Graph Computing

Guest Lecture #1: CSC526

Rangan Sukumar (Email: sukumarsr@ornl.gov)
At the end of today’s class

• Email sukumarsr@ornl.gov the following (2 sentences maximum)
  – What did you learn from the lecture?
  – What new “thing” was interesting?
  – What would you like to see “more” next class?
A quick survey...

Data Structures (Order Complexity) ?

Graph Data Structure ?

Structured Query Language- SQL ?

Python ?

Eigen Decomposition ?

SPARQL ?

W3C?
Today’s Class

• Why Graph Analytics?
  – Why Graphs?
  – Real-world Graphs
  – Graphs @ ORNL

• Graphs
  – Fundamentals /Definitions
    • Representation, Processing
  – “Big Data” Graphs
    • Data Science of Graphs
    • Science of Graph-Data
  – Graph Algorithms

Thursday’s Class

• Graph Algorithms
  – Recap
  – Graph Pattern Search and Mining using SPARQL
  – Random walk algorithm on Semantic Graphs
  – (?)

Assignment #1 on Thursday

• Best work will get access to Cray’s Urika @ ORNL
Philosophical ‘Big Data’ takeaway from today’s lecture...
Warm-up Exercise #0:

What is common to (airplane, rainbow, star, thunder)?
What did you all just do?

What is common to (airplane, rainbow, star, thunder)?

Discovery by “interrogation”
A storage and retrieval problem
Example: Memorizing a vocabulary

Discovery by “modeling/simulation”
A pattern-discovery/modeling problem
Example: Creating models for recognition

Discovery by “association”
A disparate data fusion problem
Example: Mapping visual and semantic similarity
Data-driven Discovery: Challenges at Scale

The Lifecycle of Data-Driven Discovery

What did we learn from “Big Data”?

- Knowledge discovery is a “greedy” and "never ending" thirst.
  - Big Data produces “Bigger Data”
  - “Lifecycle” management vs. “Project” management

- Big Data comes with Big Expectations
  - Data sets are expected to answer more than one question.
  - “We have lots of data – we do not know what questions to ask?”

- If Big Data => Smarter decisions, we need “Smarter Methods”
  - Discover “newer” insights in context with evolving new knowledge
  - Methods that can work well when there is more noise than signal.

Why Graph Computing?

The Lifecycle of Data-Driven Discovery

Querying and Retrieval
e.g. Google, Databases

Interrogation

Association
e.g. Mashups

Modeling, Simulation, & Validation
e.g. Climate Change Prediction

Better Data Collection

Graph Computing…

- Supports discovery by interrogation, association and predictive modeling from structured and unstructured data
- Supports discovery with evolving knowledge and incremental domain hints
- Supports exploratory and confirmatory analysis
  - Data and meta-data integrated analytics
  - Flexible data structure seamless to growth while avoiding analytical artifacts
Data-driven Discovery in the Big Data Era

The Lifecycle of Data-Driven Discovery

Querying and Retrieval
e.g. Google, Databases

Interrogation

Association
e.g. Mashups

Modeling, Simulation, & Validation
e.g. Climate Change Prediction

Better Data Collection

Predictive Modeling

A Domain Scientist's View

The Process of Data-Driven Discovery

Science of scalable predictive functions

Pattern Discovery

e.g. Hypothesis generation

Pattern Recognition

e.g. Classification, Clustering

Science of data (Data-aware)

e.g. Deep learning, Feature extraction, Meta-tagging

Data science (Infrastructure-aware)

Shared-storage, shared-memory, shared-nothing

A Data Scientist's View

Takeaway from my lectures...

• The Data Science for Graphs
• The Science of Graph-Data
• Pattern Retrieval vs. Pattern Mining

Graph Computing at Scale
Why Graphs?

Graphs are ubiquitous

Social networks
Source: LinkedIn

Internet of Things
Source: Cisco

Protein networks
Source: Cisco
Why Graphs?

Graphs are ubiquitous

Electric Grid

Transportation networks

Knowledge networks

Source: Visual complexity
Motivation: Relationship between patterns of migration and cancer
Graph Computing Projects @ ORNL

Motivation: Automatically tag and triage images for law enforcement

WordNet
(30K adjectives, 146 K nouns, ~25 K verbs) in ~117K synsets

ImageNet
~14M images, ~22k synsets

MIRFLICKR
~25K – 1 M images, ~8 tags per image

ConceptNet
~3.4 M words, ~57 K types of relationships, ~10 million assertions
Given a few examples of fraud (important activity), can we
(i) Automatically discover patterns typically associated with suspicious activity?
(ii) Extrapolate such high-risk patterns for investigation and fraud prevention?
Given a knowledgebase and new clinical data/experiments, can we
(i) Find “novel” patterns of interest?
(ii) Rank and evaluate the patterns for significance?

