cs.cmu.edu)
Thesis Committee

• Prof. Christos Faloutsos (Chair)
• Prof. Tom M. Mitchell
• Prof. Leman Akoglu
• Prof. Philip S. Yu
Mining Large Dynamic Graphs and Tensors
Graphs: Social Networks

[Diagram of social network with icons for Facebook, LinkedIn, and Google+]

Mining Large Dynamic Graphs and Tensors (by Kijung Shin)
Graphs: Purchase History

[Diagram showing the purchase history of items by different users on Amazon, eBay, and Alibaba.com]
Graphs: Many More
Properties of Real-world Graphs

• **Large**: many nodes, more edges
  
  - [WWW](#) 40B+ web pages
  - [Amazon](#) 500M+ products
  - [Facebook](#) 2B+ active users
  - [Wikipedia](#) 5M+ articles

• **Dynamic**: additions/deletions of nodes and edges
Properties of Real-world Graphs

- Rich with Attributes: timestamps, scores, text, etc.
### Matrices for Graphs

**Graph**

- Person
- Hat
- Mug
- Person
- Shoe
- Person
- Person
- Shirt

**Adjacency Matrix**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>0</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

The adjacency matrix represents the connections between the objects in the graph.
Tensors for Rich Graphs

- **Tensors**: multi-dimensional array

- 3-order tensor
  - (3-dimensional array)
- 4-order tensor
  - + Stars ★★★★
- 5-order tensor
  - + Text ...

3-order tensor
(3-dimensional array)
Research Goal and Tasks

• Goal:

To Understand
Large Dynamic Graphs and Tensors
on User Behavior

• Tasks

◦ T1. Structure Analysis
◦ T2. Anomaly Detection
◦ T3. Behavior Modeling
Tasks

Structure

Anomaly & Fraud

Behavior Model

Contrast
## Completed Work by Topics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs</strong></td>
<td><strong>Tensors</strong></td>
<td></td>
</tr>
<tr>
<td>Triangle Count</td>
<td>Summarization</td>
<td>Purchase Behavior</td>
</tr>
<tr>
<td>[ICDM17][PAKDD18]</td>
<td>[WSDM17]</td>
<td>[IJCAI17]</td>
</tr>
<tr>
<td>[submitted to KDD]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degeneracy</td>
<td>Dense Subtensors</td>
<td>Progressive Behavior</td>
</tr>
<tr>
<td>[ICDM16]* [KAIS18]*</td>
<td>[PKDD16][WSDM17]</td>
<td>[WWW18]</td>
</tr>
<tr>
<td></td>
<td>[KDD17][TKDD18]</td>
<td></td>
</tr>
</tbody>
</table>

* Duplicated
Approaches (Tools)

- A1. Distributed or external-memory algorithms
  - Hadoop
  - Spark

- A2. Streaming algorithms based on sampling

- A3. Approximation algorithms
  - and their combinations
Roadmap

• Overview

• Completed Work
  ◦ T1. Structure Analysis
  ◦ T2. Anomaly Detection
  ◦ T3. Behavior Modeling

• Proposed Work

• Conclusion
# Completed Work by Topics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triangle Count</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ICDM17] [PAKDD18]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[submitted to KDD]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degeneracy</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[ICDM16]* [KAIS18]*</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tensors</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Summarization</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[WSDM17]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Skip</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dense Subtensors</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[PKDD16] [WSDM17]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[KDD17] [TKDD18]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Purchase Behavior</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[IJCAI17]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Progressive Behavior</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[WWW18]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Duplicated
Roadmap

• Overview

• Completed Work
  ◦ T1. Structure Analysis
    ▪ T1.1 Waiting-Room Sampling
    ▪ T1.2-T1.3 Related Completed Work
  ◦ T2. Anomaly Detection
  ◦ T3. Behavior Modeling

• Proposed Work

• Conclusion

Kijung Shin, “WRS: Waiting Room Sampling for Accurate Triangle Counting in Real Graph Streams”, ICDM 2017
Graph Stream Model

- Widely-used data model for graphs

- **Sequence of edges**
  - graph is given over time as a sequence of edges
  - appropriate for **dynamic graphs**

- **Limited memory**
  - cannot store all edges in the stream
  - only samples or summaries
  - appropriate for **large graphs**
Relaxed Graph Stream Model

- **Chronological order**
  - edges are streamed in the order that they are created
  - natural for *dynamic graphs*
  - *temporal patterns can* exist
  - algorithms can *exploit* the patterns

Created at 9:21 AM
Created at 9:08 AM
Created at 9:02 AM
**Triangles in a Graph**

- **A triangle** is 3 nodes connected to each other.

