Similarity Study on Large Dynamic Information Networks

Cuiping Li,
Information School,
Renmin University of China
“Networked Data”, Everywhere!

- **Technological and communication networks**
  - power-grid, road networks
- **Biological and genetic networks**
  - food-web, protein networks
- **Social networks**
  - collaboration networks, friendships;
  - co-citation, blog crosspostings
Network Analysis: Big Picture

1. Whole graph Level
   - Macro properties (Laws, generators)
   - Summary/Visualization,
   - Index

2. Sub-graph Level
   - Frequent Pattern Mining
   - Clustering (Community/group detection)
   - Connected Sub-graph, Central Piece
   - Pattern Match

3. Node or Link Level
   - Ranking
   - Proximity/Similarity
   - Node Classification
   - Outlier Detection (Abnormal nodes/links)

We are here!
Node Proximity/Similarity: Why?

- Link prediction [Liben-Nowell+], [Tong+]
- Ranking [Haveliwala], [Chakrabarti+]
- Name Disambiguation [Minkov+ Sigir06]
- Image caption [Pan+]
- Neighborhood Formulation [Sun+]
- Connected sub-graph [Faloutsos+], [Tong+], [Koren+]
- Pattern match [Tong+]
- Collaborative Filtering [Fouss+]
- Many more...
Node Similarity: Related Work (1)

- **SimRank:**

- **Random Walk with Restart:**
  - NeighborHood Formation and Anomaly Detection in Bipartite, Jimeng Sun, Huiming Qu, Deepayan Chakrabarti, Christos Faloutsos, ICDM, 2005

- **Other:**
Node Similarity: Related Work(2)

- **Optimization on SimRank**
  - Scaling link-base similarity search, D.Fogaras, B. Racz, WWW’05: (Approximate)
  - Accuracy Estimate and Optimization Techniques for SimRank Computation, Dmitry Lizorkin, Pavel Velikhov, Maxim Grinev, Denis Turdakov, VLDB’08

- **Domain-Integrated of SimRank:**
  - Simrank++: Query Rewriting through Link Analysis of the Click Graph, Loannis Antonellis, Hector Garcia-Molina, Chi-Chao Chang, VLDB’08. (keywords, ads)

- **Clustering using SimRank:**
  - ReCom: Reinforcement Clustering of multi-type interrelated data objects, J. Wang, H.J. Zeng, Z. Chen, H.J. LU, L. Tao, SIGIR’03:
  - LinkCLus: Efficient Clustering via Heterogeneous Semantic Links, Xiaoxin Yin, Jiawei Han, Philip Yu, VLDB’06:
Existing Research: Limitations

• **Not Dynamic**
  – Static Algorithm
    • Iterative
  – Challenges of Dynamic Network
    • Re-computation even one node or edge changes
  – The only existing work only considered bipartite graph
    • Proximity tracking on time-evolving bipartite graphs, H. Tong, S. Papadimitriou, P. S. Yu, and C. Faloutsos, SDM, 2008. (Best Paper)

• **Our Solution**
  • Non-iterative, low rank
  • Incremental Computation
Outline

✓ Background and Motivation

✓ Solution for Dynamic Networks
  □ Non-iterative, low rank method
  □ Iterative aggregation method

✓ Parallel Computation for Large Networks

✓ Conclusion and future work
Node Similarity Measures

- **Text- or content-based:**
  - bag of words, vector space

- **Link- or structure-based:**
  - relationships of links
  - E.g. Random Walk With Restart, SimRank

- **SimRank formula (iterative)**
  \[
  S(a, b) = \begin{cases}
  \frac{c}{|I(a)||I(b)|} & \sum_{i=1}^{I(a)} \sum_{j=1}^{I(b)} S(I_i(a), I_j(b)) \\
  1 & , a = b
  \end{cases}
  \]

- **Or, in matrix form**
  \[
  S = cW'SW + (1 - c)I
  \]

**Intuition:**
Two objects are similar if they are referenced by similar objects
SimRank and Sylvester Equation

• Fist glance at SimRank formula
  - It is Iterative. Has no chance to be computed incrementally

• Key Observation
  - SimRank iteration formula has the same form as the well-known Sylvester Equations

\[
\text{Sylvester Equation} \quad \mathbf{X} = \mathbf{AXB} + \mathbf{C}
\]

\[
\text{SimRank Formula} \quad S = cW'SW + (1-c)I
\]
How to solve Sylvester Equations?