Graphs @ ORNL: Let’s do a 2-minute exercise

Physical contacts in an inpatient hospital facility

Who are the Patients ? Doctors ? Nurses ? Admins ?
Motivation: Based on the understanding of the contact patterns, build a quarantine strategy

Graphs @ ORNL: Imagine...

Physical contacts in an inpatient hospital facility

Ebola !!
Graphs Definitions

- **Homogenous**
- **Directed**
- **Undirected**
- **Heterogeneous**
- **Multigraph**
- **Hypergraph**

- **Bi-partite**
Graph Representation

### Adjacency list

<table>
<thead>
<tr>
<th></th>
<th>Adjacent to</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b, c</td>
</tr>
<tr>
<td>b</td>
<td>a, c</td>
</tr>
<tr>
<td>c</td>
<td>a, b</td>
</tr>
</tbody>
</table>

### Property graphs

- **Nodes**: Person, Movie
- **Edges**: Knows, Watches

### Triples

- `<subject, predicate, object>`

### Adjacency matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</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>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Tensors

**Data Format**

- `e1 e2 e3 e4`

**Matrix Representation**

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### Edge list

- **Edges**: a-b, b-c, c-a

### Incidence matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Graph Representation

• When should you use what?
  – Is data homogenous? Is data dense or sparse? Is data going to grow?

<table>
<thead>
<tr>
<th></th>
<th>Adjacency list</th>
<th>Adjacency matrix</th>
<th>Incidence matrix</th>
</tr>
</thead>
<tbody>
<tr>
<td>Storage</td>
<td>$O(</td>
<td>V</td>
<td>+</td>
</tr>
<tr>
<td>Add vertex</td>
<td>$O(1)$</td>
<td>$O(</td>
<td>V</td>
</tr>
<tr>
<td>Add edge</td>
<td>$O(1)$</td>
<td>$O(1)$</td>
<td>$O(</td>
</tr>
<tr>
<td>Remove vertex</td>
<td>$O(</td>
<td>E</td>
<td>)$</td>
</tr>
<tr>
<td>Remove edge</td>
<td>$O(</td>
<td>E</td>
<td>)$</td>
</tr>
<tr>
<td>Query: are vertices $u$, $v$ adjacent? (Assuming that the storage positions for $u$, $v$ are known)</td>
<td>$O(</td>
<td>V</td>
<td>)$</td>
</tr>
<tr>
<td>Remarks</td>
<td>When removing edges or vertices, need to find all vertices or edges</td>
<td>Slow for add/remove vertices, because matrix must be resized/copied</td>
<td>Slow to add or remove vertices and edges, because matrix must be resized/copied</td>
</tr>
</tbody>
</table>
Choice of representation relies on what we want to do...

Two kinds of graph problems

1. Graph Pattern Search
   Subject matter expert knows what to look for and has a hunch for patterns in the graph

2. Graph Mining
   Need to understand generating model behind data or
   Don’t know anything about the data.
Two kinds of graph computing problems...

Graph Pattern Search

Graph Mining

Same data but different kind of insight
Choice of representation relies on what we want to do...

Two kinds of graph computing problems

1. Graph Pattern Search

2. Graph Mining

Containment Pattern: Retrieve all graphs from a graph database, such that they contain a given query graph (exact and approximate).

Source: Khan et al., ICDE 2012
Choice of representation relies on what we want to do...

Two kinds of graph computing problems

1. Graph Pattern Search

2. Graph Mining

Similarity Pattern: Retrieves all graphs from a graph database, that are similar to the query graph (exact and approximate).

Source: Khan et al., ICDE 2012
Choice of representation relies on what we want to do...

Two kinds of graph computing problems

1. Graph Pattern Search
2. Graph Mining

Matching Pattern: Find all occurrences of a query graph in a large target network (exact and approximate).

Source: Khan et al., ICDE 2012
Exercise #2:

\[
\begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
= ?
\]
Exercise #2:

\[
\begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 \\
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
\]

\[=\]

\[
\begin{pmatrix}
0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
0 & 0 & 0 \\
0 & 0 & 0 \\
1 & 0 & 0 \\
1 & 0 & 0 \\
0 & 0 & 0 \\
0 & 0 & 1 \\
\end{pmatrix}
\begin{pmatrix}
0 & 1 & 0 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
\end{pmatrix}
\]

\[=\]
Exercise #2: What you just did was pattern search!

Is there a pattern involving 3, 4 and 6?

(adjacency matrix of pattern)

Adjacency matrix of pattern:

Yes!

Source: John Gilbert, UCSB
Exercise #2: Really !!

Are 1 and 4 connected?