- **The count of triangles** has many applications:
  - Community detection, spam detection, query optimization.

- **Global triangle count**: count of all triangles in the graph.
- **Local triangle count**: count of the triangles incident to each node.
Problem Definition

• Given:
  ◦ a sequence of edges in the chronological order
  ◦ memory budget $k$ (i.e., up to $k$ edges can be stored)

• Estimate: count of global triangles

• To Minimize: estimation error

“What are temporal patterns in real graph streams?”

“How can we exploit the patterns for accurate triangle counting?”
Roadmap

• Overview

• Completed Work
  ◦ T1. Structure Analysis
    ▪ T1.1 Waiting-Room Sampling
      ◦ Temporal Pattern
      ◦ Algorithm
      ◦ Experiments
  ▪ T1.2-T1.3 Related Completed Work
  ◦ T2. Anomaly Detection
  ◦ T3. Behavior Modeling

• Proposed Work

• Conclusion
Time Interval of a Triangle

- **Time interval** of a triangle:

arrival order of its last edge - arrival order of its first edge

Time interval: $7 - 2 = 5$
Time Interval Distribution

• Temporal Locality:
  ◦ average time interval is
  ◦ **2X shorter** in the chronological order
  ◦ than in a random order
Temporal Locality

• One interpretation:
  ◦ edges are more likely to form
  ◦ triangles with edges close in time
  ◦ than with edges far in time

• Another interpretation:
  ◦ new edges are more likely to form
  ◦ triangles with recent edges
  ◦ than with old edges

“How can we exploit temporal locality for accurate triangle counting?”
Roadmap

• Overview

• Completed Work
  ◦ T1. Structure Analysis
    ▪ T1.1 Waiting-Room Sampling
      ◦ Temporal Pattern
      ◦ Algorithm <<
    ◦ Experiments
  ▪ T1.2-T1.3 Related Completed Work
  ◦ T2. Anomaly Detection
  ◦ T3. Behavior Modeling

• Proposed Work

• Conclusion
Algorithm Overview

- \( \Delta \): estimate of triangle count
- \( p_{uvw} \): probability that triangle \((u, v, w)\) is discovered

(1) Arrival Step

new edge \( u - v \)

\[
\begin{array}{c|c|c|c|c}
  u & u & v & v \\
  | & | & | & |
  x & y & x & y \\
\end{array}
\]

memory

(2) Counting Step

\[
\Delta \leftarrow \Delta + 1/p_{uvy}
\]

\[
\begin{array}{c|c|c|c|c}
  u & u & v & v \\
  | & | & | & |
  x & y & x & y \\
\end{array}
\]

(3) Sampling Step

\[
\begin{array}{c|c|c|c|c}
  u & u & v & v \\
  | & | & | & |
  x & v & x & y \\
\end{array}
\]
Algorithm Overview (cont.)

- $\Delta$: estimate of triangle count
- $p_{uvw}$: probability that triangle $(u, v, w)$ is discovered

(1) Arrival Step

new edge $u - v$

<table>
<thead>
<tr>
<th>$u$</th>
<th>$u$</th>
<th>$v$</th>
<th>$v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
<td>$x$</td>
<td>$y$</td>
</tr>
</tbody>
</table>

memory
Algorithm Overview (cont.)

• $\Delta$: estimate of triangle count

• $p_{uvw}$: probability that triangle $(u, v, w)$ is discovered

### (1) Arrival Step

New edge: $u - v$

### (2) Counting Step

- $u - v$ discover!
- $u - v \backslash x$/

```
<table>
<thead>
<tr>
<th>u</th>
<th>u</th>
<th>v</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>u</th>
<th>u</th>
<th>v</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
</tbody>
</table>
```

$\Delta \leftarrow \Delta + 1/p_{uvx}$
**Algorithm Overview (cont.)**

- $\Delta$: estimate of triangle count
- $p_{uvw}$: probability that triangle $(u, v, w)$ is discovered

(1) Arrival Step

new edge $u - v$

<table>
<thead>
<tr>
<th>$u$</th>
<th>$u$</th>
<th>$v$</th>
<th>$v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
<td>$x$</td>
<td>$y$</td>
</tr>
</tbody>
</table>

memory

(2) Counting Step

$\Delta \leftarrow \Delta + 1/p_{uvy}$

$u - v$ discover!

