A STUDY ON WEBGRAPHS

by

Soumen Sengupta

Submitted in partial fulfillment of the
requirements for the degree of
Master of Applied Computer Science

at

Dalhousie University
Halifax, Nova Scotia
December 2009

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled "A STUDY ON WEBGRAPHS" by Soumen Sengupta in partial fulfillment of the requirements for the degree of Master of Applied Computer Science.

Dated: December 7, 2009

Supervisor: _____
Prof Jason Brown

Reader: _____
Prof Jeannette Janssen

DALHOUSIE UNIVERSITY

DATE: December 7, 2009

AUTHOR: Soumen Sengupta

TITLE: A STUDY ON WEBGRAPHS

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: M.A.C.Sc. CONVOCATION: May YEAR: 2010

Permission is herewith granted to Dalhousie University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Author

Table of Contents

Abstract

The World Wide Web is an evolving network, comprising of webpages and interconnections between the webpages. This huge linked structure of webpages along with its connections, are better known as webgraphs. Webgraphs are known to break down into several components namely the Giant Strongly Connected Component, IN, OUT, TENDRILS and TUBES and the DISCONNECTED COMPONENTS. The whole webgraph as well as the components exhibit interesting properties of which the scale free and self similar properties were studied within the scope of this project.

The link structure of a collection of websites obtained from a crawl of .GOV sites done by University of Glasgow was studied both at the macroscopic level and the microscopic level. The webgraph contained a total of 1,247,753 documents and roughly over 11 million links. Some of the salient features of the webgraph, namely the power law distribution for in degrees, out degrees, and the distribution of the size of the strongly connected components were examined. Studies of the inner structure of the components of the webgraph revealed a rather fragmented and loosely connected structure unlike the whole webgraph which contained a giant core. However, the scale free characteristics were confirmed at the component level by the the degree distributions.

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Jason Brown for his guidance and unwavering patience. It would not have been possible for me to finish this project without his supervision. He inspired me and assisted me every step of the way. I am grateful to both my parents and my brother Sunny for their assistance in every possible way. I would also like to thank Mr Ballagopal Pillai for his kind cooperation with the jobs in the clusters. Finally, I would also like to thank my department for all the help I got.

Chapter 1

Introduction

The World Wide Web was initially invented in 1989 solely for the purpose of sharing scientific documents using a common platform. It now consists of several billions of documents that are linked to one another across the universe and allow users to share information. These documents are known as *web pages* and the pages are connected to each other using *hyperlinks*. The web is dynamic in nature and the hyperlinked structure of the web keeps on changing constantly with the addition of new pages and deletion of old and expired pages. The content of the web pages also keeps on changing. At present, the number of pages has been estimated somewhere between 15 billion and 30 billion with millions of new pages being added everyday. The new pages are linked to old pages in a particular way depending on several factors that we will look at later. This linkbase creates an intricate graph structure that has been of particular interest in the recent times. A directed graph comprising of web pages that are connected to each other by hyperlinks is also known as a *webgraph*. Interestingly, despite the apparent haphazard nature in which the connections are being created, the webgraph exhibits some really interesting characteristics.

The study of webgraphs has been of particular interest to scientists because of several reasons. Developments of efficient search engines and web algorithms for searching has been benefited by a proper understanding of the structure of the web graph. Scientists have tried to find out why web authors tend to favor certain webpages over others. Certain pages are being linked to more than that of the rest. The number of documents created by scientists are constantly rising and the scientists are no longer limited to a small network. Their ideas are shared across the web to a much bigger community. The web is also playing a huge role to the business community and helping small businesses collaborate with larger business groups. With the rise of E-commerce, businessmen are turning to the web for a greater reach and the businesses are most likely to flourish with a better understanding of the structure of the

web. The presence of various community structures in the webgraph has also been of tremendous interest. However, it is very difficult to study a huge and complex network that is constantly changing. Thus there arises the need for efficient crawling strategies for bigger chunks of the web. Better crawling strategies can be developed and browsing of this huge repository of information can also be improved greatly by understanding the finer details of the structures. Compression algorithms have been used to study the web graph more efficiently. The study of webgraphs has revealed several interesting macroscopic and microscopic structures and properties that will be looked at in detail within the scope of this project.

The size of the whole world wide web was predicted to be somewhere between 15 and 30 billions and with millions of pages being added every month its size is increasing exponentially. Most crawlers cannot reach the whole web and it is difficult to study the linked structure of such a complex network. Therefore they have been studied at a smaller scale by allowing the crawler to crawl upto a certain number of pages. Smaller webgraphs tends to exhibit similar properties to larger ones and are easier to study than the larger ones. Besides, the web has been known to exhibit self similar properties. We have studied the link structure of a collection of websites obtained from a crawl of University of Glasgow done in 2002. The web graph studied, contained a total of eleven million (11,067,049) links and (1,213,307) pages. It was studied both at the microscopic and macroscopic level. We have identified some of the components and studied them in detail with respect to their link structure and other inherent properties. Power law distribution for in-degrees and out-degrees of vertices and the distribution of the size of the strongly connected components were examined both for the total web graph as well as its components. The finer structure of the components were studied and we have also examined if bow ties characteristic of a webgraphs actually exist within the components IN and OUT of the webgraph collection. The results are reported in later sections.