- **Vec Operator**
  - Flatten an $n \times n$ matrix into an $n^2 \times 1$ vector
  - Stack columns on top of each other, from left to right

$$
vec \begin{pmatrix}
a_1 \\
a_2 \\
a_3 \\
a_4 \\
\end{pmatrix} = \begin{pmatrix}
a_1 \\
a_2 \\
a_3 \\
a_4 \\
\end{pmatrix}
$$

- **Kronecker Product**
  - Each element of $A$ is multiplied with full matrix $B$

$$
A \otimes B = \begin{pmatrix}
a_{11}B & \cdots & a_{1n}B \\
\vdots & \ddots & \vdots \\
a_{n1}B & \cdots & a_{nn}B \\
\end{pmatrix}
$$
Non-iterative SimRank

\[ \text{vec}(X) = (1-c) \left( I - cW' \otimes W' \right)^{-1} \text{vec}(I) \]

- **Advantages**
  - It can be solved approximately
  - It can be computed pair-wisely
  - It can be computed incrementally
Low Rank Approximation

- Low-rank SVD of $W'$

\[
W' \approx U \Sigma V^T
\]

- Choose $k$ largest singular values from $\Sigma$
- Reserve corresponding singular vectors in $U$ and $V$

- Larger value of $k$
  - Higher computation cost
  - Higher accuracy

Error Bound

\[
\|S - S_a\| = c(1-c) \sum_{i=k+1}^{n} \frac{\lambda_1 \lambda_i}{1 - c \lambda_1 \lambda_i}
\]

$\lambda_i$ is the $i$-th largest Eigen value of $W'$
when $i > k$, $\lambda_i$ is small
Incremental Update for Dynamic Graphs

Need to update: $U, \Sigma, V$

\[
\begin{align*}
U^t &= U^{t-1} U_C \\
V^t &= V_C V^{t-1} \\
\Sigma^t &= \Sigma_C \\
\end{align*}
\]

\[
C = \Sigma^{t-1} + (U^{t-1})' AB (V^{t-1})'
\]

if $\Delta \tilde{W} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$, then $AB = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$
Experimental Result on DBLP

- We extract the 10-year (from 1998 to 2007) author-paper-term information from the whole DBLP data set.

- Every two publication years form a time step.

- For each time step, we construct an information network.
Experimental Result on DBLP

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>represent</td>
<td>extension</td>
<td>current</td>
<td>paper</td>
<td>pipelined</td>
</tr>
<tr>
<td>xml</td>
<td>tradeoff</td>
<td>panel</td>
<td>conference</td>
<td>work</td>
</tr>
<tr>
<td>change</td>
<td>trigger</td>
<td>manager</td>
<td>panel</td>
<td>uncertainty</td>
</tr>
<tr>
<td>semistructured</td>
<td>issue</td>
<td>context</td>
<td>review</td>
<td>uncertain</td>
</tr>
<tr>
<td>optimization</td>
<td>precision</td>
<td>personalize</td>
<td>pipelined</td>
<td>clean</td>
</tr>
<tr>
<td>rewrite</td>
<td>practical</td>
<td>stream</td>
<td>property</td>
<td>replicate</td>
</tr>
<tr>
<td>impact</td>
<td>replicate</td>
<td>structural</td>
<td>limit</td>
<td>integrity</td>
</tr>
<tr>
<td>query</td>
<td>warehouse</td>
<td>source</td>
<td>memory</td>
<td>skew</td>
</tr>
<tr>
<td>power</td>
<td>incremental</td>
<td>continuous</td>
<td>route</td>
<td>cad</td>
</tr>
<tr>
<td>protein</td>
<td>transformation</td>
<td>minimization</td>
<td>plan</td>
<td>service</td>
</tr>
</tbody>
</table>

