Speeding Up Algorithms on Compressed Web Graphs

Chinmay Karande (Georgia Institute of Technology)
Kumar Chellapilla (Microsoft Live Labs)
Reid Andersen (Microsoft Live Labs)

WSDM 2009
Outline

1 Fast Algorithms for Compressed Graphs
   • Graph Compression
     • Adjacency Matrix Multiplication on Compressed Graphs
     • Adapting PageRank Markov Chain to Compressed Graphs
     • Other algorithms
   • Implementation Results
WWW Graph

- A gold-mine of important information.
- Webpages are nodes, hyperlinks are edges.
A gold-mine of important information.

Webpages are nodes, hyperlinks are edges.

HUGE dataset: \(~ 1\) Trillion pages
WWW Graph

- A gold-mine of important information.
- Webpages are nodes, hyperlinks are edges.
- **HUGE** dataset: \( \sim \) 1 Trillion pages
- Graph Algorithms:
  - Importance metrics: PageRank, HITS, SALSA...
  - Finding paths
  - Clustering
Structural Graph Compression

- Replace a dense subgraph by a sparse one, such that:
  - Maintain connectivity
  - Decompressible
  - Maintain ‘structure’
Clique-Star Compression
Clique-Star Compression: Terminology

Real Node

Real Node
Clique-Star Compression: Terminology

Real Node

Virtual Node

Real Node
Clique-Star Compression: Terminology

Virtual Node

Real Node

Virtual Edge
Clique-Star Compression

- Compression performed in **phases**: In each phase compress edge-disjoint cliques.
- In each phase, virtual edges may become longer by one.
- Diminishing returns on number of phases: $\sim 6$ to $8$ phases yield $10$ fold compression. [G. Buehrer, K. Chellapilla]
Problem Statement

**Problem**: Given $G'$ compressed from $G$, how do we perform computations on $G'$ so as to infer properties of $G$?

- How to determine metrics like:
  - PageRank
  - HITS
  - SALSA

of $G$ without decompressing $G'$?
Example: PageRank

PageRank can be viewed as:
- Repeated multiplication by adjacency matrix (with adjustments)
  - We need a Black-Box procedure to multiply a vector by adjacency matrix of $G$ given only $G'$. 
Example: PageRank

PageRank can be viewed as:

- Repeated multiplication by adjacency matrix (with adjustments)
  - We need a Black-Box procedure to multiply a vector by adjacency matrix of $G$ given only $G'$.
- Steady state of a Markov Chain
  - We need a Markov Chain on $G'$ that ‘mimics’ the PageRank MC on $G$. 
Outline

1. Fast Algorithms for Compressed Graphs
   - Graph Compression
   - Adjacency Matrix Multiplication on Compressed Graphs
   - Adapting PageRank Markov Chain to Compressed Graphs
   - Other algorithms
   - Implementation Results
Adjacency Matrix Multiplication: Nuts and Bolts

\[ y = E^T \cdot x \quad x \in \mathbb{R}^n \]

\[ y_v = x_{u_1} + x_{u_2} + x_{u_3} + x_{u_4} + x_{u_5} \]
Adjacency Matrix Multiplication on Compressed Graph

\[ \mathbf{y} = \mathbf{E}^T \cdot \mathbf{x} \quad \mathbf{x} \in \mathbb{R}^n \]

\[ \mathbf{y}_v = \mathbf{x}_{u_1} + \mathbf{x}_{u_2} + \mathbf{x}_{u_3} + \mathbf{x}_{u_4} + \mathbf{x}_{u_5} \]
Adjacency Matrix Multiplication on Compressed Graph

\[ y = E^T \cdot x \quad x \in \mathbb{R}^n \]

\[ y_v = x_{u_1} + x_{u_2} + y_w \]
Adjacency Matrix Multiplication on Compressed Graph: Dependencies

Consider a virtual edge $u \rightarrow w_1 \rightarrow \ldots \rightarrow w_k \rightarrow v$:

