Graph Algorithms and Graph Databases

Dr. Daisy Zhe Wang
CISE Department
University of Florida
August 27th 2014
Google Knowledge Graph
-- Entities and Relationships
Graph Data!

Facebook Social Network graph
Graph Data! (cont.)

Citation Networks and Maps of Science, 2012
Web as a Graph

- Web as a directed graph:
  - Nodes: Webpages
  - Edges: Hyperlinks
The BowTie of the Web

- Strongly Connected Core
- IN
- OUT
- TENDRILS
- TUBES
-Disconnected components
LARGE GRAPH DATABASES AND QUERIES
Classes of Large Graphs

• Random graphs
  – Node degree is constrained
  – Less common

• Scale-free graphs
  – Distribution of node degree follows power law
  – Most large graphs are scale-free
  – Small world phenomena & hubs
  – Harder to partition
Classes of Large Graphs

Random Network

Scale-Free Network

Bell Curve Distribution of Node Linkages

Power Law Distribution of Node Linkages

Number of Nodes

Number of Links

Typical node

Number of Nodes

Number of Links

Number of Nodes (log scale)

Number of Links (log scale)
Organic Growth -> Scale Free

A SCALE-FREE NETWORK grows incrementally from two to 11 nodes in this example. When deciding where to establish a link, a new node (green) prefers to attach to an existing node (red) that already has many other connections. These two basic mechanisms—growth and preferential attachment—will eventually lead to the system's being dominated by hubs, nodes having an enormous number of links.
Let’s Go Hyper!

• Hyper-edge
  – A traditional edge is binary
  – A hyper edge relates \( n \) nodes
    • Child-of edge versus father, mother, child hyper-edge

• Hyper-node
  – A traditional node represents one entity
  – Hyper node represents a set of nodes
    • Person node versus family hyper-node
Social Networks

• Scale
  – LinkedIn
    • 70 million users
  – Facebook
    • 500 million users
    • 65 billion photos
• Queries
  – Alice’s friends
  – Photos with friends
• Rich graph
  – Types, attributes
Social Networks: Data Model

- **Node**
  - ID, type, attributes

- **Edge**
  - Connects two nodes
  - Direction, type, attributes
Managing Graph Data

• Here we focus on online access
  – Rather than offline access
    • Network analytics and graph mining

• Queries
  – Read

• Updates
  – Data update: change node payload
  – Structural update: modify nodes and edges
Example: Linked open Data (LoD)

• Semantic Web (Tim Berners-Lee)

• Scale
  – Hundreds of data sets
  – 30B+ tuples

• Queries
  – SPARQL

• Domains

http://www4.wiwiss.fu-berlin.de/lodcloud/state/
LoD Application Example

• Ozone level visualization

• 2 data sets
  – Clean air status [data.gov]
  – Castnet site information [epa.gov]

• 2 SPARQL queries
Web of Data: Data Model (1)

- Resource Description Framework (RDF)
- Uniform resource identifier (URI)
- **Triples!**
  
  \[
  1: \text{subject}, \quad 2: \text{predicate}, \quad 3: \text{object}
  \]

ex.: philippe, made, idmesh_paper: (URIs)

1: \url{http://data.semanticweb.org/person/philippe-cudre-mauroux}
2: \url{http://xmlns.com/foaf/0.1/made}
3: \url{http://data.semanticweb.org/conference/www/2009/paper/60}
Web of Data: Data Model (2)

• Naturally forms (distributed) **graphs**

• **Nodes**
  – URIs [subjects]
  – URIs / literals [objects]

• **Edges**
  – URIs [predicates]
  – Directed

![Diagram](image-url)
RDF Schemas (RDFS)

• Classes, inheritance
  – Class, Property, SubClass, SubProperty

• Constraints on structure (predicate)
  – Constraints on subjects (Domain)
  – Constraints on objects (Range)

