Techniques for Graph Analytics on Big Data

Usman Nisar, Arash Fard, John A. Miller

Computer Science Department
University of Georgia, Athens, GA
Outline

- Introduction
- Subgraph Pattern Matching
  - Types of Subgraph Pattern Matching
- Models of Computation
- Distributed Algorithms
  - Performance Evaluation
- Graph Partitioning
  - Results Analysis
- Conclusion
Introduction

- Directed graph $G(V, E, l)$
  - $V$ is a set of vertices
  - $E \subseteq V \times V$ is a set of directed edges
  - $l: V \rightarrow \mathcal{N}$ is a function mapping vertices to labels

- Versatility and expressivity
  - Social networks, web search engines, genome sequencing, etc.

- Data sizes are growing rapidly
  - Facebook: 1 billion + users, average degree of 140
  - Twitter: 200 million + users, 400 million tweets each day

- Bigger datasets mean bigger graphs
Subgraph Pattern Matching

A graph $G'(V', E', l')$ is a subgraph of $G(V, E, l)$ if:
1. $V' \subseteq V$
2. $E' \subseteq V' \times V' \subseteq E$
3. $\forall v \in V', l'(v) = l(v)$

A pattern (or query) is a graph that we want to find in a bigger graph

Subgraph pattern matching: Given a query graph, find all subgraphs of another graph (the data graph) that are similar to the query based on certain criteria
Types of Subgraph Pattern Matching

- Exact
  - Subgraph Isomorphism

- Heuristic
  - Graph Simulation
  - Dual Simulation
  - Strong Simulation
Subgraph Isomorphism

- Exact matching from the pattern to data graph
- Labels must be the same
- Ullmann’s algorithm, VF2
- NP-hard problem in general case
- Current solutions not practical for large graphs
Example: Subgraph Isomorphism

PM: Product Manager
SD: Software Developer
SA: System Analyst
DB: Database Designer
AI: AI Specialist

Matches: \{4, 8, 6, 7\}, \{5, 8, 6, 7\}
Graph Simulation

A vertex in a data graph $G$ matches a vertex in query graph $Q$ via graph simulation iff

1. both have the same label
2. a subset of its children match all the children of its corresponding vertex in the query graph

HHK algorithm[1]

Graph Simulation

PM: Product Manager
SD: Software Developer
SA: System Analyst
DB: Database Designer
AI: AI Specialist

Data Graph
Dual Simulation

- Adds a duality condition to graph simulation
- A vertex in a data graph becomes a match with a vertex in the query graph iff
  1. Both have the same label
  2. A subset of its children matches all the children of its corresponding vertex in the data graph
  3. A subset of its parents matches all the parents of its corresponding vertex in the data graph
- $O((|V_q| + |E_q|) (|V| + |E|))$
- More restrictive than graph simulation
Example: Graph → Dual Simulation

Query

PM: Product Manager
SD: Software Developer
SA: System Analyst
DB: Database Designer
AI: AI Specialist

Data Graph
Example: Graph $\rightarrow$ Dual Simulation

PM: Product Manager
SD: Software Developer
SA: System Analyst
DB: Database Designer
AI: AI Specialist

Query

Data Graph
Strong Simulation

- Extends dual simulation with a locality condition

- A **ball** $G[v, r]$ is a subset of a graph $G$ containing:
  - all vertices $V_B$ within an undirected distance $r$ of the vertex $v$
  - all the edges between the vertices in $V_B$

```
  1
  /|
 / |\
2 3 4
  |
 5 6
  |
 7
```
Strong Simulation

- Extends dual simulation with a locality condition
- A **ball** $G[v, r]$ is a subset of a graph $G$ containing:
  - all vertices $V_B$ within an undirected distance $r$ of the vertex $v$
  - all the edges between the vertices in $V_B$
Strong Simulation

- Extends dual simulation with a locality condition

- A **ball** $G[v, r]$ is a subset of a graph $G$ containing:
  - all vertices $V_B$ within an undirected distance $r$ of the vertex $v$
  - all the edges between the vertices in $V_B$

- $O(|V|(|V| + (|V_q| + |E_q|))(|V| + |E|))$

- Query graph $Q(V_q, E_q, l)$ matches data graph $G(V, E, l)$ via **strong simulation** if there exists a vertex $v \in V$ s.t.
  1. $Q$ matches $G[v, d_q]$ via dual simulation with match relation $R^b$
  2. $v$ is contained in $R^b$
Example: Dual → Strong Simulation

Query

PM: Product Manager
SD: Software Developer
SA: System Analyst
DB: Database Designer
AI: AI Specialist