$u - v$

\[
\begin{array}{cccc}
  u & u & v & v \\
  \mid & \mid & \mid & \mid \\
  x & y & x & y \\
\end{array}
\]
Algorithm Overview (cont.)

- $\Delta$: estimate of triangle count
- $p_{uvw}$: probability that triangle $(u, v, w)$ is discovered

(1) Arrival Step

New edge $u - v$

Memory

<table>
<thead>
<tr>
<th>u</th>
<th>u</th>
<th>v</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
</tbody>
</table>

(2) Counting Step

$\Delta \leftarrow \Delta + 1/p_{uvy}$

<table>
<thead>
<tr>
<th>u</th>
<th>u</th>
<th>v</th>
<th>v</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>y</td>
<td>x</td>
<td>y</td>
</tr>
</tbody>
</table>

(3) Sampling Step
Goal of Sampling Step

- to maximize discovering probability $p_{uvw}$

**Theorem. Variance** of our estimate:

$$\text{Var}[\Delta] \approx \sum_{(u,v,w)} \left( \frac{1}{p_{uvw}} - 1 \right)$$

**Theorem. Unbiasedness** of our estimate:

$$\text{Bias}[\Delta] = \text{Exp}[\Delta] - \text{True count} = 0$$

Estimation Error = Bias + Variance

- Estimation Error = 0
Increasing Discovering Prob.

“How can we increase discovering probabilities of triangles?”

• Recall Temporal Locality:
  ◦ new edges are more likely to form triangles with recent edges
  ◦ than with old edges

• Waiting-Room Sampling (WRS)
  ◦ treats recent edges better than old edges
  ◦ to exploit temporal locality
Waiting-Room Sampling (WRS)

- Divides memory space into two parts
  - Waiting Room: latest edges are **always stored**
  - Reservoir: the remaining edges are **sampled**

New edge: $e_{80}$

Waiting Room (FIFO):
- $e_{79}$, $e_{78}$, $e_{77}$, $e_{76}$

Reservoir (Random Replace):
- $e_{61}$, $e_{7}$, $e_{18}$, $e_{25}$, $e_{40}$, $e_{1}$, $e_{28}$

$\alpha$% of budget

(100 $-$ $\alpha$)% of budget
WRS: Sampling Steps (Step 1)

New edge $e_{80}$

Waiting Room (FIFO)

$e_{79}$  $e_{78}$  $e_{77}$  $e_{76}$

Reservoir (Random Replace)

$e_{61}$  $e_{7}$  $e_{18}$  $e_{25}$  $e_{40}$  $e_{1}$  $e_{28}$

Popped edge $e_{76}$

Waiting Room (FIFO)

$e_{80}$  $e_{79}$  $e_{78}$  $e_{77}$

Reservoir (Random Replace)

$e_{61}$  $e_{7}$  $e_{18}$  $e_{25}$  $e_{40}$  $e_{1}$  $e_{28}$
WRS: Sampling Steps (Step 2)

Popped edge $e_{76}$

Waiting Room (FIFO)

$e_{80}$ $e_{79}$ $e_{78}$ $e_{77}$

Reservoir (Random Replace)

$e_{61}$ $e_{7}$ $e_{18}$ $e_{25}$ $e_{40}$ $e_{1}$ $e_{28}$

store

or

discard

replace!
Summary of Algorithm

(1) Arrival Step
new edge $u - v$

<table>
<thead>
<tr>
<th>$u$</th>
<th>$u$</th>
<th>$v$</th>
<th>$v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
<td>$x$</td>
<td>$y$</td>
</tr>
</tbody>
</table>

memory

<table>
<thead>
<tr>
<th>$u$</th>
<th>$u$</th>
<th>$v$</th>
<th>$v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$y$</td>
<td>$x$</td>
<td>$y$</td>
</tr>
</tbody>
</table>

(2) Discovery Step
discover! $u - v$

$\Delta \leftarrow \Delta + \frac{1}{p_{uvx}}$

<table>
<thead>
<tr>
<th>$u$</th>
<th>$u$</th>
<th>$v$</th>
<th>$v$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x$</td>
<td>$v$</td>
<td>$x$</td>
<td>$y$</td>
</tr>
</tbody>
</table>

(3) Sampling Step

Waiting-Room Sampling!

