

k²-Trees for Compact Web Graph Representation^{*}

Nieves R. Brisaboa¹, Susana Ladra¹, and Gonzalo Navarro²

¹ Database Lab., Univ. of A Coruña, Spain

`{brisaboa,sladra}@udc.es`

² Dept. of Computer Science, Univ. of Chile, Chile

`gnavarro@dcc.uchile.cl`

Abstract. This paper presents a Web graph representation based on a compact tree structure that takes advantage of large empty areas of the adjacency matrix of the graph. Our results show that our method is competitive with the best alternatives in the literature, offering a very good compression ratio (3.3–5.3 bits per link) while permitting fast navigation on the graph to obtain direct as well as reverse neighbors (2–15 microseconds per neighbor delivered). Moreover, it allows for extended functionality not usually considered in compressed graph representations.

1 Introduction

The World Wide Web structure can be regarded as a directed graph at several levels, the finest grained one being pages that point to pages. Many algorithms of interest to obtain information from the Web structure are essentially basic algorithms applied over the Web graph [11, 16].

Running typical algorithms on those huge Web graphs is always a problem. Even the simplest external memory graph algorithms, such as graph traversals, are usually non disk-friendly [24]. This has pushed several authors to consider *compressed graph representations*, which aim to offer memory-efficient graph representations that still allow fast navigation without decompressing. The aim of this research is to allow classical graph algorithms to be run in main memory over much larger graphs than those affordable with a plain representation.

The most famous representative of this trend is surely Boldi and Vigna’s *WebGraph Framework* [6]. The WebGraph compression method is indeed the most successful member of a family of approaches to compress Web graphs based on their statistical properties [1, 5, 7, 20, 21, 23]. It allows fast extraction of the neighbors of a page while spending just a few bits per link (about 2 to 6, depending on the desired navigation performance). Their representation explicitly exploits Web graph properties such as: (1) the power-law distribution of indegrees and outdegrees, (2) the locality of reference, (3) the “copy property” (the set of neighbors of a page is usually very similar to that of some other page).

More recently, Claude and Navarro [10] showed that most of those properties are elegantly captured by applying Re-Pair compression [17] on the adjacency

^{*} Funded in part (for the Spanish group) by MEC grant (TIN2006-15071-C03-03); and for the third author by Fondecyt Grant 1-080019, Chile.

lists, and that *reverse navigation* (finding the pages that point to a given page) could be achieved by representing the output of Re-Pair using some more sophisticated data structures [9]. Reverse navigation is useful to compute several relevance ranking on pages, such as HITS, PageRank, and others. Their technique offers better space/time tradeoffs than WebGraph, that is, they offer faster navigation than WebGraph when both structures use the same space.

Asano et al. [2] achieve even less than 2 bits per link by explicitly exploiting regularity properties of the adjacency matrix of the Web graphs, but their navigation time is substantially higher, as they need to uncompress full domains in order to find the neighbors of a single page.

In this paper we also aim at exploiting the properties of the adjacency matrix, yet with a general technique to take advantage of clustering rather than a technique tailored to particular Web graphs. We introduce a compact tree representation of the matrix that not only is very efficient to represent large empty areas of the matrix, but at the same time allows efficient forward and backward navigation of the graph. An elegant feature of our solution is that it is symmetric, both navigations are carried out by similar means and achieve similar times. In addition, our proposal allows some interesting operations that are not usually present in alternative structures.

2 Our Proposal

The adjacency matrix of a Web graph of n pages is a square matrix $\{a_{ij}\}$ of $n \times n$, where each row and each column represents a Web page. Cell a_{ij} is 1 if there is a hyperlink in page i towards page j , and 0 otherwise. Page identifiers are integers, which correspond to their position in an array of alphabetically sorted URLs. This puts together the pages of the same domains, and thus locality of reference translates into closeness of page identifiers. As on average there are about 15 links per Web page, this matrix is extremely sparse. Due to locality of reference, many 1s are placed around the main diagonal (that is, page i has many pointers to pages nearby i). Due to the copy property, similar rows are common in the matrix. Finally, due to skewness of distribution, some rows and columns have many 1s, but most have very few.

We propose a compact representation of the adjacency matrix that exploits its sparseness and clustering properties. The representation is designed to compress large matrix areas with all 0s into very few bits.

We represent the adjacency matrix by a k^2 -ary tree, which we call k^2 -tree, of height $h = \lceil \log_k n \rceil$. Each node contains a single bit of data: 1 for the internal nodes and 0 for the leaves, except for the last level, where all are leaves and represent bit values of the matrix. The first level (numbered 0) corresponds to the root; its k^2 children are represented at level 1. Each child is a node and therefore it has a value 0 or 1. All internal nodes (i.e., with value 1) have exactly k^2 children, whereas leaves (with value 0 or at the last tree level) have no children.

Assume for simplicity that n is a power of k ; we will soon remove this assumption. Conceptually, we start dividing the adjacency matrix following a MX-Quadtree strategy [22, Section 1.4.2.1] into k^2 submatrices of the same size, that