Query Incidence Matrix $(Q)$

$$
\begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}
$$

Adjacency matrix $(G_1)$

$$
\begin{pmatrix}
0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & 1 \\
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0
\end{pmatrix}
$$

Query Incidence Matrix $(Q^T)$

$$
\begin{pmatrix}
1 & 0 \\
0 & 0 \\
0 & 0 \\
0 & 1 \\
0 & 0 \\
0 & 0
\end{pmatrix}
$$

$$
\begin{pmatrix}
0 & 0 \\
0 & 0
\end{pmatrix}
$$

No!
But....

A simple subgraph retrieval is $O(N^3)$ matrix multiplication !!

What if we can not fit all of the data into memory ?

What if we do not have enough compute to do a $O(N^3)$ multiplication ?
What to do when data sizes are so big they don’t fit into memory?

- **Solution: Databases**
  - Block-Matrix Files – Idea: Split the adjacency matrix into smaller files
    - Linear Algebra
  - SQL- Idea: Create a table of nodes, edges and weights
    - Columnar processing massively parallel processing databases
  - No-SQL (Key-Value Stores) - Idea: Save as adjacency lists
    - MapReduce processing
  - Graph Databases - Idea: Property Graphs, Triple stores etc.
    - Basic Graph Pattern search using ARQ algebra
There are other ways to do it…

- **Filtering Phase:** Feature-based index is used to filter out the negative results and generate a candidate sets.

- **Verification Phase:** Precise Subgraph Isomorphism Testing to generate final results from the candidate set.

Source: Khan et al., ICDE 2012
Pattern search using query languages: SQL and SPARQL

# List of all nodes
Table: [nodes]
id: varchar(100)
attrib1: varchar(100)

# List of all edges
Table: [edges]
From_node : varchar(100)
To_node : varchar(100)

# Index to speed up search
CREATE INDEX a_idx ON edges (From_node);
CREATE INDEX b_idx ON edges (To_node);

# Retrieving all neighbors of Node 1
SELECT *
FROM [nodes] as n
LEFT JOIN [edges] as e ON n.id = e.To_node
WHERE e.From_node = 1'

# Input graph as triples
<1> < connected> <2>
<1> < connected> <5>
<2> < connected> <3>
<5> < connected> <3>
<3> < connected> <4>
<4> < connected> <6>

Source: John Gilbert, UCSB

# SPARQL query to extract 1-hop neighbors
SELECT ?n2
WHERE { <1> <connected> ?n2}

# SPARQL query to extract 2-hop neighbors
SELECT ?n3
WHERE { <1> <connected> ?n2
        ?n2 <connected> ?n3}
Graph Mining....

Two kinds of graph problems

1. Graph Pattern Search

Subject matter expert knows what to look for and has a hunch for patterns in the graph

2. Graph Mining

Need to understand generating model behind data or

Don’t know anything about the data.
Graph-theoretic Algorithms (Others)

1. Computationally feasible
   - Degree distribution
   - PageRank
   - Connected components
   - Diameter, Eccentricity, Radius
   - Triangles
   - Belief Propagation
   - Random Walks
   - Community Detection
   - Recommender Systems
   - Breath/Depth-first Search
   - Minimum Spanning Tree

2. Computationally HARD
   - Graph Isomorphism
   - Vertex Coloring
   - Maximum Clique Size
   - Flow Problems
   - Motif-Matching
Graph-theoretic Algorithms

What are popular graph mining algorithms tell you about your data?

Degree Distribution: Helps understand structure

Connected Component: Helps understand communities

PageRank: Ranks nodes by importance

Eccentricity (Radius, Diameter, etc.): Find central nodes

Image courtesy: Boston University
Graph Mining Algorithms: Science of Graphs

• Given real-word graphs, how to understand them using graph mining techniques?

“In a social network of 1 million people, each node has an average of 50 friends.”

Q1: If you pick someone at random, what is the best guess for how many friends that person will have?

Q2: Can someone have a 1000 friends?
Graph Mining Algorithms: Science of Graphs

• Given real-word graphs, how to understand them using graph mining techniques?

“In a social network of 1 million people, each node has an average of 50 friends.”

Q1: If you pick someone at random, what is the best guess for how many friends that person will have?

Q2: Can someone have a 10000 friends?

Source: C. Faloutsos, CMU
What does degree-distribution tell you?

Degree Distribution: Helps understand structure

Image courtesy: Boston University
What do Eigen-values tell you?

- What do Eigen-values of the adjacency matrix tell you?

Eigen-values follow a power-law as well
Other Graph Laws...