• Collections
  – List, Bag
RDFS Example
RDF Storage (1)

• RDFa
  – Embedding RDF information (e.g., metadata, markup of attributes) in HTML pages
  – Supported by Google, Yahoo, etc
  – Example:

```html
<body>
  <div about="http://dbpedia.org/resource/Massachusetts">The Massachusetts governor is
    <span rel="db:Governor">
      <span about="http://dbpedia.org/resource/Deval_Patrick">Deval Patrick
        </span>,
    </span>
    the nickname is "<span property="db:Nickname">Bay State</span>",
    and the capital
    <span rel="db:Capital">
      <span about="http://dbpedia.org/resource/Boston">has the nickname "<span property="db:Nickname">Beantown</span>".
        </span>
    </span>
  </div>
</body>
```
RDF Storage (2)

• Various internal formats for DBMSs
  – Giant **triple table** (*triple stores*)
    • |subject|predicate|object|

  – **Property tables**
    • |subject|property1|property2|property3|...|

  – Sub-graphs
SPARQL (1/2)

• Declarative query language over RDF data
• SPJ combinations of *triple patterns*
  – E.g., “Retrieve all students who live in Seattle and take a graduate course”
  – Select ?s Where {
    ?s is_a Student
    ?s lives_in Seattle
    ?s takes ?c
    ?c is_a GraduateCourse }

23
SPARQL Query Execution

• Typically start from bound variables and performs **self-joins** on giant triple table

  – Select ?s Where {
    ?s is_a Student
    ?s lives_in Seattle
    ?s takes ?c
    ?c is_a GraduateCourse }

  – \( \pi_s \sigma_{p=“is\_a” \land o=“Student”} \)

  \( \bowtie \pi_s \sigma_{p=“lives\_in” \land o=“Seattle”} \)

  \( \bowtie \pi_s \left( \sigma_{p=“takes”} \bowtie_s \sigma_{p=“is\_a” \land o=“GraduateCourse”} \right) \)
SPARQL (2/2)

• Beyond conjunctions of triple patterns
  – Named graphs
  – Disjunctions
    • UNION
    • OPTIONAL (semi-structured data model)
  – Predicate filters
    • FILTER (?price < 30)
  – Duplicate handling (*bag semantics*)
    • DISTINCT, REDUCED
  – Wildcards
  – *Negation as failure*

WHERE { ?x foaf:givenName ?name .
  OPTIONAL { ?x dc:date ?date } .
  FILTER (!bound(?date)) }

25
Inference Queries

• Additional data can be inferred using various sets of logical rules
  PoliticianOf(Person,City) ⇒ LivesIn(Person,City)
  RetriedFrom(Prof,School) ⇒ WorkedFor(Prof,School)
• Specify which ones to use by entailment regimes [Glimm11]
  – RDF Schema has 14 entailment rules
    • E.g., (p,rdfs:domain,x) & (u, p, y)
      => (u,rdf:type,x)
Graph System Interfaces

• Relational: SQL
• Triple store: SPARQL
• Graph processing system/databases: API
Graph Database and Distributed Graph Processing Systems

- RDF-3X, JENA, RDF stores
- Oracle Triples Store and Semantic Web Technologies
- Neo4j – mostly widely used graph dbms
- Google Pregel
- Microsoft Trinity
- GraphLab, GraphX...
RDF-3X [Neumann08]

• Max Planck Institut für Informatik
  – Thomas Neumann & Gerhard Weikum
• Open-Source
• Triple-table storage
• Indexing: Clustered B+-trees on all six SPO permutations
• Compression: Leaf pages store deltas for each value
• Query Optimizations: order-preserving merge-joins, join orderings
GraphLab

- Open-source research project from CMU
- Support parallel graph operations with consistency guarantees via locking
- Do not support transactions and ad-hoc query processing (different from graph databases such as Neo4j)
Graph/Network datasets and Example Projects/Applications

