Research Directions for Big Data Graph Analytics

John A. Miller, Lakshmish Ramaswamy, Krys J. Kochut and Arash Fard

Department of Computer Science
University of Georgia
Athens, GA, USA
Outline

- Introduction
- Graph Analytics
- Problems in Graph Analytics
- Path Problems
- Pattern Problems
  - Graph Simulations
  - Graph Morphisms
- Comparison of Pattern Matching Models
- Applications
- Computational Models and Frameworks
- Conclusions and Future Work
Introduction

• Big Data Analytics

• Find patterns or relationships in large volumes of data

\[ y_1, y_2, \ldots, y_k = f(x_1, x_2, \ldots, x_k) \]

• Input \( x \) to \( f \) can be stored in an \textit{m-by-n matrix} and output \( y \) can be stored in an \textit{m-by-k matrix}.

• Numerous techniques from Regression to Neural Networks may be applied to analyze the data, e.g., find correlations, make predictions.
Graph Analytics

• Another way to look at data focuses on how \( n \) data items are connected

\[ f(u, v) \] – where \( f \) is a function \( u \) and \( v \) are data items

• Indicating for example relatedness, similarity, distance, etc.

• Again an \( n \text{-by-} n \) matrix can be used to tabulate the function \( f \)

• In large data sets, many of the data items will not be directly connected, so it is more efficient to use some form of graph
Graph Analytics

• Vertex-labeled Digraph
  – directed graph with vertex labels

• Fully-labeled Multi-Digraph
  – adds edge labels
  – allows multiple edges between 2 vertices, if labels are distinct
  – $G(V, E, L, l)$
  – vertices, labeled edges, label set, vertex labeling
There are many problems in graph analytics. Several involve the three “P”s:

- **Path** Problems
  - Reachability
  - Shortest Path

- **Pattern** Problems
  - Graph Simulation
  - Graph Morphisms

- **Partition** Problems
Path Problems

- **Reachability**
  - Given a graph $G$ and two vertices, $u, w \in G.V$,
  - find a path (sequence of edges) connecting them
    
    $$\text{path}(u, w) = u v_1 l_1 , v_1 v_2 l_2 , \ldots , v_n w l_{n+1} \in G.E$$
  - Reachability is simply
    
    $$\text{reach}(u, w) = \exists \text{path}(u, w)$$

- **Shortest Path**
  - Given $k$ vertices, find a minimum distance path that
    - includes all $k$ vertices.
  - For $k = 2$, the two vertices will be endpoints
  - Algorithms: Dijkstra, Bellman-Ford
Pattern Problems

- **Pattern Matching Model**
  - Given a query graph $Q$, match its labeled vertices to corresponding labeled vertices in a data graph $G$
    $$\Phi : Q.V \rightarrow 2^G.V$$
  - All the pattern matching models start with matching vertex labels
  - For each vertex $u$ in $Q.V$, require
    $$\forall \ u' \in \Phi(u), \ l(u') = l(u)$$

- **Graph Simulation**
  - For each label matching pair $(u, u')$ in $Q.V \times G.V$,
  - require matching children
    $$\forall \ v \in child_Q(u), \ \exists \ v' \in \Phi(v) \text{ such that } u'v' \in G.E$$
Example Query (reduced G)

a) Q: Pattern

b) G: Data Graph
Pattern Problems

• Dual Simulation
  – For each vertex pair \((u, u')\) in \(Q.V \times G.V\), also require matching parents
    \[
    \forall w \in parent_Q(u), \exists w' \in \Phi(w) \text{ such that } w'u' \in G.E
    \]

• Strong Simulation
  – Matching patterns in \(G\) must be contained in some ball \(B\)
    \[
    \text{radius}(B) = \text{diameter}(Q)
    \]
  – Locality makes the pattern in \(G\) look more like query \(Q\)
Pattern Problems

- **Strict Simulation**
  - only balls containing vertices in an *initial dual match*

- **Tight Simulation**
  - only balls with centers corresponding to a *central vertex* in Q
    \[ \text{radius}(B) = \text{radius}(Q) \]

- **CAR-tight Simulation**
  - child and parent match must satisfy
    - Type – e.g., child labels A and B
    - **Cardinality** – e.g., 2 A's and 3 B's *(new restriction)*
Pattern Problems

- **Graph Homomorphism**
  - Mapping function $f: Q.V \rightarrow G.V$ such that
    \[
    \forall u \in Q, \; u' = f(u), \; l(u') = l(u)
    \]
    \[
    uv \in Q.E \text{ implies } u'v' \in G.E
    \]

