CMSC 828E: Graph Compression

Amol Deshpande

University of Maryland, College Park

October 20, 2010

Overview

- Why? Graphs are very large and operations often done in-memory
- Overview of techniques
    - Graph layout: “minimum-linear-arrangement” problem
        - Linearize the nodes so that average “stretch” of an edge is minimized
        - Minimum-bandwidth problem asks for minimizing max (instead of avg)
    - Identify and replace dense structures
        - E.g., if there is a clique, replace with a special node and edges to the members
    - Neighborhood similarities
        - If two nodes have identical neighborhoods, can store a pointer from one to the other
        - Many works appear to have done this independently

Issues

- Incremental maintenance ?
- Labels vs no labels
  - Having edge labels would make some techniques inapplicable
- Queries ?
  - Must be able to process the graph efficiently – preferably without decompressing the whole graph
  - Any compression must come at the expense of query answering
  - So: Identify queries/tasks that you need to do
- How dense are the graphs ?
  - Denser graphs easier to compress, but most practical graphs are sparse

Clique Partitions and Graph Compression

- Feder and Motwani; STOC 1991
- Based on partitioning the edges into complete bipartite graphs
- Commonly denoted $K_{n,m}$: n nodes completely connected to m nodes on the other side
- Can replace the mn edges with $m + n$ edges (by adding a special node)
  - They replace the mn edges with a tree over those nodes, but their goal is different
- Problem NP-Hard; but can be approximated for "dense graphs"
  - Social networks or Web graphs are actually quite sparse
- Can solve many standard graph algorithms efficiently (matchings, connectivity etc)

Clique Partitions and Graph Compression

- Illustrative figure (from Buehrer et al.)

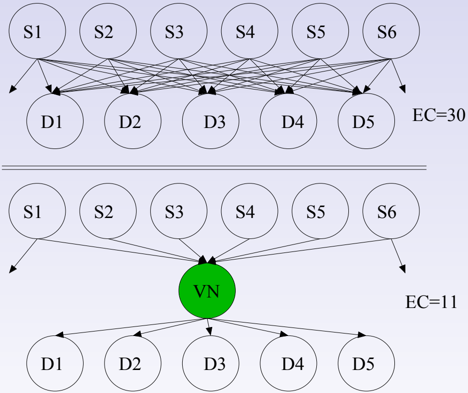

Figure 1: A bipartite graph compressed to a virtual node, removing 19/30 shared edges.

Compact Representations of Separable Graphs

- Blandford et al.; SODA 2003
- Based on existence of small separators
  - Several classes of graphs are known to have $O(n^c)$ size separators, $c < 1$
- Basic idea: Identify separator, split the graph, recurse
  - At each step, relable the nodes
  - Most edges will be between nodes that are close to each other in the numbering
- Some connections to the basic idea behind INDSEP

Buehrer et al.; WSDM 2008

- A Scalable Pattern Mining Approach to Web Graph Compression with Communities
- Based on frequent itemset mining
    - Identify groups of nodes that share the same outgoing links
    - Compress by replacing with a virtual node that points to the those targets
- Quite similar to Feder and Motwani's work
    - Paper not cited
- Discuss how PageRank can be computed without decompressing

The LINK Database

- Randall et al; Data Compression Conference 2001
- (1) Most hyperlinks are intra-source
- (2) Many nodes (pages) share outgoing edges (neighborhoods)
- Can achieve < 6 bits per link
- Can still compute strongly connected components or run HITS efficiently

The Link Database

- Link1: 32 bits per link, can only store 100 million webpage in 8GB Memory
  - Not enough
  - Disk-based methods not appropriate – too slow
- Link2: Compress each adjacency list locally
  - Most links intra-host; can compress significantly
  - Called "gap-coding": delta compression
- Link3: Compress an adjacency list using
  - A pointer to a previous adjacency list + additions - deletions
  - High decompression times: must limit to small "chains"

WebGraph Framework

- Boldi and Vigna; DCC 2004, WWW 2004
- Exploit:
  - Locality: links are mostly intra-host
  - Similarity: pages close to each other have common neighbhorhoods
- Could compress 118M nodes, 1G links in 3.08 bits per link
- They also developed a new coding scheme (in the DCC paper)
  - Suitable for compressing integers with a power law distribution

On Compressing Social Networks

- Chierichetti et al.; KDD 2009
- Follows on from Boldi and Vigna's work on compressing Web graphs
- Key idea: exploiting commonalities between neighborhoods + lexicographic ordering
- The latter doesn't work for social networks – no natural order
  - Must come up with an appropriate ordering
  - Problems NP-Hard
  - Use an approach based on Shingles (*signatures*)
    - Aside: Shingles are useful as signatures of sets in many other domains

Representing Web Graphs

- Raghavan and Garcia-Molina; ICDE 2003
- Focus on somewhat more expressive queries, over small subgraphs

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

Table 2. Some of the queries used in the experiments

Representing Web Graphs

- Hierarchical index structure

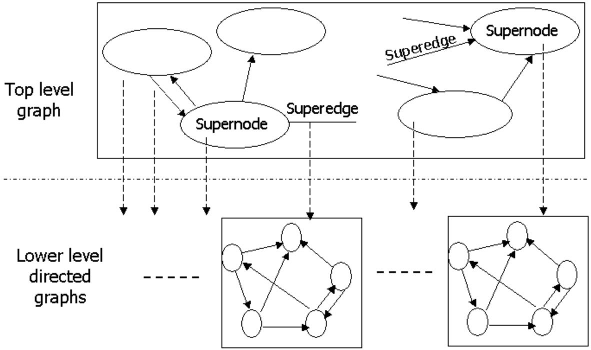

Figure 2. S-Node representation of a Web graph

Representing Web Graphs

- Key question: How to do the partitioning ?
  - Would prefer to have queries be local, and also few inter partition edges
  - Several heuristics developed based on commonalities in adjacency lists, domains etc.
- Need to maintain a mapping between original node labels and new node ids

Graph Summarization

- Navlakha et al.; SIGMOD 2008
- Similar to the previous work
- Compress a graph as:
  - A graph over supernodes
  - A "correction" graph
- Present both exact and approximate algorithms
- No discussion of querying

Neighbor Query Friendly Compression of Social Networks

- Maserrat, Pei; KDD 2010
- Store the Eulerian path directly if one exists
- If not, use a generalization to Eulerian path
- No edges need to be stored explicitly
- Can answer both in- and out-neighbor queries efficiently

Summary of Work

- Adler and Mitzenmacher: based on finding nodes with similar neighborhoods
- Randall: lexicographic ordering of URLs
- Boldi and Vigna: exploit lexicographic ordering + similar neighborhoods
- Raghavan and Garcia-Molina: decomposed Web graph into hierarchical structure
- Buehrer and Chelapilla: frequent item-set mining to mine complete bipartite graphs