- **Stanford Large Network Dataset Collection:** [http://snap.stanford.edu/data/](http://snap.stanford.edu/data/)
- **Benchmarking Graph Databases:** [http://istc-bigdata.org/index.php/benchmarking-graph-databases/](http://istc-bigdata.org/index.php/benchmarking-graph-databases/)
- **PageRank on Semantic Networks with Application to Word Sense Disambiguation:** [http://logic.cse.unt.edu/tarau/teaching/SoftEng/PapersToRead/PageRankWSD_coling04.pdf](http://logic.cse.unt.edu/tarau/teaching/SoftEng/PapersToRead/PageRankWSD_coling04.pdf)
[REVIEW] PAGE RANK: LINK ANALYSIS OVER LARGE GRAPHS

Adapted Slides from Jeff Ullman, Anand Rajaraman and Jure Leskovec from Stanford
How to organize the Web?

• First try: Human curated Web directories (e.g., Yahoo)

• Second try: Web Search
  – Information Retrieval using inverted index
  – Good for finding relevant docs in a small and trusted set (e.g., Newspaper articles, Patents)
  – But: Web is huge, full of untrusted documents, random things, web spam, etc.
    • E.g., Word Spam
2 Challenges of Web Search

1. Web contains many sources of information. Who to “trust”?
   - Observation: Trustworthy pages point to each other! (in&out-links)

2. What is the “best” answer to query “newspaper”?
   - No single right answer
   - Observation: Pages that actually know about newspapers might all be pointing to many newspapers (outlink)
   - Observation: A good newspaper is pointed to from many sources (inlink)
Solution: Ranking nodes on the Graph Based on Link Structures!

- All web pages are not equally “importance” can be captured by link structures
- There is large diversity in the web-graph node connectivity.
- Let’s rank the pages by the link structure!
  - Link Spam also possible but harder
Link Analysis

• Link Analysis algorithms: for computing importance of nodes in a graph
  – Page Rank
  – Topic-Specific (personalized) Page Rank
  – Mining for Communities
  – Web Spam Detection Algorithms
How to Rank Web Pages Based on Link Analysis

• Web pages are not equally “important”
  – www.joe-schmoe.com VS. www.ufl.edu
  – Page is more important if it has more inlinks

• Inlinks as votes
  – www.ufl.edu has 23,400 inlinks
  – www.joe-schmoe.com has 1 inlink

• Which inlinks are more important?
  – Recursive question!
  – Links from important pages count more
Example PageRank scores
Simple recursive formulation

• Each link’s vote is proportional to the importance of its source page
• If page $j$ with importance $r_j$ has $n$ outlinks, each link gets $r_j/n$ votes
• Page $j$’s own importance is the sum of the votes on its inlinks
  \[ r_j = ? = r_i/3 + r_k/4 \]
Simple “flow” model

A “vote” from an important page worth more
A page is important if it is pointed to by other important pages
Define a “rank” \( r_j \) for page \( j \)

\[
 r_j = \sum_{i \rightarrow j} \frac{r_i}{d_i}
\]

\( d_i \) is out-degree of node \( i \)

“flow” equations:
\[
 y = \frac{y}{2} + \frac{a}{2}
\]
\[
 a = \frac{y}{2} + m
\]
\[
 m = \frac{a}{2}
\]
Solving the flow equations

• 3 equations, 3 unknowns, no constants
  – No unique solution
  – All solutions equivalent modulo scale factor

• Additional constraint forces uniqueness
  – \( y + a + m = 1 \)
  – \( y = 2/5, a = 2/5, m = 1/5 \)

• Gaussian elimination method works for small examples, but we need a better method for large graphs

flow” equations:
\[
\begin{align*}
  y &= y/2 + a/2 \\
  a &= y/2 + m \\
  m &= a/2
\end{align*}
\]
Matrix formulation