Completed / Proposed | T1.1 / T1.2 / T1.3
Roadmap

• Overview

• Completed Work
  ◦ T1. Structure Analysis
    ▪ T1.1 Waiting-Room Sampling
      ◦ Temporal Pattern
      ◦ Algorithm
      ◦ Experiments <<
    ▪ T1.2-T1.3 Related Completed Work
  ◦ T2. Anomaly Detection
  ◦ T3. Behavior Modeling

• Proposed Work

• Conclusion
Experimental Results: Accuracy

• Datasets: [arXiv.org]

• WRS is most accurate (reduces error up to 47%)
Discovering Probability

• WRS increases discovering probability $p_{uvw}$
• WRS discovers up to $3 \times$ more triangles
Roadmap

• Overview

• Completed Work
  ◦ T1. Structure Analysis
    ▪ T1.1 Waiting-Room Sampling
    ▪ **T1.2-T1.3 Related Completed Work**
  ◦ T2. Anomaly Detection
  ◦ T3. Behavior Modeling

• Proposed Work

• Conclusion

Mining Large Dynamic Graphs and Tensors (by Kijung Shin)
T1.2 Distributed Counting of Triangles

• Goal: to utilize *multiple machines* for triangle counting in a graph stream?

**Tri-Fly [PAKDD18]**

- Sources
- Workers
- Aggregators

   - Broadcast
   - Shuffle

**DiSLR [submitted to KDD]**

- Sources
- Workers
- Aggregators

   - Multicast
   - Shuffle

---

**Kijung Shin**, Mohammad Hammoud, Euiwoong Lee, Jinoh Oh, and Christos Faloutsos, “Tri-Fly: Distributed Estimation of Global and Local Triangle Counts in Graph Streams”, PAKDD 2018
T1.2 Performance of Tri-Fly and DiSLR

- Estimation Error = Bias + Variance

![Graph showing comparison between Tri-Fly and DiSLR]
T1.3 Estimation of Degeneracy

• Goal: to estimate the *degeneracy* in a graph stream?

• *Core-Triangle Pattern*
  ◦ 3:1 power law between the triangle count and the degeneracy

*degeneracy*: maximum $k$ such that a subgraph where every node has degree at least $k$ exists.

Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos, “Patterns and Anomalies in kCores of Real-world Graphs with Applications”, KAIS 2018 (previously ICDM 2016)
T1.3 Core-D Algorithm

- **Core-D**: one-pass streaming algorithm for degeneracy

\[ \hat{d} = \exp(\alpha \cdot \log(\hat{\Delta}) + \beta) \]

Estimated Degeneracy

Estimated Triangle Count (obtained by WRS, etc.)

Completed / Proposed
Structure Analysis of Graphs

**Models:**
- Relaxed graph stream model
- Distributed graph stream model

**Patterns:**
- Temporal locality
- Core-Triangle pattern

**Algorithms:**
- WRS, Tri-Fly, and DiSLR
- Core-D

**Analyses:** bias and variance
### Completed Work by Topics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triangle Count</td>
<td>✓</td>
<td>skip</td>
</tr>
<tr>
<td>[ICDM17] [PAKDD18]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[submitted to KDD]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Degeneracy</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>[ICDM16] [KAIS18]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tensors</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Summarization</td>
<td>skip</td>
<td></td>
</tr>
<tr>
<td>[WSDM17]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Dense Subtensors</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[PKDD16] [WSDM17]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[KDD17] [TKDD18]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Duplicated

**Mining Large Dynamic Graphs and Tensors (by Kijung Shin)**
Roadmap

• Overview

• Completed Work
  ◦ T1. Structure Analysis
  ◦ T2. Anomaly Detection
    ▪ T2.1 M-Zoom
    ▪ T2.2-T2.3 Related Completed Work
  ◦ T3. Behavior Modeling

