Representing Web Graphs
S. Raghaven & H. Garcia-Molina
Computer Science Dept., Stanford University

Amr El-Helw
CS856
University of Waterloo
aelhelw@cs.uwaterloo.ca
Outline

- Motivation
- Contribution
- Introduction
- S-Node Representation
- Experimental Results
- Summary
Motivation

- Efficient traversal of huge Web graphs is a challenging problem.
- The lack of a schema to describe the structure of Web graphs.
- Naive graph representation schemes can increase query execution time.
Contribution

- Proposing a new representation for Web graphs, the “S-Node” representation.
- Demonstrating that S-Node representations are highly space-efficient.
- Showing, by experiment, that S-Node representations can significantly reduce query execution times.
Introduction

- Web Repositories:
  - Large special-purpose collections of Web pages and associated indexes.

- Examples:
  - Research repositories (e.g. Stanford WebBase, the Internet Archive)
  - Commercial search engines (e.g. Google, Altavista)
### Introduction

#### Access to Web Repositories

<table>
<thead>
<tr>
<th></th>
<th>Commercial Search Engines</th>
<th>Research Repositories</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Target Audience</strong></td>
<td>Non-expert users</td>
<td>Expert users</td>
</tr>
<tr>
<td><strong>Type of Access</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Access is controlled by a public search interface.</td>
<td>Perform complex analysis, mining and indexing over huge data sets.</td>
</tr>
<tr>
<td></td>
<td>No internal interface (API) is publicly available.</td>
<td>Provide “Bulk” access interface to their content</td>
</tr>
</tbody>
</table>
Introduction

There are kinds of analysis for which both either access mode is unsuitable. They have the following features:

- **Focused Access**
  - It focuses on a small set of pages and associated links (in contrast to a typical mining or analysis task using bulk access).

- **Complex Expressive Queries**
  - It uses predicates on several different properties of pages (e.g. domain, text content), and navigational operations (e.g. pages pointing to other pages).
Introduction

This kind of analysis provides 3 views of the repository:

- A collection of text documents that can be searched and ranked using keywords and/or phrases.
- A navigable directed graph.
- A set of relational tables storing properties (rank, title, domain, …) on which selection, projection and predicates can be applied.
S-Node Representation

- $W_G$ represents the directed Web graph
- Let $P = \{N_1, N_2, ..., N_n\}$ be a partition on the nodes (pages) of $W_G$.
- Some terms of S-Node representation:
  - Supernode graph
  - Intranode graph
  - Positive superedge graph
  - Negative superedge graph
S-Node Representation

Web graph

Supernode graph

Partition $P = \{N_1, N_2, N_3\}$
- $N_1 = \{P_1, P_2\}$
- $N_2 = \{P_3\}$
- $N_3 = \{P_4, P_5\}$

IntraNode
- $P_1$
- $P_2$
- $P_3$
- $P_4$
- $P_5$
S-Node Representation

An S-Node representation of $W_G$, $\text{SNode}(W_G, P)$ can be constructed using all of the following:

- A supernode graph
- A set of intranode graphs
- A set of positive and negative subedge graphs
Building an S-Node Representation

Requirements for the partition P:

- It must produce highly compressible intranode and superedge graphs, to achieve a compact representation.
- For local access queries, the set of pages and links involved must be distributed within a small number of intranode and superedge graphs → Efficient execution by loading only the relevant graphs into main memory.
Building an S-Node Representation

Observations about Web graphs:

- **Link copying.** There are clusters of pages on the Web that have very similar adjacency lists.
- **Domain and URL Locality.** Many links from a page point to other pages on the same domain, and possibly with lexicographically close URLs.
- **Page similarity.** Pages that have very similar adjacency lists (i.e., pages which point to almost the same set of pages) are likely to be related.
Building an S-Node Representation

- Desired partition properties:
  - Pages with similar adjacency lists are grouped together, as much as possible.
  - All the pages assigned to a given element of a partition belong to the same domain.
  - Among pages belonging to the same host, those with lexicographically similar URLs are more likely to be grouped together.
Reference Encoding

