Outlines

- TimeCrunch: Interpretable Dynamic Graph Summarization by Neil Shah et. al. (*KDD 2015*)
- From Micro to Macro: Uncovering and Predicting Information Cascading Process with Behavioral Dynamics by Linyun Yu (*Best Student paper award ICDM 2015*)
- Edge-Weighted Personalized PageRank: Breaking A Decade-Old Performance Barrier by Wenlei Xie et. al (*Best Student paper award KDD 2015*)
TimeCrunch: Interpretable Dynamic Graph Summarization

Neil Shah  
Carnegie Mellon University  
neilshah@cs.cmu.edu

Danai Koutra  
Carnegie Mellon University  
danai@cs.cmu.edu

Tianmin Zou  
Carnegie Mellon University  
tzou@andrew.cmu.edu

Brian Gallagher  
Lawrence Livermore Lab  
bgallagher@llnl.gov

Christos Faloutsos  
Carnegie Mellon University  
christos@cs.cmu.edu
Problem (INFORMAL). Given a dynamic graph, find a set of possibly overlapping temporal subgraphs to concisely describe the given dynamic graph in a scalable fashion.
Main contributions

1. **Problem Formulation**: They show how to define the problem of dynamic graph understanding in a compression context.

2. **Effective and Scalable Algorithm**: They develop TIMECRUNCH, a fast algorithm for dynamic graph summarization.

3. **Practical Discoveries**: They evaluate TIMECRUNCH on multiple real, dynamic graphs and show *quantitative* and *qualitative* results.
What is MDL?
MDL is a "Model Selection" method.

\[ \min \quad L(M) + L(D|M) \]

OR

\[ \min \quad - \log p(M) - \log p(D|M) \]
We consider models $M \in \mathcal{M}$ to be composed of ordered lists of temporal graph structures with node, but not edge overlaps. Each $s \in M$ describes a certain region of the adjacency tensor $A$ in terms of the interconnectivity of its nodes.
PROBLEM 2 (MINIMUM DYNAMIC GRAPH DESCRIPTION). Given a dynamic graph $G$ with adjacency tensor $A$ and temporal phrase lexicon $\Phi$, find the smallest model $M$ which minimizes the total encoding length

$$L(G; M) = L(M) + L(E)$$

$$E = M \oplus A$$

$$\Phi = \Delta \times \Omega$$

$\Delta = \{o; r; p; f; c\}$ set of temporal signatures

$\Omega = \{st; fc; nc; bc; nb; ch\}$ set of static identifiers
Encoding the Model

\[ L(M) = L_N(|M| + 1) + \log_2 \left( \frac{|M| + |\Phi| - 1}{|\Phi - 1|} \right) + \sum_{s \in M} (-\log_2 P(v(s)|M) + L(c(s)) + L(u(s))) \]

\( u(s) \)  \( timesteps in which structure s appears \)
\( c(s) \)  \( connectivity \)
Encoding Connectivity and Temporal Presence

$L(c(c))$
- Stars
- Cliques (fc; nc)
- Bipartite Cores (bc; nb)
- Chains

$L(u(s))$
- Oneshot
- Ranged
- Periodic
- Flickering
- Constant
Encoding the Errors (in Connectivity)

\[ E = M \oplus A \]

\( E^+ \): The area of \( A \) which \( M \) models and \( M \) includes extraneous edges not present in the original graph.

\( E^- \): The area of \( A \) which \( M \) does not model and therefore does not describe.

\[
L(E^+) = \log_2(|E^+|) + ||E^+||_{\rho_1} + ||E^+||_{\rho_0}
\]

\[
L(E^-) = \log_2(|E^-|) + ||E^-||_{\rho_1} + ||E^-||_{\rho_0}
\]

In both cases, we encode the number of 1s in \( E^+ \) (or \( E^- \)), followed by the actual 1s and 0s using optimal prefix codes.
Encoding the Errors (in Temporal Presence)

\[ L(e_u(s)) = \sum_{k \in h(e_u(s))} \left( \log_2(k) + \log_2 c(k) + c(k) \rho_k \right) \]

\[ e_u(s) = u(s) - \tilde{u}(s) \]

\( h(e_u(s)) \) denotes the set of elements with unique magnitude in \( e_u(s) \)
\( c(k) \) denotes the count of element \( k \) in \( e_u(s) \)
\( \rho_k \) denotes the length of the optimal prefix code for \( k \)
Algorithm 1 TimeCrunch

1: Generating Candidate Static Structures: Generate static subgraphs for each $G_1 \cdots G_t$ using traditional static graph decomposition approaches.