• Proposed Work

• Conclusion

Motivation: Review Fraud

Alice’s

🌟🌟🌟🌟 8 reviews

Bob’s

🌟🌟🌟🌟🌟 149 reviews

Carol’s

🌟🌟🌟🌟🌟 239 reviews

Get more 5-star Yelp reviews for your business

Alice
Fraud Forms Dense Block

Restaurants

Accounts

Completed / Proposed

T2.1 / T2.2 / T2.3
Problem: Natural Dense Subgraphs

• Question. How can we distinguish them?

natural dense blocks (core, community, etc.)

suspicious dense blocks formed by fraudsters

**Completed / Proposed** | **T2.1 / T2.2 / T2.3**
Solution: Tensor Modeling

- Along the time axis...
  - Natural dense blocks are sparse (formed gradually)
  - Suspicious dense blocks are dense (synchronized behavior)

- In the tensor model
  - Suspicious dense blocks become denser than natural dense blocks
Solution: Tensor Modeling (cont.)

• High-order tensor modeling:
  ◦ any side information can be used additionally

“Given a large-scale high-order tensor, how can we find dense blocks in it?”
Problem Definition

• Given: (1) $R$: an $N$-order tensor,
  (2) $\rho$: a density measure,
  (3) $k$: the number of blocks we aim to find

• Find: $k$ distinct dense blocks maximizing $\rho$
Density Measures

• How should we define “density” (i.e., $\rho$)?
  ◦ no one absolute answer
  ◦ depends on data, types of anomalies, etc.

• Goal: flexible algorithm working well with various reasonable measures
  ✓ Arithmetic avg. degree $\rho_A$
  ✓ Geometric avg. degree $\rho_G$
  ✓ Suspiciousness (KL Divergence) $\rho_S$
  ✗ Traditional Density: $\rho_T(B) = \frac{\text{EntrySum}(B)}{\text{Vol}(B)}$
    - maximized by a single entry with the maximum value
Clarification of Blocks (Subtensors)

- The concept of blocks (subtensors) is independent of the orders of rows and columns
- Entries in a block do not need to be adjacent
Roadmap

• Overview

• Completed Work
  ◦ T1. Structure Analysis
  ◦ T2. Anomaly Detection
    ▪ T2.1 M-Zoom [PKDD 16]
      ◦ Algorithm <<
      ◦ Experiments
    ▪ T2.2-T2.3 Related Completed Work
  ◦ T3. Behavior Modeling

• Proposed Work

• Conclusion
Single Dense Block Detection

- Greedy search
- Starts from the entire tensor

\[ \rho = 2.9 \]

\[
\begin{array}{ccc}
5 & 3 & 0 \\
4 & 6 & 1 \\
2 & 0 & 0 \\
\end{array}
\]
Single Dense Block Detection (cont.)

- Remove a slice to maximize density $\rho$
Single Dense Block Detection (cont.)

- Remove a slice to maximize density $\rho$

$\rho = 3.3$
Single Dense Block Detection (cont.)

- Remove a slice to maximize density $\rho$

$\rho = 3.6$
Single Dense Block Detection (cont.)

- Until all slices are removed
Single Dense Block Detection (cont.)

- Output: return the densest block so far

\[ \rho = 3.6 \]
Speeding Up Process

- Lemma 1 [Remove Minimum Sum First]

Among slices in the same dimension, removing the slice with smallest sum of entries increases $\rho$ most
Accuracy Guarantee

• Theorem 1 [Approximation Guarantee]

\[ \rho_A(B) \geq \frac{1}{N} \rho_A(B^*) \]

M-Zoom Result
Order
Densest Block

• Theorem 2 [Near-linear Time Complexity]

\[ O(NM \log L) \]

Order
# Non-zeros
# Entries in each mode
Optional Post Process

• Local search
  ◦ grow or shrink until a local maximum is reached

\[ \rho = 2 \]
\[
\begin{array}{c|c}
  1 & 0 \\
  3 & 4 \\
  5 & 7 \\
  1 & 0 \\
\end{array}
\]
grow

\[ \rho = 1.8 \]
\[
\begin{array}{c|c}
  1 & 0 \\
  3 & 4 \\
  5 & 7 \\
  1 & 0 \\
\end{array}
\]
shrink

\[ \rho = 3.29 \]
\[
\begin{array}{c|c}
  1 & 0 \\
  3 & 4 \\
  5 & 7 \\
  1 & 0 \\
\end{array}
\]
result of our previous greedy search
Optional Post Process (cont.)