- Stochastic adjacency matrix \( M \)
  - Matrix \( M \) has one row and one column for each web page
  - Let page \( i \) has \( d_i \) outlinks
  - If \( i \rightarrow j \), then \( M_{ij} = \frac{1}{d_i} \), Else \( M_{ij} = 0 \)
  - \( M \) is a column stochastic matrix, where columns sum to 1

- Rank vector \( r \): vector with one entry per web page
  - \( r_i \) is the importance score of page \( i \)
  - \( \sum_i r_i = 1 \)

- The flow equations can be written as
  \[
  \mathbf{r} = \mathbf{M}\mathbf{r}
  \]
Matrix Formulation Example

\[ \begin{align*}
  y &= y/2 + a/2 \\
  a &= y/2 + m \\
  m &= a/2
\end{align*} \]

\[
\begin{pmatrix}
y \\
a \\
m
\end{pmatrix}
= \frac{1}{2}
\begin{pmatrix}
y \\
a \\
m
\end{pmatrix}
+ \begin{pmatrix}
y \\
a \\
m
\end{pmatrix}
\]

\[
\begin{pmatrix}
y \\
a \\
m
\end{pmatrix}
= \begin{pmatrix}
1/2 & 1/2 & 0 \\
1/2 & 0 & 1 \\
0 & 1/2 & 0
\end{pmatrix}
\begin{pmatrix}
y \\
a \\
m
\end{pmatrix}
\]

\[
r = Mr
\]
A More General Example: Flow Equations = Matrix Formulation

• Remember the flow equation: \( r_j = \sum_{i \rightarrow j} r_i / d_i \)

• Flow equation in the matrix form: \( r = Mr \)

• Suppose page \( i \) links to 3 pages, including \( j \)
Rank Vector $r = \text{Eigenvector of } M$

• The flow equations can be written
  \[ r = Mr \]

• The rank vector $r$ is an eigenvector of the stochastic web matrix $M$
  – $x$ is an eigenvector with the corresponding eigenvalue $\lambda$
    if: $Ax = \lambda x$
  – In fact, its first or principal/dominant eigenvector, with corresponding eigenvalue $1$

• We can now efficiently solve for $r$!
  – The method is called Power iteration
Power Iteration method

- Given a web graph with $N$ nodes, where the nodes are pages and edges are hyperlinks
- Power iteration: a simple iterative scheme
  - Suppose there are $N$ web pages
  - Initialize: $r^0 = [1/N, \ldots, 1/N]^T$
  - Iterate: $r^{k+1} = Mr^k$
  - Stop when $|r^{k+1} - r^k|_1 < \varepsilon$
    - $|x|_1 = \sum_{1 \leq i \leq N} |x_i|$ is the $L_1$ norm
    - Can use any other vector norm e.g., Euclidean norm

\[ r^{(t+1)}_j = \sum_{i \to j} r^{(t)}_i / d_i \]

$d_i$ is out-degree of node $i$
Power Iteration Example

\[ y = \frac{y}{2} + \frac{a}{2} \]
\[ a = \frac{y}{2} + m \]
\[ m = \frac{a}{2} \]

\[
\begin{bmatrix}
y \\
a \\
m
\end{bmatrix}
= \begin{bmatrix}
\frac{1}{2} & \frac{1}{2} & 0 \\
\frac{1}{2} & 0 & 1 \\
0 & \frac{1}{2} & 0
\end{bmatrix}
\begin{bmatrix}
y \\
a \\
m
\end{bmatrix}
\]

\[ r = M_r \]
Power Iteration Example (cont.)

