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

WWW Graph

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

WWW Graph

- A gold-mine of important information.
- Webpages are nodes, hyperlinks are edges.
- HUGE dataset: $\sim$ 1 Trillion pages

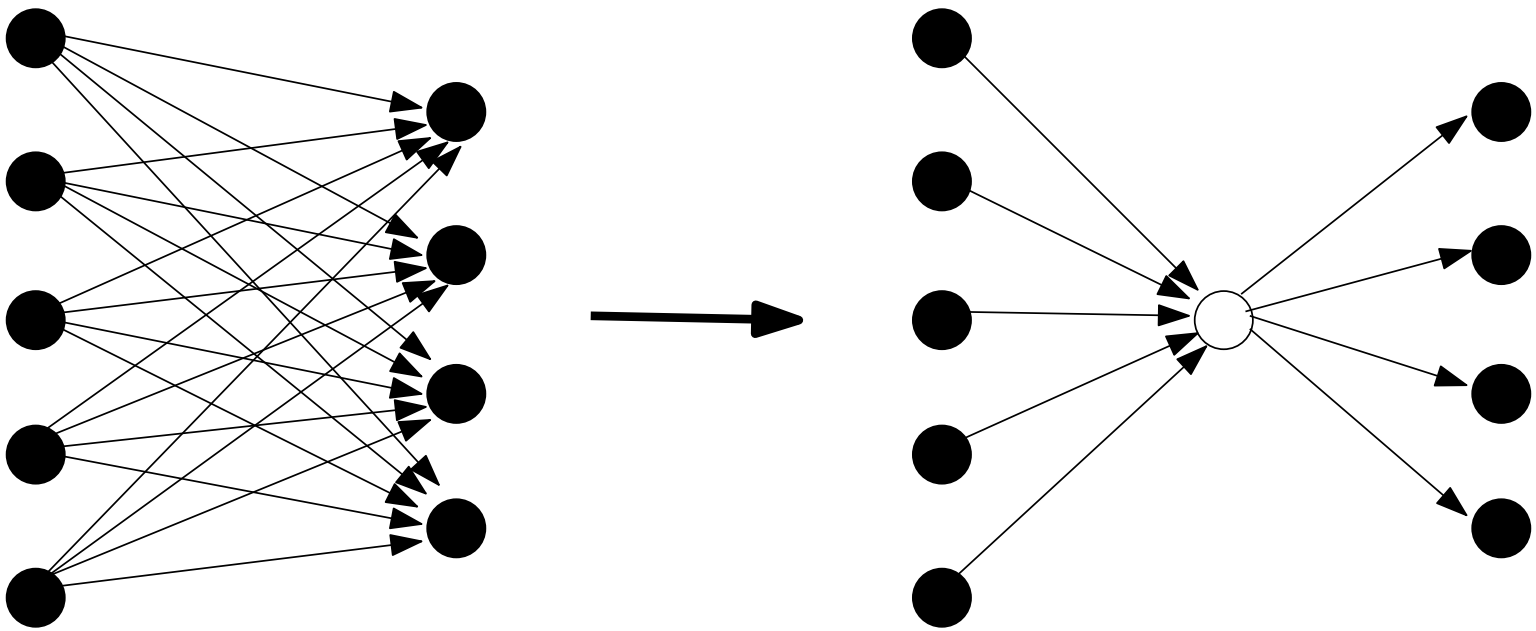