• Local search
  ◦ grow or shrink until a local maximum is reached

\[ \rho = 3.29 \]

\[ \begin{array}{ccc}
1 & 0 & 3 \\
3 & 4 & 0 \\
5 & 7 & 0 \\
1 & 0 & 1 \\
\end{array} \]

\[ \rightarrow \]

\[ \begin{array}{ccc}
1 & 0 & 3 \\
3 & 4 & 0 \\
5 & 7 & 0 \\
1 & 0 & 1 \\
\end{array} \]

\[ \rho = 3.33 \]

\[ \rho = 3.25 \]
Optional Post Process (cont.)

• Local search
  ◦ grow or shrink until a local maximum is reached

\[ \rho = 3.29 \]

\[ \rho = 3.33 \]

\[ \rho = 3.8 \]
Optional Post Process (cont.)

- Local search
  - grow or shrink until a local maximum is reached
- Return the local maximum
Multiple Block Detection

- **Deflation**: Remove found blocks before finding others

---

Completed / Proposed  | T2.1 / T2.2 / T2.3

70/106
Roadmap

- Overview

- Completed Work
  - T1. Structure Analysis
  - T2. Anomaly Detection
    - T2.1 M-Zoom [PKDD 16]
      - Algorithm
      - Experiments <<
    - T2.2-T2.3 Related Completed Work
  - T3. Behavior Modeling

- Proposed Work

- Conclusion
Speed & Accuracy

- Datasets: Y!, Wikipedia, TCP/IP, Yelp, Android, SMS, ...

Density metric: \( \rho_G \)

Density metric: \( \rho_A \)

Density metric: \( \rho_S \)
Discoveries in Practice

Korean Wikipedia
- 11 accounts revised 10 pages 2,305 times within 16 hours

English Wikipedia
- 8 accounts revised 12 pages 2.5 million times
Discoveries in Practice (cont.)

App Market (4-order)
- 9 accounts gives 1 product
- 369 reviews with the same rating within 22 hours

TCP Dump (7-order)
- A block whose volume = 2 and mass = 2 millions

Completed / Proposed | T2.1 / T2.2 / T2.3
Roadmap

• Overview

• Completed Work
  ◦ T1. Structure Analysis
  ◦ T2. Anomaly Detection
    ▪ M-Zoom
    ▪ T2.2-T2.3 Related Completed Work
  ◦ T3. Behavior Modeling

• Proposed Work

• Conclusion
T2.2 Extension to Web-scale Tensors

• Goal: to find dense blocks in a disk-resident or distributed tensor

• D-Cube: gives the same accuracy guarantee of M-Zoom with much less iterations

Entry sum in slices

Average

Elapsed Time (sec)

Number of Non–zeros

100 B nonzeros in 5 hours

Kijung Shin, Bryan Hooi, Jisu Kim, and Christos Faloutsos,
“D-Cube: Dense-Block Detection in Terabyte-Scale Tensors”, WSDM 2017
T2.3 Extension to Dynamic Tensors

- Goal: to maintain a dense block in a **dynamic tensor** that changes over time

- **DenseStream**: incrementally computes a dense block with the **same accuracy guarantee** of M-Zoom

---

Anomaly Detection in Tensors

 Algorithms:  
   ◦ M-Zoom, D-Cube, and DenseStream

 Analyses: approximation guarantees

 Discoveries:  
   ◦ Edit war, vandalism, and bot activities  
   ◦ Network intrusion  
   ◦ Spam reviews
# Completed Work by Topics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Triangle Count</td>
<td>✓ [ICDM17][PAKDD18]</td>
<td>skip</td>
<td></td>
</tr>
<tr>
<td>Degeneracy</td>
<td>✓ [ICDM16]* [KAIS18]*</td>
<td>skip</td>
<td></td>
</tr>
<tr>
<td>Tensors</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Summarization</td>
<td>skip</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dense Subtensors</td>
<td>✓ [PKDD16][WSDM17][KDD17][TKDD18]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Duplicated
Problem Definition