Power Iteration:
- Set $r_j = 1/N$
- 1: $r'_j = \sum_{i \rightarrow j} r_i / d_i$
  (i.e., $r' = Mr$)
- 2: $r = r'$
- Goto 1

\[
\begin{bmatrix}
y & a & m \\
y & 1/2 & 1/2 & 0 \\
a & 1/2 & 0 & 1 \\
m & 0 & 1/2 & 0 \\
\end{bmatrix}
\]

\[
\begin{align*}
y &= 1/3 & 1/3 & 5/12 & 3/8 & 2/5 \\
a &= 1/3 & 1/2 & 1/3 & 11/24 & \ldots & 2/5 \\
m &= 1/3 & 1/6 & 1/4 & 1/6 & 1/5 \\
\end{align*}
\]
Why Power Iteration works?

- **Power iteration**: A method for finding dominant eigenvector (the vector corresponding to the largest eigenvalue)
  - $r(1) = M \cdot r(0)$
  - $r(2) = M \cdot r1 = M^2 \cdot r0$
  - $r(3) = M \cdot r2 = M^3 \cdot r0$

- **Claim**: Sequence $M \cdot r0, M^2 \cdot r0, \ldots M^k \cdot r0, \ldots$ approaches the dominant eigenvector of $M$
Random Walk Interpretation

• Imagine a random web surfer
  – At any time t, surfer is on some page P
  – At time t+1, the surfer follows an outlink from P uniformly at random
  – Ends up on some page Q linked from P
  – Process repeats indefinitely

• Let $p(t)$ be a vector whose $i^{th}$ component is the probability that the surfer is at page $i$ at time $t$
  – $p(t)$ is a probability distribution on pages
  – $p(t)$ is a link-based importance rank $t \rightarrow \infty$
The stationary distribution

• Where is the surfer at time $t+1$?
  – Follows a link uniformly at random
  – $p(t+1) = Mp(t)$

• Suppose the random walk reaches a state such that $p(t+1) = Mp(t) = p(t)$
  – Then $p(t)$ is called a stationary distribution for the random walk

• Our rank vector $r$ satisfies $r = Mr$
  – So $r$ is a stationary distribution $p(t)$ for the random surfer
Existence and Uniqueness

A central result from the theory of random walks (aka Markov processes):

For graphs that satisfy certain conditions (i.e., strongly connected, and no dead ends), the stationary distribution is unique and eventually will be reached no matter what the initial probability distribution at time $t = 0$.

Is the Web a strongly connected graph?
PageRank: Problems

2 problems:

(1) Spider traps (all out-links are within the group)
   - Eventually spider traps absorb all importance

(2) Some pages are dead ends (have no out-links)
   - Such pages cause importance to “leak out”
Spider traps

• A group of pages is a spider trap if there are no links from within the group to outside the group
  – Random surfer gets trapped

• Spider traps violate the conditions needed for the random walk theorem
Random teleports

• The Google solution for spider traps
• At each time step, the random surfer has two options:
  – With probability $\beta$, follow a link at random
  – With probability $1-\beta$, jump to some page uniformly at random
  – Common values for $\beta$ are in the range 0.8 to 0.9
• Results: Surfer will teleport out of spider trap within a few time steps
Dead ends

- Pages with no outlinks are “dead ends” for the random surfer
  - Nowhere to go on next step
  - All importance becomes zero
Dealing with dead-ends

• Teleport
  – Follow random teleport links with probability 1.0 from dead-ends
  – Adjust matrix accordingly

• Prune and propagate
  – Preprocess the graph to eliminate dead-ends
  – Might require multiple passes
  – Compute page rank on reduced graph
  – Approximate values for deadends by propagating values from reduced graph
More Graph Analytics and Mining Algorithms

• Wiki Book on Graph Algorithms: http://en.wikipedia.org/wiki/Book:Graph_Algorithms
  – Shortest path finding
  – Topological sorting
  – Minimum spanning tree
  – Strongly connected components
  – Cliques ...
Summary

• Graph Data and Graph Algorithms
  – Page Rank Algorithm
  – Shortest path, Strongly connected components, etc.

• Graph Database and Queries (e.g., SPARQL)

• Project Ideas – find relevant dataset and papers → problem statement and ideas for techniques