Graphs / Networks

Basics, how to build & store graphs, laws, etc.
Centrality, and algorithms you should know

Duen Horng (Polo) Chau
Assistant Professor
Associate Director, MS Analytics
Georgia Tech

Partly based on materials by Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos, Parishit Ram (GT PhD alum; SkyTree), Alex Gray
Internet
50 Billion Web Pages
Facebook
1.2 Billion Users

Modified from Marc_Smith, flickr
Many More

Twitter
Who-follows-whom (288 million users)

Amazon
Who-buys-what (120 million users)

AT&T Cellphone Network
Who-calls-whom (100 million users)

Protein-protein interactions
200 million possible interactions in human genome

# Large Graphs I Analyzed

<table>
<thead>
<tr>
<th>Graph</th>
<th>Nodes</th>
<th>Edges</th>
</tr>
</thead>
<tbody>
<tr>
<td>YahooWeb</td>
<td>1.4 Billion</td>
<td>6 Billion</td>
</tr>
<tr>
<td>Symantec Machine-File Graph</td>
<td>1 Billion</td>
<td>37 Billion</td>
</tr>
<tr>
<td>Twitter</td>
<td>104 Million</td>
<td>3.7 Billion</td>
</tr>
<tr>
<td>Phone call network</td>
<td>30 Million</td>
<td>260 Million</td>
</tr>
</tbody>
</table>
How to **represent** a graph?

Conceptually.

Visually.

Programmatically.
How to **Represent** a Graph?

Visually

Adjacency matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Adjacency list

1: 2, 3
2: 4
3: 2

**Edge list**

- most common distribution format
- sometimes **painful** to parse when edges/nodes have many columns (some are text with double/single quotes, some are integers, some decimals, …)
How to **Represent** a Graph?

**Visually**

**Adjacency matrix**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>3</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Adjacency list**

1: 2, 3
2: 4
3: 2

**Source node**

**Target node**

**Edge list**

1, 2, 1
1, 3, 3
2, 4, 2
3, 2, 1

Each node is often identified by a numeric ID. Why?
Assigning an ID to a node

- Use a “map” (Java) / “dictionary” (Python) / SQLite
- Same concept: given an entity/node (e.g., “Tom”) not seen before, assign a number to it
- Example of using SQLite to map names to IDs

<table>
<thead>
<tr>
<th>rowid</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
</tr>
<tr>
<td>2</td>
<td>Sandy</td>
</tr>
<tr>
<td>3</td>
<td>Richard</td>
</tr>
<tr>
<td>4</td>
<td>Polo</td>
</tr>
</tbody>
</table>

Hidden column; SQLite automatically created for you
How to use the node IDs?

Create an index for “name”. Then write a “join” query.

<table>
<thead>
<tr>
<th>rowid</th>
<th>name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tom</td>
</tr>
<tr>
<td>2</td>
<td>Sandy</td>
</tr>
<tr>
<td>3</td>
<td>Richard</td>
</tr>
<tr>
<td>4</td>
<td>Polo</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>source</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tom</td>
<td>Sandy</td>
</tr>
<tr>
<td>Polo</td>
<td>Richard</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>source</th>
<th>target</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>
How to store “large” graphs?
How large is “large”?

What do you think?

• In what units? Thousands? Millions?

How do you measure a graph’s size?

• By ...

(Hint: highly subjective. And domain specific.)
Storing large graphs...

On your laptop computer

- SQLite
- Neo4j (GPL license)
  http://neo4j.com/licensing/

On a server

- MySQL, PostgreSQL, etc.
- Neo4j(?)
Storing large graphs...

With a cluster (more details a few lectures down)

- **Hadoop** (generic framework)
- **HBase/Titan** (inspired by Google’s BigTable), S2Graph
- **Hama**, inspired by Google’s Pregel
- **FlockDB**, by Twitter
- **Dato** (previously GraphLab)
- **Comparison of “graph databases”** (a little outdated)
Storing large graphs on your computer

I like to use **SQLite**. Why? **Good enough for my use.**

- Easily handle up to **gigabytes**
  - Roughly **tens of millions** of nodes/edges (perhaps up to billions?). Very good! For today’s standard.
- Very easy to maintain: **one** cross-platform file
- Has programming wrappers in numerous languages
  - C++, Java (Andriod), Python, Objective C (iOS),...
- Queries are so easy!
  - e.g., find all nodes’ degrees = 1 SQL statement
- Bonus: SQLite even supports full-text search
- Offline application support (iPad)
SQLite graph database schema

Simplest schema:

edges(source_id, target_id)

More sophisticated (flexible; lets you store more things):

```sql
CREATE TABLE nodes (  
id INTEGER PRIMARY KEY,
type INTEGER DEFAULT 0,
name VARCHAR DEFAULT ''
);

CREATE TABLE edges (  
source_id INTEGER,
target_id INTEGER,
type INTEGER DEFAULT 0,
weight FLOAT DEFAULT 1,
timestamp INTEGER DEFAULT 0,
PRIMARY KEY(source_id, target_id, timestamp))
```
Full-Text Search (FTS) on SQLite

http://www.sqlite.org/fts3.html

Very simple. Built-in. Only needs 3 lines of commands.