Data Graph
Example:
Dual → Strong Simulation

Query

PM: Product Manager
SD: Software Developer
SA: System Analyst
DB: Database Designer
AI: AI Specialist
Example:
Dual $\rightarrow$ Strong Simulation

Query

PM: Product Manager
SD: Software Developer
SA: System Analyst
DB: Database Designer
AI: AI Specialist

Data Graph
Example: Dual → Strong Simulation

Query

PM: Product Manager
SD: Software Developer
SA: System Analyst
DB: Database Designer
AI: AI Specialist

Data Graph
# Models of Computation

<table>
<thead>
<tr>
<th><strong>MapReduce</strong></th>
<th><strong>Bulk Synchronous Parallel (BSP)</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Useful paradigm for large batch processing of datasets</td>
<td>• Computation performed in a series of supersteps</td>
</tr>
<tr>
<td>• Not ideal for many graph algorithms</td>
<td>• Interleaved with communication/synchronization phases</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Vertex-Centric (VC) BSP</strong></th>
<th><strong>Message Passing</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>• Each vertex treated as a processing unit</td>
<td>• Different threads communicate with each other via messages</td>
</tr>
<tr>
<td>• Vertices can communicate with each other to obtain information</td>
<td>• Programmer has more control over communication and synchronization</td>
</tr>
</tbody>
</table>
## Distributed Graph Simulation

| Superstep 1: | If there is any query vertex with the same label:  
|             |   - Set `match` to **true**  
|             |   - Make a match set of potential vertices in the query  
|             |   - Ask children about their status  
|             | Otherwise: Vote to halt |

| Superstep 2: | If `match` is **true**: Reply back to parent with label  
|             | Otherwise: Vote to halt |

| Superstep 3: | If `match` is **true**:  
|             |   - Evaluate match set based on children’s responses  
|             |   - If there are removals from the match set:  
|             |     - Inform parents  
|             |     - Set `match` flag accordingly  
|             |   - Otherwise: Vote to halt  
|             | Otherwise: Vote to halt |

| Superstep 4: | If there is any incoming removal message:  
|             |   - Evaluate match set based on children’s responses  
|             |   - If there are removals from the match set:  
|             |     - Inform parents  
|             |     - Set `match` flag accordingly  
|             |   - Otherwise: Vote to halt  
|             | Otherwise: Vote to halt |
Distributed Graph Simulation

Pattern Graph

Data Graph

1\textsuperscript{st} Superstep

2\textsuperscript{nd} Superstep

3\textsuperscript{rd} Superstep
Other Distributed Graph Algorithms

- **Dual simulation**
  - Similar to graph simulation
  - In addition to storing the match sets of its children, a vertex also stores the match sets of its parents
  - During match set evaluation, a vertex takes both of these into account

- **Strong simulation (optimized)**
  - First run dual simulation to obtain match relation $R$
  - For each matching vertex $v$ in $R$:
    - Create a ball centered at $v$ with radius $d_q$ containing only vertices in $R$
    - Perform dual simulation on the ball
Performance Evaluation

Experimental setup

- Cluster with 12 machines
- Each with two 2Ghz Intel Xeon E5-2620 CPUs, each with six cores
- Ethernet: 1 Gb/s
- Set up HDFS/GPS on all of the machines

| Dataset       | |V|   | |E|   | |I|   |
|---------------|---|----|---|----|---|
| Synthesized   | 100 M | 4 B | 200 |
| uk-2005       | 39 M  | 940 M | 200 |
| enwiki-2005   | 4.2 M | 101 M | 200 |
Runtime Evaluation

Synthesized, $|V| = 10^8$
Runtime Evaluation

uk-2005, $|V| = 3.9 \times 10^7$

![Graph Simulation](image1.png)

- Graph Simulation
- Dual Simulation
- Strict Simulation

Running Time (secs)

# Workers
enwiki-2013, |V| = 4.2 \times 10^6
Speedup Evaluation

Synthesized, $|V| = 10^8$
Speedup Evaluation

uk-2005, $|V| = 3.9 \times 10^7$
Speedup Evaluation

enwiki-2013, $|V| = 4.2 \times 10^6$
Conclusions

- The three algorithms exhibit excellent scalable behavior in terms of speedup and efficiency as we increase the number of workers.

- Distributed implementations (GPS):
  - Graph simulation
  - Dual simulation
  - Strong simulation

- Ongoing work:
  - Distributed implementations with message passing in Akka
  - Our own sequential/distributed isomorphism algorithm
Questions?
Thanks
Depth-First Ball

- Algorithm works in a depth-first fashion. Message is generated at the center which is then propagated through the system for ballSize supersteps.

