Web Graph Compression by Edge Elimination

A. Mahdian H. Khalili E. Nourbakhsh M. Ghodsi
{mahdian, khalili, nourbakhsh, ghodsi} @ce.sharif.edu

Department of Computer Engineering,
Sharif University of Technology, Tehran, Iran.

In our work we focus on the problem of compressing the web graph by means of eliminating some of the edges in its link structure. Our algorithm divides the task, so that it can be executed on parallel processors. It can be summarized in the following three steps:

1. Finding the communities with desirable density: Among different declarations of density, in this work we use the ratio of average out-degrees of nodes to the total number of nodes in a detected community as a measure of its density.
2. Re-indexing each node in detected communities: Here we try to re-index every node of the community so that at the end, for almost every node in the community, the numerical dispersion of indices of the nodes participating in its adjacency list is much less than before. We use an initially empty list for keeping the final order of the nodes, which we refer to as the general list, and add the adjacency list of each node to this list one at a time. In order to get the best result, nodes are added based on their out-degree in increasing order. Considering the process of adding adjacency list of Node(i) to the general list, one of the two situations can occur:
   - There is no common edge between the general list and the adjacency list of Node(i): we simply add the adjacency list of Node(i) to the end of the general list.
   - There is a common list of edges between the general list and the adjacency list of Node(i): we move the common list, with respect to ordering of its nodes in the general list, to the end of general list and add the remaining nodes right after the common edges at the end of the general list. By this procedure, nodes which are ordered based on the adjacency list of previously added nodes will not be affected.
3. Eliminating some of the edges: In this step each node is assigned to a dynamic length group based on its new numerical index value. With the occurrence of several edges that point to nodes of the same group in the adjacency list of a particular node, all of these edges will be replaced by one edge that points to the group indicator of that group. In order to keep track of those eliminated edges, we use an auxiliary data structure so that we can decompress the web graph later.

Ran on a test bed of generated web graphs, our algorithm improved the compression ratio of both huffman-coding schemes and Adler and Mitzenmacher's *Find Reference* algorithm tangibly. In the *Find Reference* case, improvement was up to 90%.