- **Create** FTS table (index)
  
  ```sql
  CREATE VIRTUAL TABLE critics_consensus USING fts4(consensus);
  ```

- **Insert** text into FTS table
  
  ```sql
  INSERT INTO critics_consensus SELECT critics_consensus FROM movies;
  ```

- **Query** using the “match” keyword
  
  ```sql
  SELECT * FROM critics_consensus WHERE consensus MATCH 'funny OR horror';
  ```

Originally developed by Google engineers
I have a graph dataset. Now what?

Analyze it! Do “data mining” or “graph mining”.

How does it “look like”? Visualize it if it’s small.

Does it follow any expected patterns?
Or does it *not* follow some patterns (outliers)?

- Why does this matter?

- If we know the patterns (models), we can do prediction, recommendation, etc.
  e.g., is Alice going to “friend” Bob on Facebook?
  People often buy beer and diapers together.

- Outliers often give us new insights
  e.g., telemarketer’s friends don’t know each other
Finding patterns & outliers in graphs

Outlier/Anomaly detection (will be covered later)

• To spot them, we need to patterns first
• Anomalies = things that do not fit the patterns

To effectively do this, we need large datasets

• patterns and anomalies don’t show up well in small datasets
Are real graphs random?

Random graph (Erdos-Renyi)
100 nodes, avg degree = 2

No obvious patterns

Generated with pajek
http://vlado.fmf.uni-lj.si/pub/networks/pajek/
Laws and patterns

• Are real graphs random?

• A: NO!!!
  • Diameter (longest shortest path)
  • in- and out- degree distributions
  • other (surprising) patterns

• So, let’s look at the data
Power Law in Degree Distribution

- Faloutsos, Faloutsos, Faloutsos [SIGCOMM99] Seminal paper. Must read!

Zipf’s law: http://en.wikipedia.org/wiki/Zipf%27s_law
Power Law in Degree Distribution

- Faloutsos, Faloutsos, Faloutsos [SIGCOMM99]
  Seminal paper. Must read!

Zipf’s law: http://en.wikipedia.org/wiki/Zipf%27s_law
Power Law in Eigenvalues of Adjacency Matrix

Eigenvalue

Rank of decreasing eigenvalue

Eigen exponent = slope = -0.48
How about graphs from other domains?
More Power Laws

• Web hit counts
  [Alan L. Montgomery and Christos Faloutsos]
epinions.com

- who-trusts-whom
  [Richardson + Domingos, KDD 2001]

(out) degree

trusts-2000-people user
And numerous more

- # of sexual contacts
- Duration of downloads [Bestavros+]
- Duration of UNIX jobs
- File sizes
  - …
Any other ‘laws’?

• Yes!
• Small diameter (~ constant!) –
  • six degrees of separation / ‘Kevin Bacon’
  • small worlds [Watts and Strogatz]
Problem: Time evolution

- Jure Leskovec (CMU -> Stanford)
- Jon Kleinberg (Cornell)
- Christos Faloutsos (CMU)
Evolution of the Diameter

• Prior work on Power Law graphs hints at slowly growing diameter:
  • diameter \sim O(\log N)
  • diameter \sim O(\log \log N)

• What is happening in real data?
Evolution of the Diameter

• Prior work on Power Law graphs hints at slowly growing diameter:
  • diameter \sim O(\log N)
  • diameter \sim O(\log \log N)

• What is happening in real data?
• Diameter shrinks over time
Diameter – Patents Network

- Patent citation network
- 25 years of data
- @1999
  - 2.9 M nodes
  - 16.5 M edges

![Graph showing the effective diameter over time](image)

- **Effective diameter**
- **Time [years]**

Legend:
- Full graph
- Post ’85 subgraph
- Post ’85 subgraph, no past
Temporal Evolution of the Graphs

• N(t) … nodes at time t
• E(t) … edges at time t
• Suppose that
  • N(t+1) = 2 * N(t)
• Q: what is your guess for
  • E(t+1) =? 2 * E(t)
Temporal Evolution of the Graphs

• N(t) … nodes at time t
• E(t) … edges at time t
• Suppose that
  \[ N(t+1) = 2 \times N(t) \]
• Q: what is your guess for
  \[ E(t+1) =? 2 \times E(t) \]
• A: over-doubled!
  But obeying the ``Densification Power Law”
Densification – Patent Citations

• Citations among patents granted

• @1999
  • 2.9 M nodes
  • 16.5 M edges

• Each year is a datapoint

\[ N(t) \quad E(t) \quad N(t) \]

\[ E(t) = 0.0002 \times 1.66 \quad R^2 = 0.99 \]
So many laws!

There will be more to come...

To date, there are **11 (or more) laws**

- RTG: A Recursive Realistic Graph Generator using Random Typing [Akoglu, Faloutsos]

What should you do?

- **Try as many distributions as possible** and see if your graph fits them.

- **If it doesn’t, find out the reasons.** Sometimes it’s due to errors/problems in the data; sometimes, it signifies some new patterns!
Polonium: Tera-Scale Graph Mining and Inference for Malware Detection [Chau, et al]