- **Subgraph Isomorphism**
  - Bijective function $f: Q.V \rightarrow G'.V$ (G' subgraph of G) such that
    \[
    \forall u \in Q, \; u' = f(u), \; l(u') = l(u)
    \]
    \[
    uv \in Q.E \text{ iff } u'v' = f(u)f(v) \in G'.E
    \]
## Comparison of Matching Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Subgraph Results</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph Sim</strong></td>
<td>(\Phi(1, 2, 3, 4) \to {(1, 6, 8, 12, 16, 19, 20, 24, 27, 30), {2, 7, 13, 15, 17, 21, 23, 26, 29},){3, 4, 5, 9, 11, 14, 18, 22, 25, 28}, {3, 4, 5, 9, 11, 14, 18, 22, 25, 28}}</td>
<td>29</td>
</tr>
<tr>
<td><strong>Dual Sim</strong></td>
<td>(\Phi(1, 2, 3, 4) \to {(1, 6, 8, 12, 16, 19, 20, 24, 27, 30), {2, 7, 13, 15, 17, 21, 23, 26, 29},){3, 4, 5, 9, 14, 18, 22, 25, 28}, {3, 4, 5, 9, 14, 18, 22, 25, 28}}</td>
<td>28</td>
</tr>
<tr>
<td><strong>Strong Sim</strong></td>
<td>(\Phi(1, 2, 3, 4) \to {(1, 6, 8), {2, 7}, {3, 4, 5, 9}, {3, 4, 5, 9}}, (12, 13, 14, 14), ({16, 19, 20}, {15, 17, 21}, {14, 18, 22}, {14, 18, 22})}</td>
<td>20</td>
</tr>
<tr>
<td><strong>Strict Sim</strong></td>
<td>(\Phi(1, 2, 3, 4) \to {(1, 6, 8), {2, 7}, {3, 4, 5, 9}, {3, 4, 5, 9}}, (12, 13, 14, 14)}</td>
<td>12</td>
</tr>
<tr>
<td><strong>Tight Sim</strong></td>
<td>(\Phi(1, 2, 3, 4) \to {(1, 2, {3, 4, 5}, {3, 4, 5}}, (12, 13, 14, 14)}</td>
<td>8</td>
</tr>
<tr>
<td><strong>car-Tight Sim</strong></td>
<td>(\Phi(1, 2, 3, 4) \to {(1, 2, {3, 4, 5}, {3, 4, 5}}}</td>
<td>5</td>
</tr>
<tr>
<td><strong>Graph Homomorph</strong></td>
<td>(f(1, 2, 3, 4) \to {(1, 2, 3, 4), (1, 2, 3, 5), (1, 2, 4, 5), (1, 2, 3, 3), (1, 2, 4, 4), (1, 2, 5, 5), (12, 13, 14, 14)}}</td>
<td>8</td>
</tr>
<tr>
<td><strong>Subgraph Isomorph</strong></td>
<td>(f(1, 2, 3, 4) \to {(1, 2, 3, 4), (1, 2, 3, 5), (1, 2, 4, 5)}}</td>
<td>5</td>
</tr>
</tbody>
</table>
## Comparison of Matching Models

<table>
<thead>
<tr>
<th>Model</th>
<th>Complexity Class</th>
<th>Source</th>
<th>Results Contained In</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph Sim</td>
<td>Quadratic</td>
<td>Henzinger et al. 1995</td>
<td>-</td>
</tr>
<tr>
<td>Dual Sim</td>
<td>Cubic</td>
<td>Ma et al. 2011</td>
<td>Graph Sim</td>
</tr>
<tr>
<td>Strong Sim</td>
<td>Cubic</td>
<td>Ma et al. 2011</td>
<td>Dual Sim</td>
</tr>
<tr>
<td>Strict Sim</td>
<td>Cubic</td>
<td>Fard et al. 2013</td>
<td>Strong Sim</td>
</tr>
<tr>
<td>Tight Sim</td>
<td>Cubic</td>
<td>Fard et al. 2014a</td>
<td>Strict Sim</td>
</tr>
<tr>
<td>Car-Tight Sim</td>
<td>Cubic</td>
<td>Fard et al. 2014b</td>
<td>Tight Sim</td>
</tr>
<tr>
<td>Graph Homeomorph</td>
<td>NP-hard</td>
<td>Fortune et al. 1980</td>
<td>-</td>
</tr>
<tr>
<td>Graph Homomorph</td>
<td>NP-hard</td>
<td>Hell and Nesetril, 1990</td>
<td>Graph Homeomorph and Tight Sim</td>
</tr>
<tr>
<td>Subgraph Isomorph</td>
<td>NP-hard</td>
<td>Garey and Johnson, 1979</td>
<td>Graph Homomorph and Car-Tight Sim</td>
</tr>
</tbody>
</table>
Applications

- **Graph Databases**
  - Neo4j, OrientDB and Titan

- **Semantic Web**
  - SPARQL Queries
  - (subject, predicate, object) subject -edge-> object

- **Social Media**
  - Facebook, Twitter, LinkedIn, Amazon
  - Find patterns in graphs
Computational Models and Frameworks

- MapReduce/Hadoop
  - May not be ideal for highly iterative algorithms
- Apache Storm
  - More efficient stream processing
- Apache Spark
  - Can maintain intermediate results in main memory
- Bulk Synchronous Parallel (BSP)
  - Vertex-Centric Model
  - Testing of Graph through Tight Simulation showed good scalability
Conclusions and Future Work

- Research in Graph Pattern Matching
  - Improved Sequential Algorithms
    - Recent ones for Subgraph Isomorphism
      - DualIso [Saltz et al. 2014] and Turbolso [Han et al. 2013]
      - much faster than prior generation Ullmann and VF2
  - Parallel and Distributed Algorithms/Implementations
    - More Vertex-Centric Programming
    - Asynchronous Programming – harder, faster
  - Extensions
    - Adding Edge Labels
    - Practical Solutions for Graph Database and SPARQL Query Engines
Full Example

A: Arts Book
B: Biography Book
C: Children’s Book
M: Music CD

a) Q: Pattern

b) G: Data Graph