- The approach results in an exponential number of messages that slows down the whole system and renders the approach impractical.
Breadth-First Ball

- Works on a simple ping-reply model.
- Center vertex starts by sending a ping message to all of its adjacent nodes in the first superstep.
- In the second superstep, all the recipient nodes reply back with their label and the ids of their children and parents.
- Center vertex upon receiving this information in the third superstep, saves the returned labels and then ping the boundary nodes. This process is repeated till we have a ball of size $d_q$. 
Breadth-First Ball

(a) Ball: \{1,7\}, Labels: \{\}\n
(b) Ball: \{1,7,2,3,4\}, Labels: \{1,7\}\n
Ping: \rightarrow p \rightarrow\nReply: \rightarrow r \rightarrow\n
(c) Ball: \{1,7,2,3,4\}, Labels: \{1,7\}\n
(d) Ball: \{1,7,2,3,4\}, Labels: \{1,7,2,3,4\}\n
### Ping-Reply Protocol

- **Ping:** \rightarrow p \rightarrow
- **Reply:** \rightarrow r \rightarrow

---

### Graphs

- **Graph 1:**
  - Ball: \{1,7\}, Labels: \{\}\n  - Ping: \rightarrow p \rightarrow
  - Reply: \rightarrow r \rightarrow

- **Graph 2:**
  - Ball: \{1,7,2,3,4\}, Labels: \{1,7\}\n  - Ping: \rightarrow p \rightarrow
  - Reply: \rightarrow r \rightarrow

- **Graph 3:**
  - Ball: \{1,7,2,3,4\}, Labels: \{1,7\}\n  - Ping: \rightarrow p \rightarrow
  - Reply: \rightarrow r \rightarrow

- **Graph 4:**
  - Ball: \{1,7,2,3,4\}, Labels: \{1,7,2,3,4\}\n  - Ping: \rightarrow p \rightarrow
  - Reply: \rightarrow r \rightarrow
Efficiency

- Calculated as \( \text{Efficiency}_k = \text{Speedup}_k / k \)

Synthesized, \( |V| = 10^8 \)  
uk-2005, \( |V| = 3.9 \times 10^7 \)  
enwiki-2013, \( |V| = 4 \)
Graph Partitioning

- By default, the data graph is partitioned in a round-robin fashion among workers

- The goals of min-cut partitioning are two-fold:
  - to create *well-balanced* partitions
  - to reduce the *inter-partition* edges

- Number of other algorithms have been shown to use min-cut graph partitioning successfully for speed-ups

- METIS\(^1\) – graph partitioning tool
  - Written in C
  - Takes an undirected graph and outputs the partitions

\(^1\) [http://glaros.dtc.umn.edu/gkhome/metis/metis/overview](http://glaros.dtc.umn.edu/gkhome/metis/metis/overview)
Performance Evaluation

- Experimental Setup
  - Cluster with 5 machines
  - Each with two 2Ghz Intel Xeon E5-2620 CPU, each with six cores
  - Ethernet: 1 Gb/sec

- Setup HDFS/GPS on all the machines

- Datasets:
  - Labels \((l) = 200\)

| Datasets     | \(|V|\)  | \(|E|\)          |
|--------------|---------|-----------------|
| Synthesized  | \(10^7\)| \(251 \times 10^6\) |
| uk-2002      | \(1.8 \times 10^7\)| \(298 \times 10^6\) |
Results - Runtime

- Min-cut
- Round-robin

Graph Simulation
- Running Time (secs)
- Query Size

Dual Simulation
- Running Time (secs)
- Query Size

Strict Simulation
- Running Time (secs)
- Query size

Synthesized Dataset, $|V| = 10^7$, $\alpha = 1.2$
Results - Runtime

uk-2002, |V| = 1.8x10^7

Graph Simulation

Dual Simulation

Strict Simulation
Results – Network I/O

Graph Simulation

Dual Simulation

Strict Simulation

Synthesized Dataset, $|V|=10^7$, $\alpha=1.2$
Results – Network I/O

uk-2002, $|V| = 1.8 \times 10^7$

Graph Simulation

Dual Simulation

Strict Simulation
Planned Pattern Matching Implementations

<table>
<thead>
<tr>
<th></th>
<th>Sequential</th>
<th>Distributed (GPS)</th>
<th>Distributed (Akka VC)</th>
<th>Distributed (Akka MP)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Graph Simulation</strong></td>
<td>0.581 s</td>
<td>2.924 s</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Dual Simulation</strong></td>
<td>0.642 s</td>
<td>3.403 s</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Strong Simulation</strong></td>
<td>----------</td>
<td>8.453 s</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Subgraph Iso. (ours)</strong></td>
<td>2.837 s</td>
<td>-- --</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Subgraph Iso. (VF2)</strong></td>
<td>----------</td>
<td>-- --</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>