WWW Graph

- 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

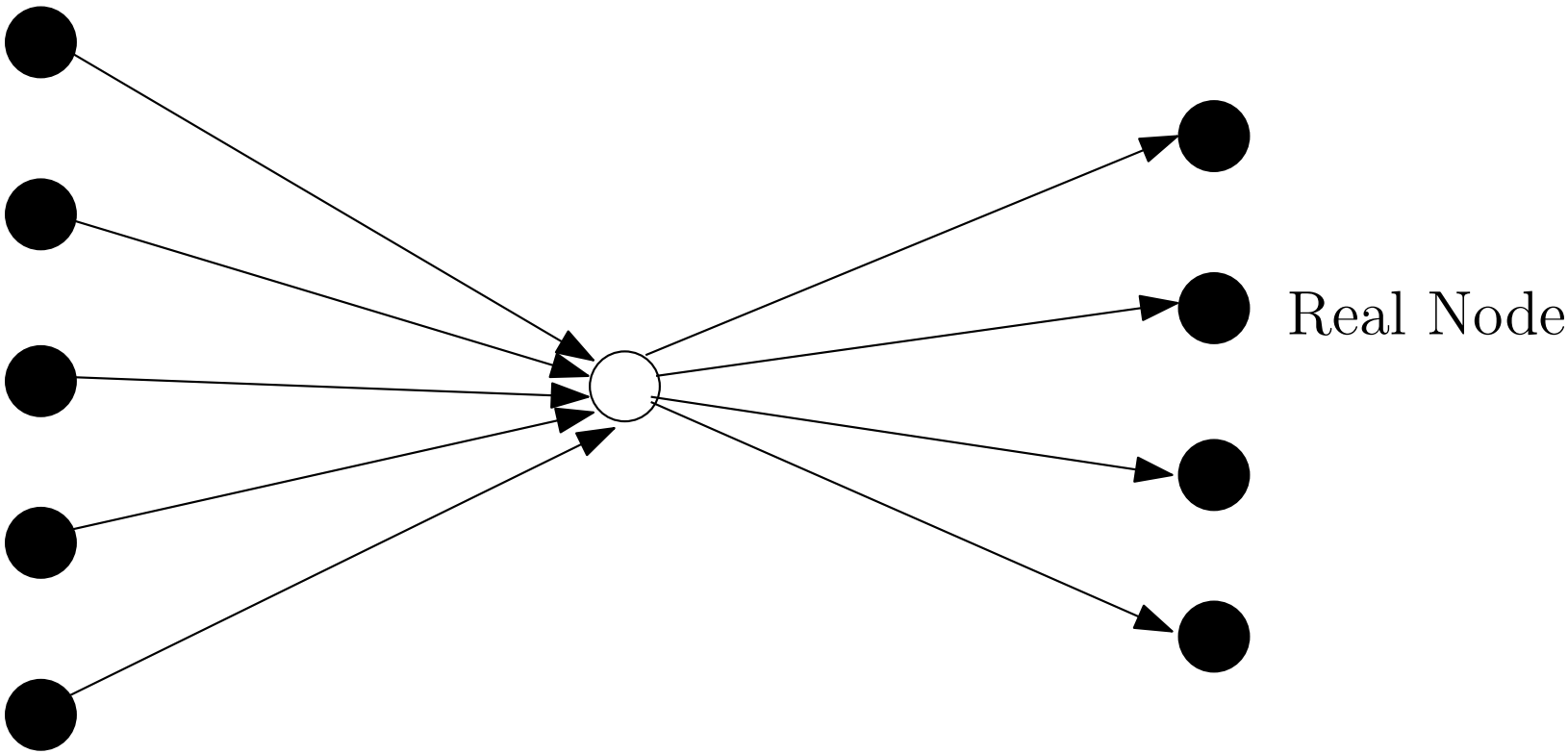
Structural Graph Compression

- Replace a dense subgraph by a sparse one, such that:
  - Maintain connectivity
  - Decompressible
  - Maintain 'structure'