- It is a graph compression technique.
- We can compress the adjacency list of \( y \) by representing it in terms of the adjacency list of \( x \).
- For each page \( x \) in a graph \( G \), we decide whether the adjacency list for \( x \) is represented as is or in terms of a reference page, and in that case, the page that will act as reference.
- An affinity graph \( G_{aff} \) can be used to encode the intranode and superedge graphs.
Iterative Partition Refinement

- We begin with an initial coarse-grained partition $P_0 = \{N_{01}, N_{02}, \ldots, N_{0n}\}$.
- This partition is *refined* during successive iterations, generating a sequence of partitions $P_1, P_2, \ldots, P_f$.
- $P_0$ groups pages based on their domain.
- During every iteration, one of the elements of the existing partition is further broken into smaller pieces.
Iterative Partition Refinement

**URL Split:**
- Partitions the pages in $N_{ij}$ based on their URL patterns.
- Pages that share the same URL prefix are grouped together.
- Every application of URL split on a partition uses a URL prefix, one level/directory longer than the prefix used to generate that partition.
Iterative Partition Refinement

- **Clustered Split:**
  - Partitions the pages in $N_{ij}$ by using a clustering algorithm (e.g., k-means), to identify groups of pages with similar adjacency lists.

![Diagram showing the clustered split of pages]

<table>
<thead>
<tr>
<th></th>
<th>$N_{i1}$</th>
<th>$N_{i5}$</th>
<th>$N_{i9}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{adj}(P_1)$</td>
<td>1 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$\text{adj}(P_2)$</td>
<td>0 1 1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Physical Organization

- The supernode graph is encoded using standard adjacency lists.
- A simple Huffman-based compression scheme (based on supernode in-degree)
- Intranode and superedge graphs are encoded using the reference encoding scheme.
- Supernodes are numbered from 1 to n.
- All pages belonging to same supernode are numbered and placed consecutively, in lexicographic order of URLs.
Physical Organization

### Domain Index

<table>
<thead>
<tr>
<th>Domain</th>
<th>Page-ids</th>
</tr>
</thead>
<tbody>
<tr>
<td>stanford.edu</td>
<td>1, 4</td>
</tr>
<tr>
<td>berkeley.edu</td>
<td>2, 3, 5</td>
</tr>
</tbody>
</table>

### Page-id Index

<table>
<thead>
<tr>
<th>Page-id</th>
<th>Page-ids</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1, 987</td>
</tr>
<tr>
<td>2</td>
<td>[988, 15001]</td>
</tr>
<tr>
<td>3</td>
<td>[15005, ...]</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

### Supernode Graph (Huffman encoded)

- Node 1
- Node 2
- Node 3
- Node n
- Node n-1

### Index Files

- Reference-encoded IntraNode2
- Reference-encoded SEdgePos_{n-1,n}
- Reference-encoded IntraNode_n
Experimental Results

- **Source data**: about 120 million pages (approximately 900 GB of uncompressed HTML text) from the Stanford WebBase repository, using 5 different-sized data sets.

- The S-Node representation was compared to the following Web graph representation schemes:
  - **Connectivity Server - Link3 scheme**
  - **Huffman-encoded representation** (Huffman codes are assigned to each page based on in-degree).
  - **Relational database**. (using the PostgreSQL object relational database to store the adjacency lists as rows of a database table).
  - **Uncompressed files**.
Experimental Results

**Scalability Experiments:**

- From 50 million to 75 million pages (50% increase) → 11% increase in supernodes, and 15% increase in superedges.
- From 5 million to 100 million pages (20-fold increase) → almost a 3-fold increase in supernodes and superedges.
Experimental Results