• Triangles

Source: C. Faloutsos, CMU

X-axis: degree
Y-axis: mean # triangles
$n$ friends $\rightarrow \sim n^{1.6}$ triangles

Further reading: Morgan and Claypool lectures: Patterns in Static Graphs by D. Chakrabarti and C. Faloutsos.

• Effective Diameter

KDD’13 BIGMINE
(c) 2013, C. Faloutsos
**Exercise #3:**

\[ A = \begin{pmatrix}
0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 \\
1 & 0 & 0 & 1 \\
1 & 1 & 1 & 0
\end{pmatrix} \]

\[ A^2 = \begin{pmatrix}
3 & 1 & 1 & 1 \\
1 & 2 & 2 & 1 \\
1 & 2 & 2 & 1 \\
2 & 1 & 1 & 3
\end{pmatrix} \]

\[ A^3 = \begin{pmatrix}
4 & 5 & 5 & 5 \\
5 & 2 & 2 & 5 \\
5 & 2 & 2 & 5 \\
5 & 5 & 5 & 4
\end{pmatrix} \]

Compute \((\text{trace}(A^3)/6)\) ?

Any guesses what you just did?
Exercise #3: Hint!

\[ A = \begin{pmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{pmatrix} \quad A^2 = \begin{pmatrix} 3 & 1 & 1 & 1 \\ 1 & 2 & 2 & 1 \\ 1 & 2 & 2 & 1 \\ 2 & 1 & 1 & 3 \end{pmatrix} \quad A^3 = \begin{pmatrix} 4 & 5 & 5 & 5 \\ 5 & 2 & 2 & 5 \\ 5 & 2 & 2 & 5 \\ 5 & 5 & 5 & 4 \end{pmatrix} \]

Compute \((\text{trace}(A^3)/6)\) ?

Any guesses what you just did?
Computing triangles using SQL and SPARQL

SELECT count(*)
from edges e1
join edges e2 on e1.to_node = e2.from_node and e1.from_node < e2.from_node
join edges e3 on e2.to_node = e3.from_node and e3.to_node = e1.from_node and e2. from_node < e3.from_node

SELECT (COUNT(DISTINCT *) AS ?count)
WHERE
{
  FILTER(STR(?x) < STR(?y))
  FILTER(STR(?y) < STR(?z))
}

Which approach would take the longest time and why?
Recap

- The Data Science for Graphs
- The Science of Graph-Data
- Pattern Retrieval vs. Pattern Mining

Graph Computing at Scale
Graph Computing at Scale

Scaling for Volume

Accommodating Heterogeneity

Open Challenges:

(i) Processing massive non-partitionable graphs in memory
(ii) Complexity of graph-pattern search (NP-hardness)
(ii) Generic features for pattern discovery and predictive modeling
## What are we doing at ORNL?

### Unique Opportunity

<table>
<thead>
<tr>
<th></th>
<th>Titan</th>
<th>Apollo</th>
<th>CADES (Cloud)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Discovery Approach</strong></td>
<td>Modeling and Simulation</td>
<td>Association</td>
<td>Querying, Prediction</td>
</tr>
<tr>
<td><strong>Architecture</strong></td>
<td>Shared-compute</td>
<td>Shared-memory</td>
<td>Shared-storage</td>
</tr>
<tr>
<td><strong>Scalability</strong></td>
<td>Compute (# of cores)</td>
<td>Horizontal (# of datasets)</td>
<td>Vertical (# of rows)</td>
</tr>
<tr>
<td><strong>Algebra</strong></td>
<td>Linear</td>
<td>Relationship</td>
<td>Set-theoretic</td>
</tr>
<tr>
<td><strong>Challenge (Pros)</strong></td>
<td>Resolution</td>
<td>Heterogeneity</td>
<td>Cost</td>
</tr>
<tr>
<td><strong>Challenge (Cons)</strong></td>
<td>Dimensionality</td>
<td>Custom Solution</td>
<td>Flexibility</td>
</tr>
<tr>
<td><strong>Leadership</strong></td>
<td>#2 in the world (2013)</td>
<td>1 of 15 installs (2013)</td>
<td>--</td>
</tr>
<tr>
<td><strong>User-interface</strong></td>
<td>OpenMP, MPI, CUDA</td>
<td>SPARQL</td>
<td>SQL</td>
</tr>
</tbody>
</table>
At the end of today’s class

• Email sukumarsr@ornl.gov the following (2 sentences maximum)
  – What did you learn from the lecture?
  – What new “thing” was interesting?
  – What would you like to see “more” next class?
Next Class

• Graph Computing at Scale
• The Random Walker Algorithm
• Reasoning with Semantic Graphs
  – https://github.com/ssrangan/gm-sparql
• Assignment #1
  – Using Random walks to explore knowledge graphs
  – Parallelize algorithm : Best implementation gets to try it out on Urika