Top-10 Most Similar Terms for 'Prof. Jennifer Widom' up to Each Time Step
However,

- The non-iterative, low rank method can only handle link-update problem
  - Link-Updating: assume the node number of a graph is fixed.
    - i.e. No nodes are added or deleted when the graph is evolving
  - Node-Updating (more difficult): the node number can change.
  - Link-updating is a special case of node-updating

- Need re-computation when nodes change
Outline

✓ Background and Motivation

✓ Solution for Dynamic Networks
  - Non-iterative, low rank method
  - Iterative aggregation method

✓ Parallel Computation for Large Networks

✓ Conclusion and future work
Iterative aggregation (IA) method (1)

- **Based on Markov Chain**
  - Distribute compute PageRank [ZYI05]
  - Incremental update PageRank [LM04]

- **Use IA method to compute SimRank**
  - Step 1: Transform SimRank to Markov Chain
  - Step 2: Global Iteration + Local Iteration

\[ \begin{align*}
\phi^T &= (\phi_1, \phi_1, \ldots, \phi_q) \\
\pi^T &= (\pi_1, \pi_1, \ldots, \pi_n) \\
\alpha^T &= (\alpha_1, \alpha_1, \ldots, \alpha_{g+1}) \\
\end{align*} \]
Iterative aggregation (IA) method (2)

- Transform SimRank into Markov Chain
  \[ S = cW^T SW + (1 - c)I \]

- Vectorization (vec)

- Kronecker product (⊗)
  \[ vec(S)^T = c(vec(S)^T (W \otimes W)) + (1 - c)vec(I)^T \]

- Let \( \pi = vec(S) \), \( Q = c(W \otimes W) \), \( \omega = (1 - c)vec(I) \)

- First order Markov Chain
  \[ \pi^T = \pi^T Q + \omega \]
Iterative aggregation (IA) method (3)

- Partition nodes into groups
  - $V = V_1 \cup V_2 \cup V_3 \cup ... \cup V_p$
  - $S_{i,j}$: SimRank scores between nodes in $V_i$ & $V_j$
Node-Updating

Step 1: Partition nodes further
- $V_u$: affected by update (including new nodes)
- $V_o$: unaffected by update

Step 2: Initialize SimRank matrix
- If $V_i$ and $V_j$ are in $V_o$, set to old score $S_{i,j}$
- else, set to $I_{i,j}$

Step 3: Different region, different method
- $V_u$: use iterative aggregation method
- $V_o$: use naïve method
Iterative aggregation (IA) method (4)

\[ S^l_{i,j} = c \]

\[ S^{l-1}_{i,j} \]

\[ S^k_{i,j} = c \]

\[ H_{i,j} \]

\[ S^{k-1}_{0,0} \]
\[ S^{k-1}_{0,1} \]
\[ S^{k-1}_{0,2} \]

\[ S^{k-1}_{1,0} \]
\[ S^{k-1}_{1,1} \]
\[ S^{k-1}_{1,2} \]

\[ S^{k-1}_{2,0} \]
\[ S^{k-1}_{2,1} \]
\[ S^{k-1}_{2,2} \]

\[ +(1-c) \]

\[ I_{i,j} \]

\[ \text{IA-SimRank: Local Iteration} \]

Extract local part:

\[ cW_{i,i}^T \tilde{S}^{l-1}_{i,j}W_{j,j} \]

Compress remain part:

\[ H_{i,j} = c \left( \sum_{u=1}^{p} \sum_{v=1}^{p} W_{u,i}^T S^{k-1}_{u,v} W_{v,j} - W_{i,i}^T S^{k-1}_{i,j} W_{j,j} \right) + (1-c)I_{i,j} \]

Local iterative step:

\[ \tilde{S}^{l}_{i,j} = cW_{i,i}^T \tilde{S}^{l-1}_{i,j}W_{j,j} + H_{i,j} \]
Outline

✓ Background and Motivation

✓ Solution for Dynamic Networks
  □ Non-iterative, low rank method
  □ Iterative aggregation method