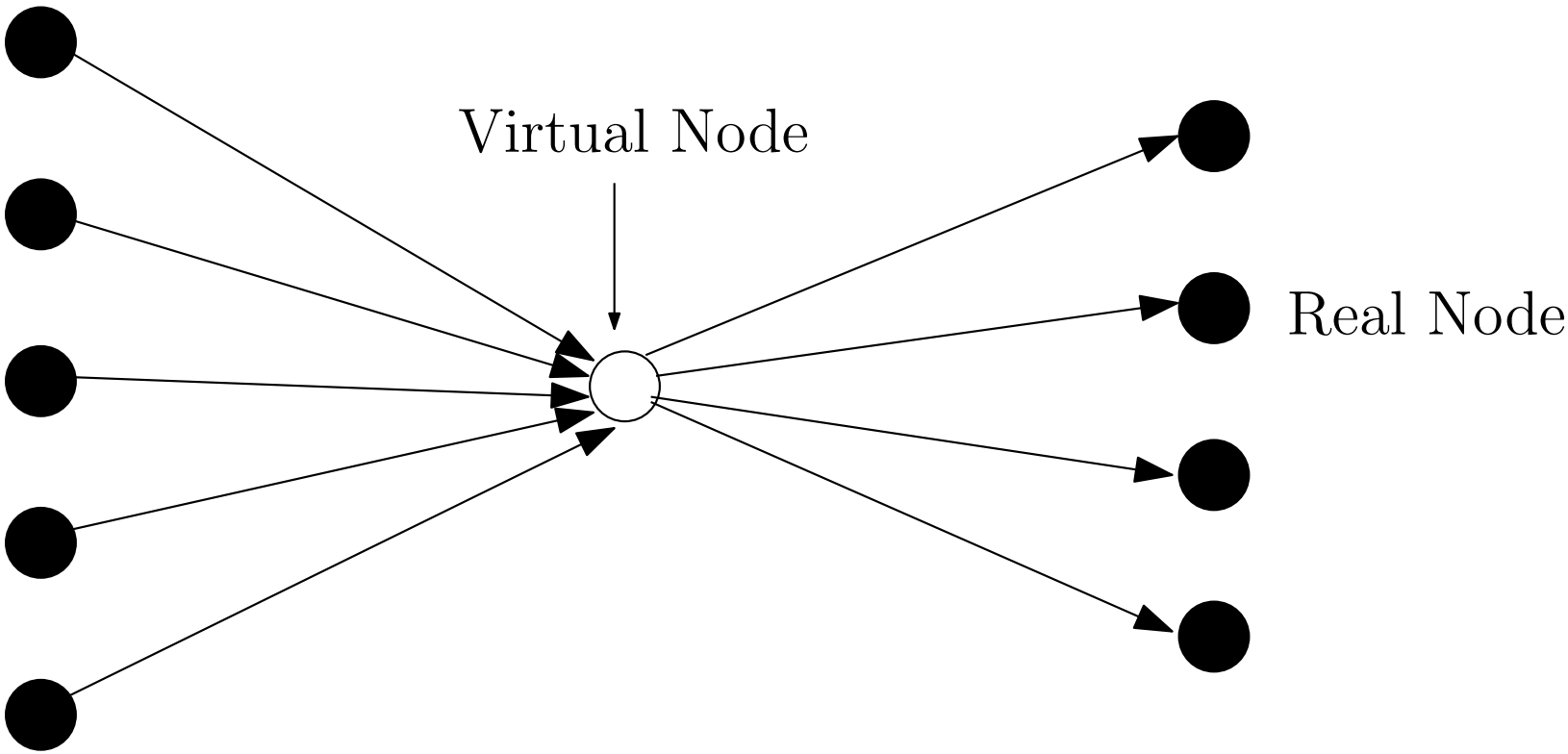
Clique-Star Compression



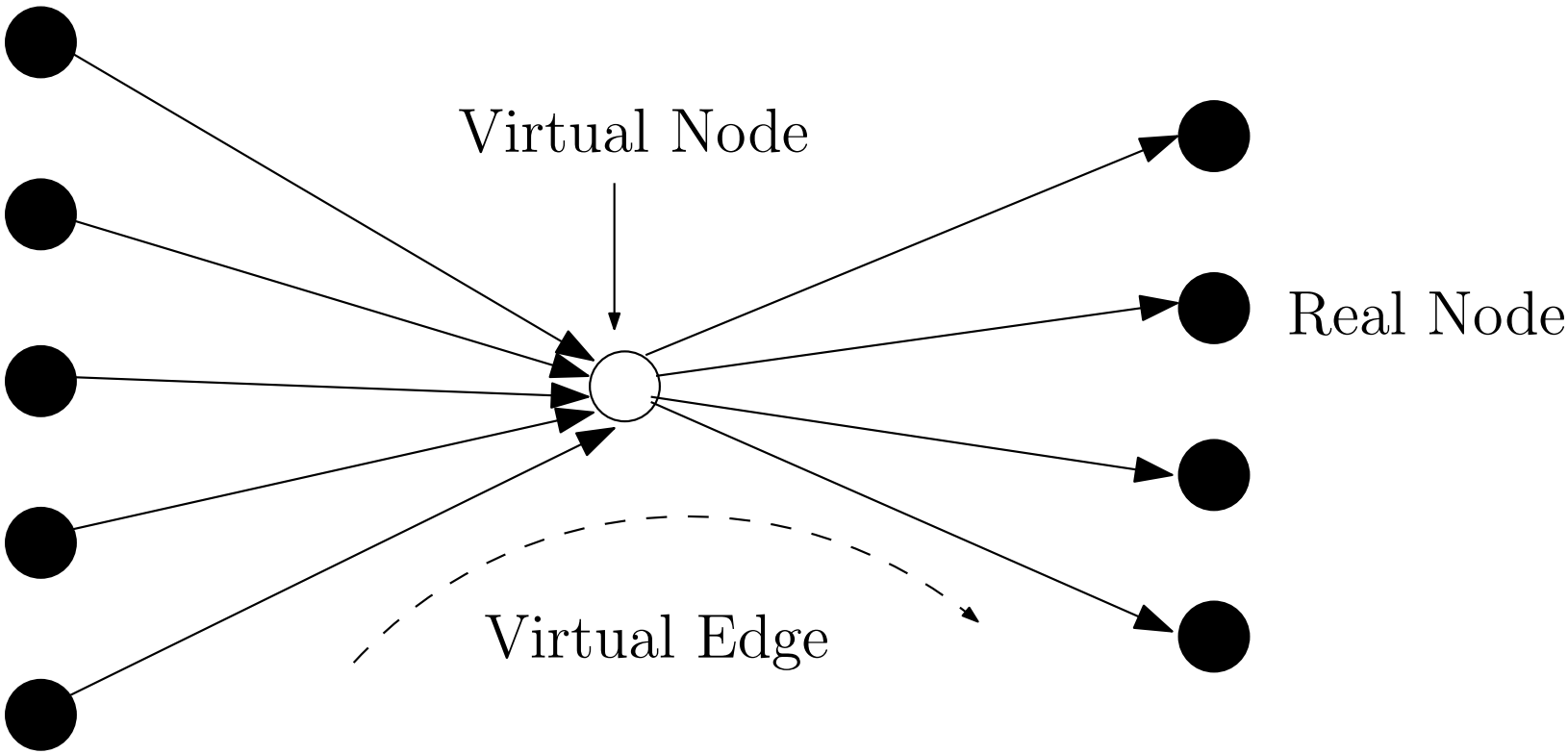
Clique-Star Compression: Terminology



Clique-Star Compression: Terminology



Clique-Star Compression: Terminology



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

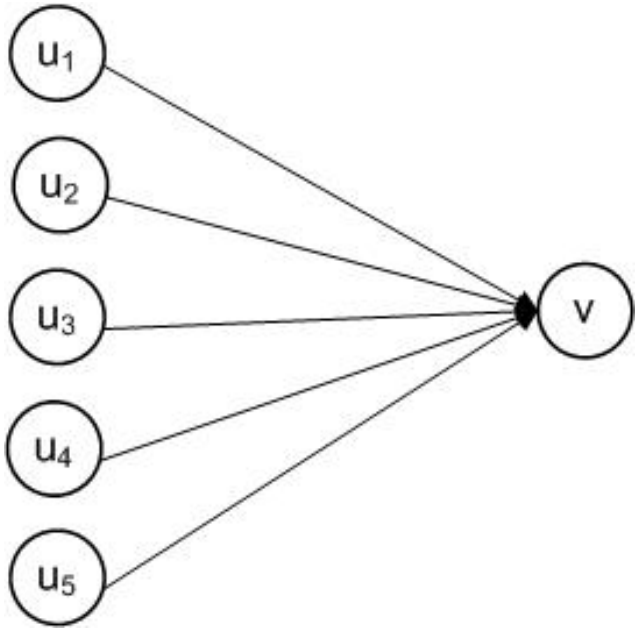
PageRank can be viewed as:

- 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

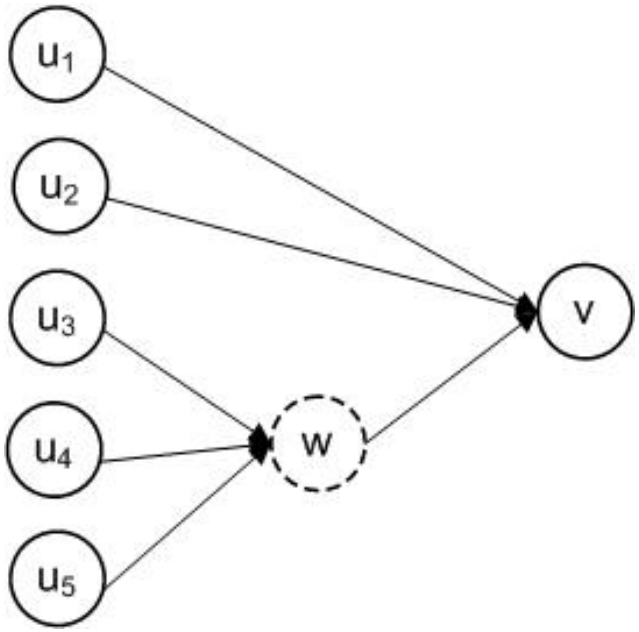
Adjacency Matrix Multiplication: Nuts and Bolts



$$\mathbf{y} = E^T \cdot \mathbf{x} \qquad x \in \mathbb{R}^n$$

$$\mathbf{y}_v = \mathbf{x}_{u_1} + \mathbf{x}_{u_2} + \mathbf{x}_{u_3} + \mathbf{x}_{u_4} + \mathbf{x}_{u_5}$$