2: Labeling Candidate Static Structures: Label each static subgraph as a static structure corresponding to the identifier $x \in \Omega$ which minimizes the local encoding cost.

3: Stitching Candidate Temporal Structures: Stitch the static structures from $G_1 \cdots G_t$ together to form temporal structures with coherent connectivity behavior and label them according to the the phrase $p \in \Phi$ which minimizes temporal presence encoding cost. Populate the candidate set $C$.

4: Composing the Summary: Compose a model $M$ of important, non-redundant temporal structures which summarize $G$ using the VANILLA, TOP-10, TOP-100 and STEPWISE heuristics. Choose $M$ associated with the heuristic that produces the smallest total encoding cost.
$\mathcal{F}$: set of static subgraphs over $G_1, \ldots, G_t$
we seek to find static subgraphs which have the same patterns of connectivity over one or more timesteps and stitch them together.
we formulate the problem of finding coherent temporal structures in $G$ as a clustering problem over $\mathcal{F}$.
two structures in the same cluster should have

- substantial overlap in the node-sets composing their respective subgraphs
- exactly the same, or similar (full and near clique, or full and near bipartite core) static structure identifiers.
Composing the Summary

Given the candidate set of temporal structures $C$, they next seek to find the model $M$ which best summarizes $G$.

**Local encoding benefit:** The ratio between the cost of encoding the given temporal structure as error and the cost of encoding it using the best phrase (local encoding cost).

- **VANILLA:** This is the baseline approach, in which our summary contains all the structures from the candidate set, or $M = C$.

- **TOP-K:** In this approach, $M$ consists of the top $k$ structures of $C$, sorted by local encoding benefit.

- **STEPWISE:** This approach involves considering each structure of $C$, sorted by local encoding benefit, and adding it to $M$ if the global encoding cost decreases. If adding the structure to $M$ increases the global encoding cost, the structure is discarded as redundant or not worthwhile for summarization purposes.
<table>
<thead>
<tr>
<th>Graph</th>
<th>Nodes</th>
<th>Edges</th>
<th>Timesteps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron [24]</td>
<td>151</td>
<td>20 thousand</td>
<td>163 weeks</td>
</tr>
<tr>
<td>Yahoo-IM [28]</td>
<td>100 thousand</td>
<td>2.1 million</td>
<td>4 weeks</td>
</tr>
<tr>
<td>Honeynet</td>
<td>372 thousand</td>
<td>7.1 million</td>
<td>32 days</td>
</tr>
<tr>
<td>DBLP [1]</td>
<td>1.3 million</td>
<td>15 million</td>
<td>25 years</td>
</tr>
<tr>
<td>Phonecall</td>
<td>6.3 million</td>
<td>36.3 million</td>
<td>31 days</td>
</tr>
</tbody>
</table>
They used TIMECRUNCH to summarize each of the real-world dynamic graphs from dataset's table and report the resulting encoding costs. Specifically,

<table>
<thead>
<tr>
<th>Graph</th>
<th>Original (bits)</th>
<th>Vanilla</th>
<th>Top-10</th>
<th>Top-100</th>
<th>Stepwise</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enron</td>
<td>86,102</td>
<td>89% (563)</td>
<td>88%</td>
<td>81%</td>
<td>78% (130)</td>
</tr>
<tr>
<td>Yahoo-IM</td>
<td>16,173,388</td>
<td>97% (5000)</td>
<td>99%</td>
<td>98%</td>
<td>93% (1523)</td>
</tr>
<tr>
<td>Honeynet</td>
<td>72,081,235</td>
<td>82% (5000)</td>
<td>96%</td>
<td>89%</td>
<td>81% (3740)</td>
</tr>
<tr>
<td>DBLP</td>
<td>167,831,004</td>
<td>97% (5000)</td>
<td>99%</td>
<td>99%</td>
<td>96% (1627)</td>
</tr>
<tr>
<td>Phonecall</td>
<td>478,377,701</td>
<td>100% (1000)</td>
<td>100%</td>
<td>99%</td>
<td>98% (370)</td>
</tr>
</tbody>
</table>
Encoding Cost vs. Model Size

Vanilla encoding
Stepwise encoding

Encoding Cost (in bits)

Number of Structures in Model
Qualitative Analysis

(a) 8 employees of the Enron legal team forming a flickering near clique
(b) 10 employees of the Enron legal team forming a flickering star with the boss as the hub
(c) 40 users in Yahoo–IM forming a constant near clique with 55% density over the observed 4 weeks

(d) 82 users in Yahoo–IM forming a constant star
(c) 589 honeypot machines were attacked on Honeynet over 2 weeks, forming a ranged star
(f) 43 authors that publish together in biotechnology journals forming a ranged near clique on DBLP