✓ Parallel Computation for Large Networks

✓ Conclusion and future work
GPU as A Co-Processor of CPU

GPU Stream Processors

Parallel Cache

CPU (Device)

CPU (Host)
GPU & CPU

Advan 1: More processors
CPU: 2-16
GPU: hundreds

Advan 2: Higher bandwidth between processors & memory
CPU: <50GBps
GPU: 141GBps

Limit 2: Relative low bandwidth between main memory & device memory
Max: 8GBps, Avg: <5GBps
GPU Matrix Operators

Step 1: Load input matrices into device memory;
Step 2: Launch proper number of GPU threads to accomplish computation task;
Step 3: Transfer result matrix back to main memory;
Experiments

- Different parallel hardware platform

<table>
<thead>
<tr>
<th></th>
<th>CPU</th>
<th>Main Memory</th>
<th>GPU</th>
<th>Device Memory</th>
<th>Num of CPU threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPUS</td>
<td>P4 3.0GHz</td>
<td>1GB</td>
<td>-</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>CPUQ</td>
<td>Core 2 2.66GHzx4</td>
<td>4GB</td>
<td>-</td>
<td>-</td>
<td>4</td>
</tr>
<tr>
<td>GPU</td>
<td>P4 3.0GHz</td>
<td>1GB</td>
<td>9600GT</td>
<td>256MB</td>
<td>1</td>
</tr>
</tbody>
</table>
Experiments

- Different parallel hardware platform

(a) Avg step time for Iteration (Ite)
(b) Avg step time for IADC
Experiments

- 3 category graphs from 3 English Wikipedia dumps:
  - 2009-8-16 (wiki0816), 3,027,633 nodes, 1,104,571 edges
  - 2009-8-27 (wiki0827), 3,042,063 nodes, 1,112,619 edges
  - 2009-9-09 (wiki0909), 3,055,136 nodes, 1,116,820 edges

<table>
<thead>
<tr>
<th></th>
<th>Ite</th>
<th>IADC</th>
<th>IADU</th>
</tr>
</thead>
<tbody>
<tr>
<td>wiki0816</td>
<td>$k = 5$</td>
<td>1.222E-4</td>
<td>9.791E-5</td>
</tr>
<tr>
<td></td>
<td>the avg step time</td>
<td>2313s</td>
<td>2944s</td>
</tr>
<tr>
<td>wiki0827</td>
<td>$k = 5$</td>
<td>1.012E-4</td>
<td>9.367E-5</td>
</tr>
<tr>
<td></td>
<td>the avg step time</td>
<td>2365s</td>
<td>3077s</td>
</tr>
<tr>
<td>wiki0909</td>
<td>$k = 5$</td>
<td>1.290E-4</td>
<td>9.573E-5</td>
</tr>
<tr>
<td></td>
<td>the avg step time</td>
<td>2389s</td>
<td>3146s</td>
</tr>
</tbody>
</table>
Conclusion

- **Fast Similarity Computation over Large Dynamic Networks**
  - Non-iterative, low rank method for Dynamic Networks
  - Iterative aggregation method for Dynamic networks
  - **GPU Parallel Computation for Large Networks**

- **Ongoing work**
  - Explore other parallel paradigm to deal with extremely large data
  - Incorporate the other information such as node attributes into the similarity measure definition
  - Apply similarity to kNN query, link prediction, Collaborative Filtering...
Reference

1. Christos Faloutsos, Hanghang Tong, Large Graph Mining: Patterns, Tools and Case Studies, CIKM'08, Tutorials

2. Jiawei Han, Yizhou Sun, Xifeng Yan, and Philip S. Yu, Mining Heterogeneous Information Networks, KDD’10, Tutorials

3. Michael Mahoney, Geometric Tools for Graph Mining of Large Social and Information Networks, KDD’10, Tutorials

4. Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, Tianyi Wu, "Fast Computation of SimRank for Static and Dynamic Information Networks", EDBT'10, Switzerland

5. Guoming He, Haijun Feng, Cuiping Li and Hong Chen, Parallel SimRank Computation on Large Graphs with Iterative Aggregation, KDD’10, Washington