- **Compression Experiments:**

<table>
<thead>
<tr>
<th>Representation scheme</th>
<th>Number of bits/edge</th>
<th>Max. repository size using 8GB</th>
<th>Seq. access (ns/edge)</th>
<th>Random access (ns/edge)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$W_g$</td>
<td>$W_g^T$</td>
<td>$W_g$</td>
<td>$W_g^T$</td>
</tr>
<tr>
<td>Plain Huffman</td>
<td>15.2</td>
<td>15.4</td>
<td>323 million</td>
<td>318 million</td>
</tr>
<tr>
<td>Connectivity Server</td>
<td>5.8</td>
<td>5.92</td>
<td>845 million</td>
<td>829 million</td>
</tr>
<tr>
<td>(Link3)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-Node</td>
<td>5.0</td>
<td>5.63</td>
<td>968 million</td>
<td>872 million</td>
</tr>
</tbody>
</table>
Experimental Results

**Complex Queries:** for example:

<table>
<thead>
<tr>
<th>No.</th>
<th>Description</th>
<th>Main graph operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Generate a list of universities that Stanford researchers working on <em>Mobile networking</em> refer to and collaborate with. (Analysis 1 in Section 1).</td>
<td>Subset of the out-neighborhood of a set of pages</td>
</tr>
<tr>
<td>2</td>
<td>Compute the relative popularity of three different comic strips among students at Stanford University. (Analysis 2 in Section 1).</td>
<td>Count number of links between 3 different pairs of sets of pages</td>
</tr>
<tr>
<td>3</td>
<td>Compute the <em>Kleinberg base set</em> [10] for <em>S</em>, where <em>S</em> is the set of top 100 pages (in order of PageRank) that contain the phrase ‘Internet censorship’.</td>
<td>Union of out-neighborhood and in-neighborhood of a set of pages</td>
</tr>
<tr>
<td>4</td>
<td>Identify the 10 most popular pages on <em>Quantum cryptography</em> at each of the following four universities - Stanford, MIT, Caltech, and Berkeley. Popularity of a page is measured by the number of incoming links from pages located outside the domain to which the page belongs.</td>
<td>In-neighborhood for four different sets of pages</td>
</tr>
<tr>
<td>5</td>
<td>Suppose <em>S</em> is the set of pages in the repository that contain the phrase <em>Computer music synthesis</em>. Rank each page in <em>S</em> by the number of incoming links from other pages in <em>S</em>. Output the top ranked 10 .edu pages in <em>S</em>.</td>
<td>Computation of graph induced by a set of pages</td>
</tr>
<tr>
<td>6</td>
<td>Suppose <em>S1</em> is the set of Stanford pages (i.e., pages in stanford.edu) that contain the phrase <em>Optical Interferometry</em> and <em>S2</em> is the set of Berkeley pages (i.e., pages in berkeley.edu) that contain the same phrase. Let <em>R</em> be the set of pages (not in stanford.edu and berkeley.edu) that are pointed to by at least one page in <em>S1</em> and one page in <em>S2</em>. Rank each page in <em>R</em> by the number of incoming links from <em>S1</em> and <em>S2</em> and output <em>R</em> in descending order by rank.</td>
<td>Intersection of out-neighborhoods of two different sets of pages</td>
</tr>
</tbody>
</table>
Experimental Results

<table>
<thead>
<tr>
<th>Query</th>
<th>Navigation time reduction by using S-Node</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>73.5%</td>
</tr>
<tr>
<td>2</td>
<td>76.9%</td>
</tr>
<tr>
<td>3</td>
<td>77.7%</td>
</tr>
<tr>
<td>4</td>
<td>82.2%</td>
</tr>
<tr>
<td>5</td>
<td>79.2%</td>
</tr>
<tr>
<td>6</td>
<td>89.2%</td>
</tr>
</tbody>
</table>
Summary

- The paper addresses the problem of efficiently representing massive Web graphs.
- It proposes a novel two-level representation of Web graphs, called an S-Node representation.
- It is based on partitioning the set of pages in the repository.
- S-Node representation can provide impressive compression characteristics (just over 5 bits per edge to represent Web graphs).
- It can also achieve a significant reduction in query execution time (10 to 15 times faster than other schemes for representing Web graphs).