- $y_v$ depends upon $y_{w_k}$
- $y_{w_k}$ depends upon $y_{w_{k-1}} \ldots$
Subgraph induced by edges incident on virtual nodes is a forest. [G. Buehrer, K. Chellapilla]
⇒ There exists a way to resolve dependencies.
Acyclic Dependencies on Virtual Nodes

Subgraph induced by edges incident on virtual nodes is a forest. [G. Buehrer, K. Chellapilla]
⇒ There exists a way to resolve dependencies.

\[ \textbf{while } y \text{ is undefined on some virtual nodes do} \]
  \[ \text{Pick virtual node } w \text{ such that } y \text{ is defined on all virtual predecessors of } w. \]
  \[ \text{Compute and define } y_w. \]
\[ \textbf{end while} \]
Solution

Permute virtual nodes in the order of dependencies
Solution

Permute virtual nodes in the order of dependencies

- Practical Considerations
  - Sequential File Access
  - Synchronous algorithm
  - For SALSA: Inverted adjacency required for virtual nodes.
Solution

Permute virtual nodes in the order of dependencies

- Practical Considerations
  - Sequential File Access
  - Synchronous algorithm
  - For SALSA: Inverted adjacency required for virtual nodes.
  - **Speed-up almost matches the storage reduction ratio.**
Outline

1. Fast Algorithms for Compressed Graphs
   - Graph Compression
   - Adjacency Matrix Multiplication on Compressed Graphs
   - Adapting PageRank Markov Chain to Compressed Graphs
   - Other algorithms
   - Implementation Results
PageRank Scheme

PageRank is a Markov Chain:

- With probability $\alpha$, perform a uniform ‘jump’.
- $Pr[u \rightarrow v] = (1 - \alpha)\frac{1}{|\delta_{out}(u)|}$
PageRank on Compressed Graph

\[ Pr[X_t = u_i] = p_i \]

\[ Pr[X_{t+1} = v_i | p_1, p_2, p_3] = \frac{1}{|\delta(u_1)|} + \frac{1}{|\delta(u_2)|} + \frac{1}{|\delta(u_3)|} \]
PageRank on Compressed Graph

\[ Pr[X_t = u_i] = p_i \]

\[ Pr[X_{t+1} = v_i | p_1, p_2, p_3] = \frac{1}{|\delta(u_1)|} + \frac{1}{|\delta(u_2)|} + \frac{1}{|\delta(u_3)|} \]

\[ = \frac{1}{|\delta(w)|} \sum_i \frac{|\delta(w)|}{|\delta(u_i)|} \]
Defining the ‘reach’ of a node

\[ \Delta(u) = \begin{cases} 
1 & \text{If } u \text{ is real} \\
\sum_{uv \in E'} \Delta(v) & \text{If } u \text{ is virtual}
\end{cases} \]
Illustration of $\Delta$ function

\[ \Delta(u) \; = \; 1 \]
\[ \Delta(v) \; = \; 5 \]
\[ \Delta(w) \; = \; 3 \]
Defining the true out-degree of a node

$$\Gamma(u) = \sum_{uv \in E'} \Delta(v)$$

If $G'$ is compressed from $G$ then:
- For real $u$, $\Gamma(u)$ is the out-degree of $u$ in $G$.
- For virtual $u$, $\Gamma(u) = \Delta(u)$. 

Illustration of $\Gamma$ function

\[ \Gamma(u) = 7 \]
\[ \Gamma(v) = 5 \]
\[ \Gamma(w) = 3 \]
With probability $\alpha$, perform a uniform ‘jump’ but don’t jump to and from virtual nodes.
With probability $\alpha$, perform a uniform ‘jump’ but don’t jump to and from virtual nodes.

$$Pr[u \rightarrow v] = \begin{cases} 
(1 - \alpha) \frac{\Delta(v)}{\Gamma(u)} & \text{If } u \text{ is real} \\
\frac{\Delta(v)}{\Gamma(u)} & \text{If } u \text{ is virtual}
\end{cases}$$
Correctness theorem

**Theorem**

*If $G'$ is compressed from $G$ and $p'$, $p$ are respective PageRank vectors, then for every real node $u$, $p'(u) = \epsilon p(u)$.***
Proof.