(g) 82 authors forming a ranged near clique
(h) 111 callers in Phonecall forming a periodic on DBLP, with burgeoning collaboration from star appearing strongly on odd numbered days, especially Dec. 25 and 31
(i) 792 callers in Phonecall forming a oneshot near bipartite core appearing strongly on Dec. 31
From Micro to Macro: Uncovering and Predicting Information Cascading Process with Behavioral Dynamics

Linyun Yu\textsuperscript{1}, Peng Cui\textsuperscript{1}, Fei Wang\textsuperscript{2}, Chaoming Song\textsuperscript{3}, Shiqiang Yang\textsuperscript{1}

\textsuperscript{1}Department of Computer Science and Technology, Tsinghua University
\textsuperscript{2}Department of Computer Science and Engineering, University of Connecticut
\textsuperscript{3}Department of Physics, University of Miami

yuly12@mail.tsinghua.edu.cn, cuip@tsinghua.edu.cn, fei_wang@uconn.edu
chaoming.song@gmail.com, yangshq@tsinghua.edu.cn
The ultimate purpose of this paper is to predict the cascading process.

- Is the cascading process predictable?
- Given the early stage of an information cascade, can we predict its cumulative cascade size of any later time?
Cascade Prediction: Given the early stage of a cascade $C_t$, predict the cascade size $\text{size}(C_{t'})$ with $t' > t$.

\[
\begin{align*}
C &= \{u_1, u_2, \ldots, u_m\} \\
\tau(u_i) &\leq \tau(u_{i+1}) \\
C_t &= \{u_i | \tau(u_i) \leq t\} \\
\text{size}(C_t) &= |C_t|
\end{align*}
\]
A fundamental way to address this problem is to look into the *micro* mechanism of cascading processes. Intuitively, an information cascading process can be decomposed into multiple local (one-hop) subcascades.
(a) Cascading Process

(b) Partially observed cascade at $t$
the behavioral dynamics of a user capture the changing process of the cumulative number of his/her followers retweet a post after the user retweeting the post.
Survival analysis is a branch of statistics that deals with analysis of time duration until one or more events happen, such as death in biological organisms and failure in mechanical systems.

\[ S(t) = Pr\{\tau_0 \geq t\} = \int_t^\infty f(t) \]

\[ \lambda(t) = \lim_{dt \to 0} \frac{Pr(t \leq \tau_0 < t + dt|\tau_0 \geq t)}{dt} = \frac{f(t)}{S(t)} \]
NEtworked WEibull Regression Model

\[ f_i(t) = \frac{k_i}{\lambda_i} \left( \frac{t}{\lambda_i} \right)^{k_i-1} \exp\left( \frac{t}{\lambda_i} \right)^{k_i} \]  

(3)

\[ S_i(t) = \exp\left( \frac{t}{\lambda_i} \right)^{k_i} \]  

(4)

\[ h_i(t) = \frac{k_i}{\lambda_i} \left( \frac{t}{\lambda_i} \right)^{k_i-1} \]  

(5)

\( \lambda_i > 0 \): Scale parameter. \( k_i > 0 \): shape parameter.

\[ F(\lambda, k, \beta, \gamma) = G_1(\lambda, k) + \mu G_2(\beta, \lambda) + \eta G_3(\gamma, k) \]

\[ G_1(\lambda, k) = -\log L(\lambda, k) \]

\[ G_2(\lambda, \beta) = \frac{1}{2N} \| \log \lambda - \log X \cdot \beta \|^2 + \alpha_\beta \| \beta \|_1 \]

\[ G_3(k, \gamma) = \frac{1}{2N} \| \log k - \log X \cdot \gamma \|^2 + \alpha_\gamma \| \gamma \|_1 \]
The parameters of the user’s behavioral dynamics should be correlated with the behavioral features of his/her followers

\[
\log \lambda_i = \log x_i \ast \beta \\
\log k_i = \log x_i \ast \gamma
\]