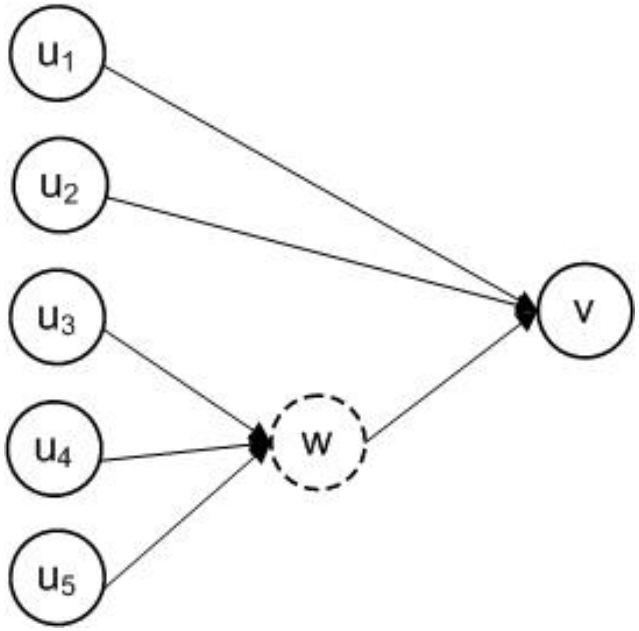
Adjacency Matrix Multiplication on Compressed Graph



$$\mathbf{y} = E^T \cdot \mathbf{x} \qquad x \in \mathbb{R}^n$$

$$\mathbf{y}_v = \mathbf{x}_{u_1} + \mathbf{x}_{u_2} + \underline{\mathbf{x}_{u_3} + \mathbf{x}_{u_4} + \mathbf{x}_{u_5}}$$

Adjacency Matrix Multiplication on Compressed Graph



$$\mathbf{y} = E^T \cdot \mathbf{x} \qquad x \in \mathbb{R}^n$$

$$\mathbf{y}_v = \mathbf{x}_{u_1} + \mathbf{x}_{u_2} + \mathbf{y}_w$$

Adjacency Matrix Multiplication on Compressed Graph: Dependencies

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

- $\mathbf{y}_v$ depends upon $\mathbf{y}_{w_k}$
- $\mathbf{y}_{w_k}$ depends upon $\mathbf{y}_{w_{k-1}}$ ...

Acyclic Dependencies on Virtual Nodes

Subgraph induced by edges incident on virtual nodes is a forest. [G. Buehrer, K. Chellapilla]

$\Rightarrow$ 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]

$\Rightarrow$ There exists a way to resolve dependencies.

while $\mathbf{y}$ is undefined on some virtual nodes **do**

Pick virtual node w such that $\mathbf{y}$ is defined on all virtual predecessors of w.

Compute and define $\mathbf{y}_w$.