Split the compression from $G$ to $G'$ in phases:

\[ G = G_0 \succ G_1 \succ \ldots \succ G_k = G' \]

Let $p_i$ be the steady state of (modified) PageRank on $G_i$.

Conclusion: For $u \in V(G_i)$, $p_i(u)$'s and $p_{i+1}(u)$'s satisfy the same equations \[ p_{i+1}(u) = \epsilon_{i+1} p_i(u) \]

$\epsilon = \epsilon_1 \cdot \epsilon_2 \cdot \ldots \cdot \epsilon_k$
Proof

Proof.

- Split the compression from $G$ to $G'$ in phases:

$$G = G_0 \succ G_1 \succ \ldots \succ G_k = G'$$

Let $p_i$ be the steady state of (modified) PageRank on $G_i$. 

\[ \epsilon = \epsilon_1 \cdot \epsilon_2 \cdot \ldots \cdot \epsilon_k \]
Proof

Split the compression from $G$ to $G'$ in phases:

$$G = G_0 \succ G_1 \succ ... \succ G_k = G'$$

Let $p_i$ be the steady state of (modified) PageRank on $G_i$.

Conclusion: For $u \in V(G_i)$, $p_i(u)$'s and $p_{i+1}(u)$'s satisfy the same equations $\Rightarrow p_{i+1}(u) = \epsilon_{i+1}p_i(u)$
Proof

Proof.

Split the compression from $G$ to $G'$ in phases:

$$G = G_0 \succ G_1 \succ \ldots \succ G_k = G'$$

Let $p_i$ be the steady state of (modified) PageRank on $G_i$.

Conclusion: For $u \in V(G_i)$, $p_i(u)$’s and $p_{i+1}(u)$’s satisfy the same equations $\Rightarrow p_{i+1}(u) = \epsilon_{i+1}p_i(u)$

$\epsilon = \epsilon_1 \cdot \epsilon_2 \cdot \ldots \cdot \epsilon_k$
Run (modified) PageRank on compressed graph, and normalize the values on real nodes to unit norm.
**Precision Theorem**

Theorem

\[ \epsilon \geq 2^{-k} \]

where \( k \) is the length of the longest virtual edge.
Precision Theorem

\[ \epsilon \geq 2^{-k} \]

where \( k \) is the length of the longest virtual edge.

Proof.

- Split the compression from \( G \) to \( G' \) in phases:
  \[ G = G_0 \succ G_1 \succ \ldots \succ G_k = G' \]
  Let \( p_i \) be the steady state of (modified) PageRank on \( G_i \).

- To prove: \( \epsilon_i \geq 1/2 \).
Precision Theorem

\[ \epsilon \geq 2^{-k} \]

where \( k \) is the length of the longest virtual edge.

Proof.

- Split the compression from \( G \) to \( G' \) in phases:
  \[
  G = G_0 \succ G_1 \succ \ldots \succ G_k = G'
  \]
  Let \( p_i \) be the steady state of (modified) PageRank on \( G_i \).
- To prove: \( \epsilon_i \geq 1/2 \).
- Follows from the fact that
  \[
  \sum_{u \in V(G_i)} p_i(u) + \sum_{u \in Q} p_i(u) = 1
  \]
Solution

Run (modified) PageRank on compressed graph, and normalize the values on real nodes to unit norm.

- Practical Considerations:
  - Modified only the weights - Can run any existing PageRank implementation almost unchanged.
  - Sequential File Access
  - Asynchronous: Distributed computing feasible.
  - Convergence may be slower due to longer path lengths.
Solution

Run (modified) PageRank on compressed graph, and normalize the values on real nodes to unit norm.

- Practical Considerations:
  - Modified only the weights - Can run any existing PageRank implementation almost unchanged.
  - Sequential File Access
  - Asynchronous: Distributed computing feasible.
  - Convergence may be slower due to longer path lengths.
  - Speed-up per iteration almost matches the storage reduction ratio!
Fast Algorithms for Compressed Graphs

- Graph Compression
- Adjacency Matrix Multiplication on Compressed Graphs
- Adapting PageRank Markov Chain to Compressed Graphs
- Other algorithms
- Implementation Results
Both the Synchronous and Asynchronous methods can be adapted for SALSA.