\(\beta\) and \(\gamma\) are \(r\)-dimensional parameter vector for \(\lambda\) and \(k\). \(x_i\) is \(r\)-dimensional feature vector for user \(i\).
<table>
<thead>
<tr>
<th>Behavioral features</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><em>inflow_rate</em></td>
<td>the number of the posts user received in a certain period.</td>
</tr>
<tr>
<td><em>outflow_rate</em></td>
<td>the number of the posts user sent in a certain period.</td>
</tr>
<tr>
<td><em>follower_avg</em>_inflow_rate</td>
<td>average inflow rate of fans to the user, or ( \frac{\sum_i \text{retweet}(i) \cdot \text{in_flow}(i)}{\sum_i \text{retweet}(i)} ) where ( i ) is the fans to the user(and the same as following).</td>
</tr>
<tr>
<td><em>follower_avg</em>_retweet_rate</td>
<td>average retweet rate of fans to the user, or ( \frac{\sum_i \text{retweet}(i) \cdot \text{retweet_rate}(i)}{\sum_i \text{retweet}(i)} ).</td>
</tr>
<tr>
<td>Structural features</td>
<td></td>
</tr>
<tr>
<td><em>follower_number</em></td>
<td>number of the followers to the user.</td>
</tr>
<tr>
<td><em>follow_number</em></td>
<td>number of users this user follows.</td>
</tr>
</tbody>
</table>
Algorithm 1 Basic Model

Input:
Set of users $U$ involved in the cascade $C$ before time $t_{limit}$,
survival functions of users $S_{u_j}(t)$, predicting time $t_e$;

Output:
Size of cascade $size(C_{t_e})$;

\begin{algorithmic}
\STATE for all user $u_i \in U$ do
\STATE creates a subcascade process with $replvnum(u_i) = 0$
\STATE if $u_i$ is not root node then
\STATE $replvnum(rp(u_i)) = replvnum(rp(u_i)) + 1$
\STATE end if
\STATE end for
\STATE $sum = 1$
\STATE for all user $u_i \in U$ do
\STATE $deathrate(u_i) = \max \left( 1 - S_{u_i}(t_{limit} - t(u_i)), \frac{1}{|V|} \right)$
\STATE $fdrate(u_i) = \max \left( 1 - S_{u_i}(t_e - t(u_i)), \frac{1}{|V|} \right)$
\STATE $sum = sum + \frac{replvnum(u_i) \cdot fdrate(u_i)}{deathrate(u_i)}$
\STATE end for
\STATE return $size(C_{t_e}) = sum$
\end{algorithmic}
For a subcascade generated by $u_i$, the estimation of the size will always be zero if there is no user involved into it, which means we can ignore the calculation.

If we do not re-estimate the final number of a subcascade (when there is no new user involved into it), the temporal size counter $\text{replynum}(u_i)$ and final death rate $\text{edrate}(u_i)$ will not change but the death rate $\text{deathrate}_{u_i}(t)$ will increase over time.
<table>
<thead>
<tr>
<th>Method</th>
<th>Base Model</th>
<th>Improved Model ($\delta = 0.1$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size $\geq 20$</td>
<td>$8.47 \times 10^5$ s</td>
<td>$10.73$ s</td>
</tr>
<tr>
<td>Size $\geq 50$</td>
<td>$7.61 \times 10^5$ s</td>
<td>$8.62$ s</td>
</tr>
<tr>
<td>Size $\geq 100$</td>
<td>$6.65 \times 10^5$ s</td>
<td>$7.09$ s</td>
</tr>
<tr>
<td>Size $\geq 500$</td>
<td>$4.35 \times 10^5$ s</td>
<td>$4.33$ s</td>
</tr>
<tr>
<td>Size $\geq 1000$</td>
<td>$3.4 \times 10^5$ s</td>
<td>$3.30$ s</td>
</tr>
</tbody>
</table>
Cascade Size Prediction

- cox
- exp
- rayleigh
- NEWER
- log-linear

Cascades with size at least 100
Cascades with size at least 300
Cascades with size at least 600

RMSLE vs Observation Number
Outbreak Time Prediction

The graphs illustrate the comparison of different models for outbreak time prediction. The left graph shows the Relative Mean Square Logarithmic Error (RMSLE) against the observation number, where "cox", "exp", "rayleigh", and "NEWER" are the models being compared. The right graph shows the 0.2-precision against the observation number, with a similar comparison of models.
Cascading Process Prediction

\[ \delta \text{ 0.2-Precision} \]

\[ \text{Early_Stage_Percentage} \]
Out-of-sample Prediction

Graphs showing the RMSLE (Root Mean Squared Logarithmic Error) for different models (NEWER, wbl, cox) under three conditions:
- Cascades with size at least 300
- Cascades with size at least 600
- Cascades with size at least 1000

The graphs compare the performance of the models as the observation number increases, with each model showing a different trend and rate of improvement.
Edge-Weighted Personalized PageRank: Breaking A Decade-Old Performance Barrier

Wenlei Xie, David Bindel, Alan Demers, Johannes Gehrke

Cornell University
Ithaca, NY

{wenleix, bindel, ademers, johannes}@cs.cornell.edu
In this paper, we introduce the first truly fast method to compute $x(w)$ in the edge-weighted personalized PageRank case.