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

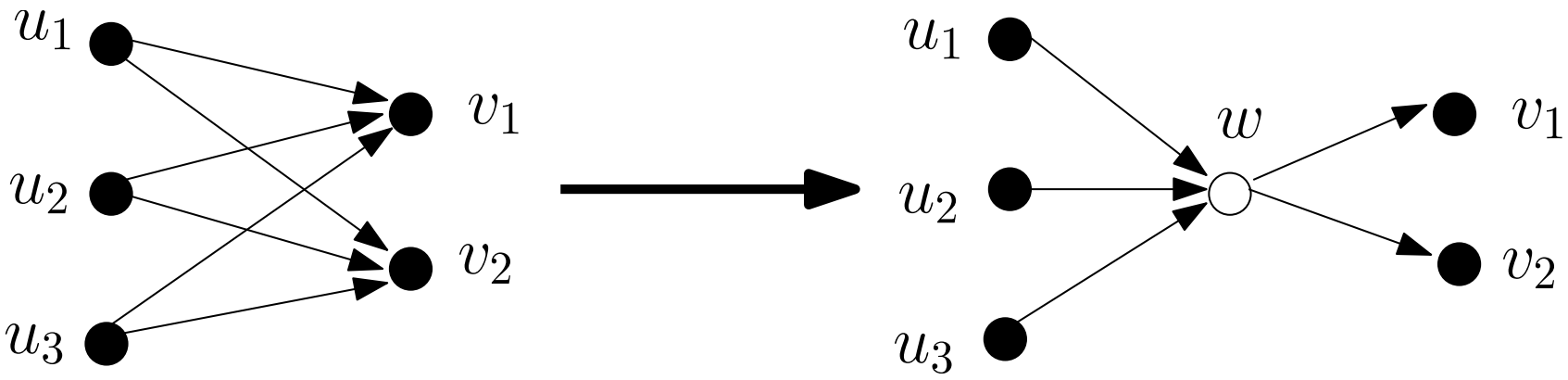
1. Fast Algorithms for Compressed Graphs
   - Graph Compression
   - Adjacency Matrix Multiplication on Compressed Graphs
   - Adapting PageRank Markov Chain to Compressed Graphs
   - Other algorithms
   - Implementation Results

PageRank Scheme

PageRank is a Markov Chain:

- With probability α, perform a uniform 'jump'.
- $Pr[u \rightarrow v] = (1 - \alpha)\frac{1}{|\delta_{out}(u)|}$