- In-link counterparts of $\Delta$ and $\Gamma$ required.
Shortest Paths: BFS

- Simply define edge weights as:

\[ w(u, v) = \begin{cases} 
1 & \text{If } v \text{ is real} \\
0 & \text{If } v \text{ is virtual}
\end{cases} \]

- Use a Deque.
Outline

1. Fast Algorithms for Compressed Graphs
   - Graph Compression
   - Adjacency Matrix Multiplication on Compressed Graphs
   - Adapting PageRank Markov Chain to Compressed Graphs
   - Other algorithms
   - Implementation Results
Experiments: Proof of Concept

If $\beta$ is the reduction ratio in the number of edges, we cannot hope for the programs to run $\beta$ times faster.

$O(|V|)$ operations such as:

- Allocating variables
- Copying and zeroing values between iterations

bring down the speed-up to a small extent.
PageRank on eu-2005

- Uncompressed graph
  No. of nodes: 862,664
  No. of edges: 19,235,140

- Compressed graph has $\beta = 4.34$
  No. of nodes: 1,196,536
  No. of edges: 4,429,375

<table>
<thead>
<tr>
<th></th>
<th>Uncompressed</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time/iteration (sec)</td>
<td>5.37</td>
<td>1.58</td>
<td>1.50</td>
</tr>
<tr>
<td>No. of iterations</td>
<td>19</td>
<td>19</td>
<td>50</td>
</tr>
<tr>
<td>Speed-up</td>
<td>1</td>
<td>3.40</td>
<td>1.36</td>
</tr>
</tbody>
</table>
PageRank on uk-2005

- Uncompressed graph
  - No. of nodes: 39,459,925
  - No. of edges: 936,364,282

- Compressed graph has $\beta = 6.18$
  - No. of nodes: 47,482,140
  - No. of edges: 151,456,024

<table>
<thead>
<tr>
<th></th>
<th>Uncompressed</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time/iteration (sec)</td>
<td>264.40</td>
<td>59.52</td>
<td>59.15</td>
</tr>
<tr>
<td>No. of iterations</td>
<td>21</td>
<td>21</td>
<td>53</td>
</tr>
<tr>
<td>Speed-up</td>
<td>1</td>
<td>4.44</td>
<td>2.53</td>
</tr>
</tbody>
</table>
SALSA on eu-2005

- Uncompressed graph
  - No. of nodes: 862,664
  - No. of edges: 19,235,140

- Compressed graph has $\beta = 4.34$
  - No. of nodes: 1,196,536
  - No. of edges: 4,429,375

<table>
<thead>
<tr>
<th>Time/iteration (sec)</th>
<th>Uncompressed</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>No. of iterations</td>
<td>91</td>
<td>91</td>
<td>100</td>
</tr>
<tr>
<td>Speed-up</td>
<td>1</td>
<td>2.31</td>
<td>2.70</td>
</tr>
<tr>
<td>Storage Reduction</td>
<td>1</td>
<td>2.36</td>
<td>3.21</td>
</tr>
</tbody>
</table>
PageRank on uk-2005

- Uncompressed graph
  No. of nodes: 39,459,925
  No. of edges: 936,364,282

- Compressed graph has $\beta = 6.18$
  No. of nodes: 47,482,140
  No. of edges: 151,456,024

<table>
<thead>
<tr>
<th></th>
<th>Uncompressed</th>
<th>Synchronous</th>
<th>Asynchronous</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time/iteration (sec)</td>
<td>276.09</td>
<td>72.93</td>
<td>88.69</td>
</tr>
<tr>
<td>No. of iterations</td>
<td>104</td>
<td>104</td>
<td>124</td>
</tr>
<tr>
<td>Speed-up</td>
<td>1</td>
<td>3.11</td>
<td>3.18</td>
</tr>
<tr>
<td>Storage Reduction</td>
<td>1</td>
<td>3.47</td>
<td>4.54</td>
</tr>
</tbody>
</table>
Thank you!