• **Given:**
  ◦ behavior log
  ◦ number of desired latent stages: $k$

• **Find:** $k$ progression stages
  ◦ types of actions
  ◦ frequency of actions
  ◦ transitions to other stages

• **To best describe** the given behavior log
Behavior Model

- Generative process:
  - $\Theta_s$: action-type distribution in stage $s$
  - $\phi_s$: time-gap distribution in stage $s$
  - $\psi_s$: next-stage distribution in stage $s$

- Constraint: “no decline” (progression but no cyclic patterns)
Optimization Algorithm

- **Goal:** to fit our model to given data
  - parameters: distributions (i.e., $\Theta_s, \phi_s, \psi_s$) and latent stages

- **repeat** until convergence
  - **assignment step:** assign latent stages while fixing prob. distributions
  - **update step:** update prob. distributions while fixing latent stages
    - e.g., $\Theta_s \leftarrow$ ratio of the types of actions in stage $s$

```
1 | 2 | 3
```

"no decline" → Dynamic Programming
Scalability & Convergence

- Three versions of our algorithm
  - In-memory
  - Out-of-core (or external-memory)
  - Distributed

- 1 trillion actions in 2 hours
- 5 latent stages

- Completed / Proposed
Progression of Users in LinkedIn

Join → Build one’s Profile → Onboarding Process

Poke around the service → Grow one’s Social Network → Consume Newsfeeds → Have 30 connections

Completed / Proposed | T3.1 | 85/106
## Completed Work by Topics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Triangle Count</td>
<td><strong>✓</strong> [ICDM17]</td>
<td><strong>skip</strong></td>
<td><strong>skip</strong></td>
</tr>
<tr>
<td>[ICDM17]</td>
<td></td>
<td>[submitted to KDD]</td>
<td></td>
</tr>
<tr>
<td>Degeneracy</td>
<td><strong>✓</strong> [ICDM16]* [KAIS18]</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Tensors</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Summarization</td>
<td><strong>skip</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[WSDM17]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dense Subtensors</td>
<td><strong>✓</strong> [PKDD16][WSDM17]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>[KDD17][TKDD18]</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Progressive</td>
<td><strong>✓</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Behavior</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>[WWW18]</td>
<td><strong>skip</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>* Duplicated</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Roadmap

• Overview

• Completed Work
  ◦ T1. Structure Analysis
  ◦ T2. Anomaly Detection
  ◦ T3. Behavior Modeling

• Proposed Work <<

• Conclusion
## Proposed Work by Topics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs</strong></td>
<td><strong>P1. Triangle Counting in Fully Dynamic Stream</strong></td>
<td><strong>P3. Polarization Modeling</strong></td>
</tr>
<tr>
<td><strong>Tensors</strong></td>
<td><strong>P2. Fast and Scalable Tucker Decomposition</strong></td>
<td></td>
</tr>
</tbody>
</table>

* Duplicated
Proposed Work by Topics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Graphs</td>
<td>P1. Triangle Counting in Fully Dynamic Stream</td>
<td></td>
</tr>
<tr>
<td>P2. Fast and Scalable Tucker Decomposition</td>
<td></td>
<td>P3. Polarization Modeling</td>
</tr>
</tbody>
</table>

* Duplicated
P1: Problem Definition

• Given:
  ◦ a **fully dynamic** graph stream,
    ▪ i.e., list of edge **insertions** and edge **deletions**
  ◦ Memory budget $k$

• Estimate: the **counts of global and local triangles**

• To Minimize: estimation error
<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy</th>
<th>Handle Deletions?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Triest-FD</td>
<td>Lowest</td>
<td>Yes</td>
</tr>
<tr>
<td>MASCOT</td>
<td>Low</td>
<td>No</td>
</tr>
<tr>
<td>Triest-IMPR</td>
<td>High</td>
<td>No</td>
</tr>
<tr>
<td>WRS</td>
<td>Highest</td>
<td>No</td>
</tr>
<tr>
<td>Proposed</td>
<td>Highest</td>
<td>Yes</td>
</tr>
</tbody>
</table>
## Proposed Work by Topics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs</strong></td>
<td><strong>P1. Triangle Counting in Fully Dynamic Stream</strong></td>
<td><strong>P3. Polarization Modeling</strong></td>
</tr>
<tr>
<td><strong>Tensors</strong></td>
<td><strong>P2. Fast and Scalable Tucker Decomposition</strong></td>
<td></td>
</tr>
</tbody>
</table>