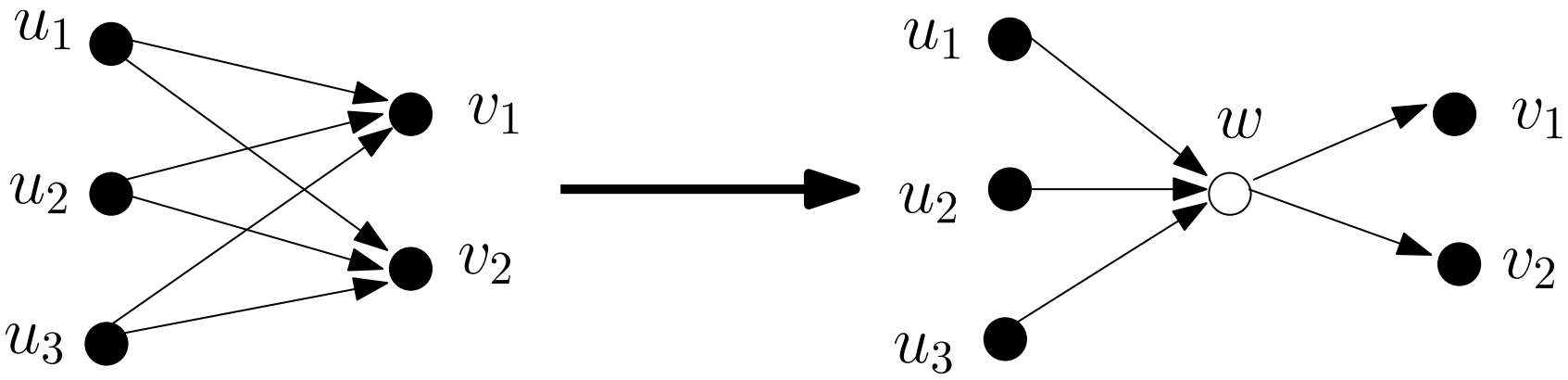
PageRank on Compressed Graph



$Pr[X_t = u_i] = p_i$

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

PageRank on Compressed Graph



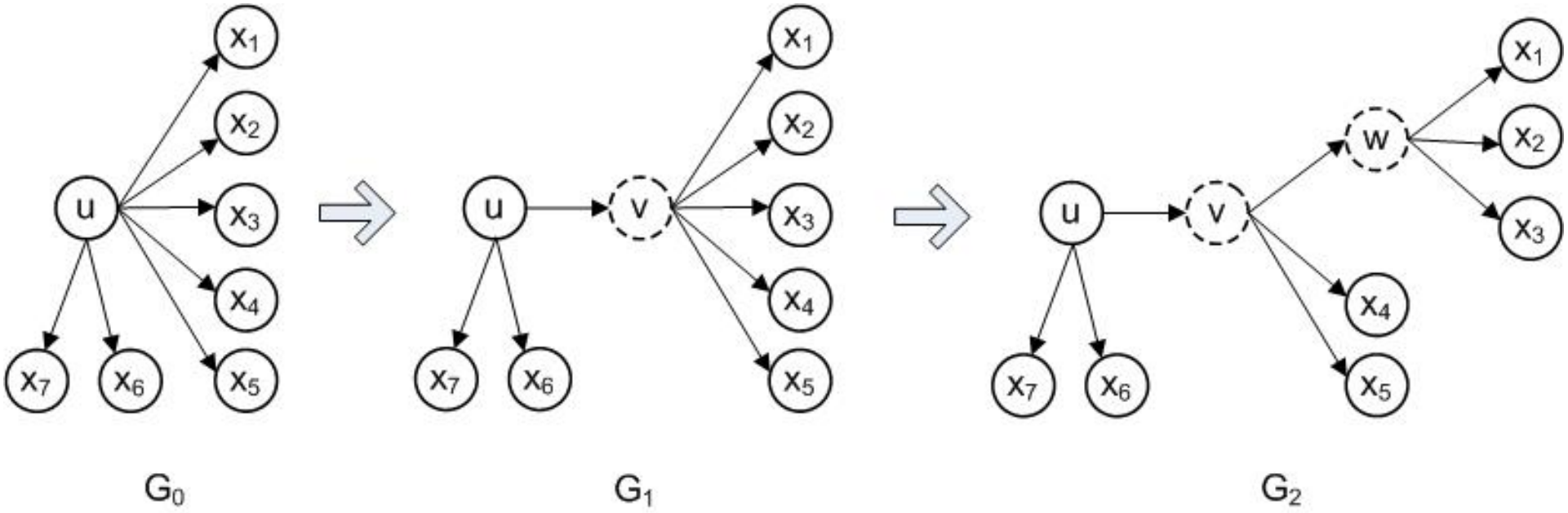
$Pr[X_t = u_i] = p_i$

$$\begin{aligned} Pr[X_{t+1} &= v_i \mid p_1, p_2, p_3] = \frac{1}{|\delta(u_1)|} + \frac{1}{|\delta(u_2)|} + \frac{1}{|\delta(u_3)|} \\ &= \frac{1}{|\delta(w)|} \sum_i \frac{|\delta(w)|}{|\delta(u_i)|} \end{aligned}$$

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 Δ function



$$\begin{aligned} \Delta(u) &= 1 \\ \Delta(v) &= 5 \\ \Delta(w) &= 3 \end{aligned}$$