* Duplicated
P2: Problem Definition

- Tucker Decomposition (a.k.a High-order PCA)
  - **Given**: an $N$-order input tensor $X$
  - **Find**: $N$ factor matrices $A^{(1)} ... A^{(N)}$ & core-tensor $Y$
  - **To satisfy**:

\[ X \approx Y \]

\[ X \approx A^{(1)} A^{(2)} A^{(3)} \]
P2: Standard Algorithms

Input (large & sparse) → Intermediate Data (large & dense) → Output (small & dense)

- Materialized:
  - Input: 2GB
  - Intermediate Data: 400GB - 4TB (Scalability bottleneck)
  - Output: 2GB

- SVD
P2: Completed Work

• Our completed work [WSDM17]

P2: Proposed Work

- Proposed algorithm

**Input** (large & sparse)  
**Intermediate Data** (small & dense)  
**Output** (small & dense)

- Partially materialize intermediate data!
P2: Expected Performance Gain

- Which part of intermediate data should we materialize?
- Exploit skewed degree distributions!
# Proposed Work by Topics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P1. Triangle Counting in Fully Dynamic Stream</td>
<td>✔️</td>
<td></td>
<td>P3. Polarization Modeling</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Tensors</th>
<th>P2. Fast and Scalable Tucker Decomposition</th>
</tr>
</thead>
</table>

* Duplicated
P3. Polarization Modeling

- **Polarization** in social networks: division into contrasting groups

Use of marijuana should be: Legal   Illegal

OR

- change of beliefs
- change of edges

“How do people choose between *two ways of polarization*?”
P3. Problem Definition

• **Given**: time-evolving social network with nodes’ beliefs on controversial issues
  ◦ e.g., legalizing marijuana

• **Find**: actor-based model with a utility function
  ◦ depending on network features, beliefs, etc.

• **To best describe**: the polarization in data

• **Applications**:
  ◦ predict future edges
  ◦ predict the cascades of beliefs
## Proposed Work by Topics

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graphs</strong></td>
<td><strong>P1. Triangle Counting in Fully Dynamic Stream</strong></td>
<td><strong>P3. Polarization Modeling</strong></td>
</tr>
<tr>
<td><strong>Tensors</strong></td>
<td><strong>P2. Fast and Scalable Tucker Decomposition</strong></td>
<td></td>
</tr>
</tbody>
</table>

* Duplicated
Timeline

• Mar-May 2018
  ◦ **P1.** Triangle counting in fully dynamic graph streams

• Jun-Aug 2018
  ◦ **P3.** Polarization modeling

• Sep-Oct 2018
  ◦ **P2.** Fast and scalable tucker decomposition

• Nov 2018 –April 2019
  ◦ Thesis Writing & Job Application

• May 2019
  ◦ Defense
Roadmap

• Overview

• Completed Work
  ◦ T1. Structure Analysis
  ◦ T2. Anomaly Detection
  ◦ T3. Behavior Modeling

• Proposed Work

• Conclusion <<
Conclusion

• **Goal:**
  To Understand Large Dynamic Graphs and Tensors

• **Subtasks:**
  ◦ structure analysis
  ◦ anomaly detection
  ◦ behavior modeling

• **Approaches:**
  ◦ distributed or external-memory algorithms
  ◦ streaming algorithms based on sampling
  ◦ approximation algorithms
References (Completed work)


Thank You

• Papers, software, data: http://www.cs.cmu.edu/~kijungs/proposal/

• Email: kijungs@cs.cmu.edu

• Thanks to:
  ◦ Sponsors: NSF, [Other sponsors]
  ◦ Admins: [Names of admin team members]
  ◦ Collaborators: [List of collaborator names and photos]

Mining Large Dynamic Graphs and Tensors (by Kijung Shin)