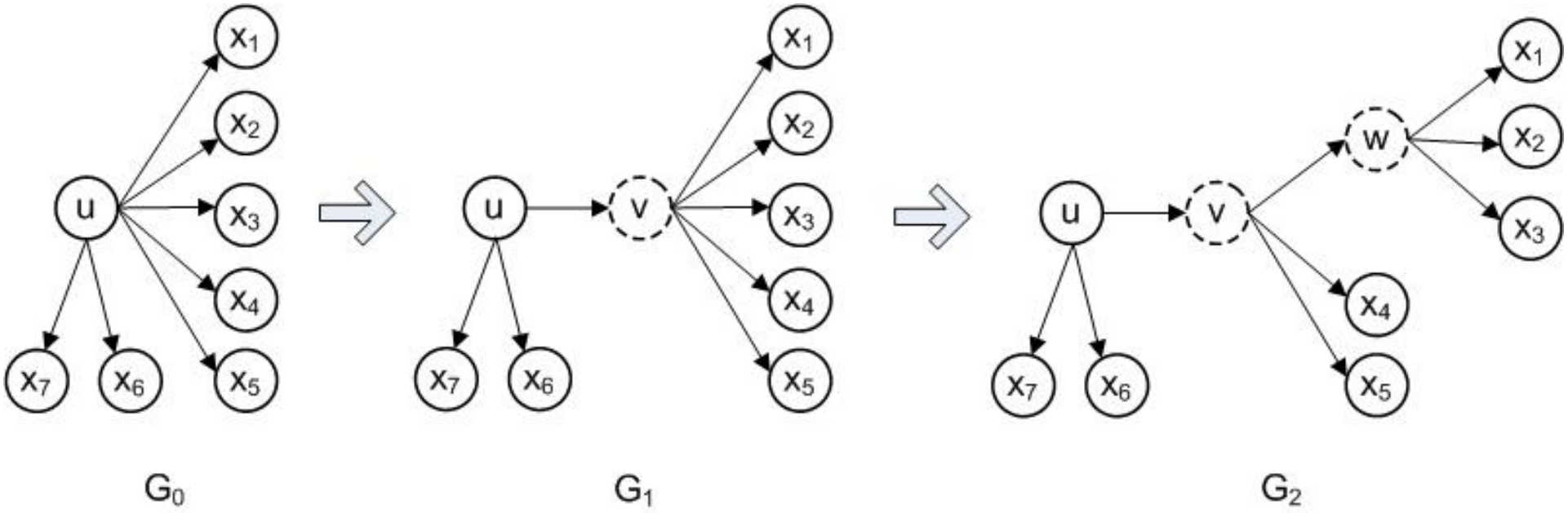
Defining the true out-degree of a node

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

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 Γ function



$$\begin{aligned} \Gamma(u) &= 7 \\ \Gamma(v) &= 5 \\ \Gamma(w) &= 3 \end{aligned}$$

PageRank on Compressed Graph

- With probability α, perform a uniform 'jump' **but** don't jump to and from virtual nodes.

PageRank on Compressed Graph

- With probability α, perform a uniform 'jump' **but** don't jump to and from virtual nodes.
- $$Pr[u \rightarrow v] = \begin{cases} (1-\alpha)\dfrac{\Delta(v)}{\Gamma(u)} & \text{If } u \text{ is real} \\ \dfrac{\Delta(v)}{\Gamma(u)} & \text{If } u \text{ is virtual} \end{cases}$$

Correctness theorem

Theorem

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

Proof

Proof.

Proof

Proof.

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

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

Let $\mathbf{p}_i$ be the steady state of (modified) PageRank on G_i.

Proof

Proof.

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

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

  Let $\mathbf{p}_i$ be the steady state of (modified) PageRank on G_i.
- Conclusion: For $u \in V(G_i)$, $\mathbf{p_i}(u)$'s and $\mathbf{p_{i+1}}(\mathbf{u})$'s satisfy the same equations $\Rightarrow \mathbf{p_{i+1}}(u) = \epsilon_{i+1}\mathbf{p_i}(u)$

Proof

Proof.

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

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

  Let $\mathbf{p}_i$ be the steady state of (modified) PageRank on G_i.
- Conclusion: For $u \in V(G_i)$, $\mathbf{p_i}(u)$'s and $\mathbf{p_{i+1}}(\mathbf{u})$'s satisfy the same equations $\Rightarrow \mathbf{p_{i+1}}(u) = \epsilon_{i+1}\mathbf{p_i}(u)$
- $\epsilon = \epsilon_1 \cdot \epsilon_2 \cdot \ldots \cdot \epsilon_k$

Solution

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

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 $\mathbf{p}_i$ be the steady state of (modified) PageRank on G_i.
- To prove: $\epsilon_i \geq 1/2$.

Precision Theorem

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 $\mathbf{p}_i$ be the steady state of (modified) PageRank on G_i.
- To prove: $\epsilon_i \geq 1/2$.
- Follows from the fact that

$$\sum_{u \in V(G_i)} \mathbf{p}_i(u) + \sum_{u \in Q} \mathbf{p}_i(u) = 1$$

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.!**

Outline

SALSA: Stochastic Approach to Link-Structure Analysis

Both the Synchronous and Asynchronous methods can be adapted for SALSA.

- In-link counterparts of Δ and Γ 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 β is the reduction ratio in the number of edges, we cannot hope for the programs to run β 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

	Uncompressed	Synchronous	Asynchronous
Time/iteration (sec)	5.37	1.58	1.50
No. of iterations	19	19	50
Speed-up	1	3.40	1.36

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

	Uncompressed	Synchronous	Asynchronous
Time/iteration (sec)	264.40	59.52	59.15
No. of iterations	21	21	53
Speed-up	1	4.44	2.53

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

	Uncompressed	Synchronous	Asynchronous
Time/iteration (sec)	5.48	2.37	1.97
No. of iterations	91	91	100
Speed-up	1	2.31	2.70
Storage Reduction	1	2.36	3.21

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

	Uncompressed	Synchronous	Asynchronous
Time/iteration (sec)	276.09	72.93	88.69
No. of iterations	104	104	124
Speed-up	1	3.11	3.18
Storage Reduction	1	3.47	4.54

Thank you.!