The next sections will throw further light on the various structural and topological properties of the webgraph as well as our findings on the webcrawl. In section 2 the various graph theoretic definitions, data structures and algorithms used in the project are described. Section 3 describes the various structural and topological properties of the webgraph. Section 4 gives a brief overview of different models of the webgraph

available while the results of the experiments on the webcrawl are described in Section 5.

Chapter 2

Graph Theoretic Definitions, Data Structures and Algorithms

2.1 Definitions

A ***graph*** $G = (V, E)$ is a collection of vertices V together with a set E which consists of either ordered or unordered pairs of edges of vertices V. An ***edge*** (u, v) denotes a connection from node u to node v. The total number of vertices in the graph is usually denoted $N = |V|$ while $M = |E|$ represents the total number of edges in the graph.

The ***degree*** k of a vertex u in a directed graph is the number of edges incident with it. The ***in-degree*** of any vertex v in a digraph is the number of edges, (u, v) present in the digraph. The ***out-degree*** of a vertex v is the number of edges (v, u) present. The degree of the vertex 2 in Figure 2.1 is 6 with in-degree 2 and out-degree 4 as shown in Figure 2.1.

A ***walk***: A walk can be defined as an alternating sequence of vertices and edges starting and ending with vertices, such that each edge is directed from the vertex preceding it to the one following it. The length of a walk is its number of edges. Thus $\{\mathbf{8}, \mathbf{81}, \mathbf{1}, \mathbf{12}, \mathbf{2}, \mathbf{23}, \mathbf{3}, \mathbf{36}, \mathbf{6}\}$ is a walk from vertex 8 to vertex 6 with a length of 4 in Figure 2.1. We often omit the edges in listing a walk.

A ***u-v path*** of length k from node u to v can be defined as the set k of edges $\{uv_1, v_1v_2, v_2v_3, v_3v_4, \ldots v_kv\}$ that form a walk from u to v such that no vertex is repeated more than once. In the directed case, a path from u to v does not guarantee a path from v to u. For example in Figure 2.1, $\{81, 13, 36, 65\}$ would form a path from 8 to 5 and it can be seen in this case that there exists no directed path from 5 to 8 . The ***shortest path*** from a vertex u to v is a set of edges that form a path between u and v of shortest length.

A ***cycle*** can be defined as a closed walk where no vertex is repeated more than

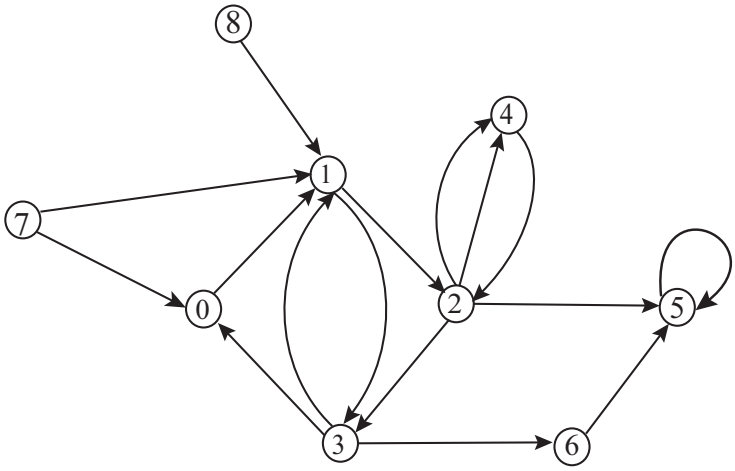


Figure 2.1: A Directed Graph with eight vertices

once Thus in Figure 2.1, $\{0, 1, 2, 3, 0\}$, $\{2, 4, 2\}$ and $\{1, 3, 1\}$ are cycles while $\{5, 5\}$, a self loop on the vertex 5 represents the smallest cycle in the directed graph.

The ***diameter*** of a graph G can be defined as the maximum of the shortest $u - v$ path lengths over all ordered pairs (u, v) in the digraph.

A ***subgraph***, $S = (T, U)$ is a digraph that consists of a set of vertices T and edges U such that T is a subset of V and U a subset of E.

A ***bipartite subgraph*** $B = (H, A)$ is a digraph that consists of two sets of vertices U and U' such that the edges run from the vertices in set U to that of the vertices in set U'.

A ***tree*** T is an acyclic graph $G = (V, E)$ where all the vertices in the graph are connected. A tree is characterized by a root vertex and each of the vertices is connected to their ancestors or descendants through a single edge. If there is a path from a vertex v to another vertex u where v and u are vertices in the tree T then v is the **ancestor** of u while u is the **descendant** of v. A root node in a tree cannot have any ancestors. If a vertex v is connected to another vertex w by a forward edge then w is a ***proper descendant*** of v and w is a ***proper ancestor*** of v.

A ***strongly connected component*** (**scc**) is a maximal subgraph X of graph G such that every pair of vertices u and v in X are equivalent. There is a path from u to v and vice versa. The graph $G = (V, E)$ can be partitioned into its strongly connected components C_1, C_2, C_3 and so on, with C_i on vertexset. The digraph in Figure 2.1 can be decomposed into strongly connected components, $V_1 = \{0, 1, 2, 3, 4\}$, $V_2 = \{5\}$,

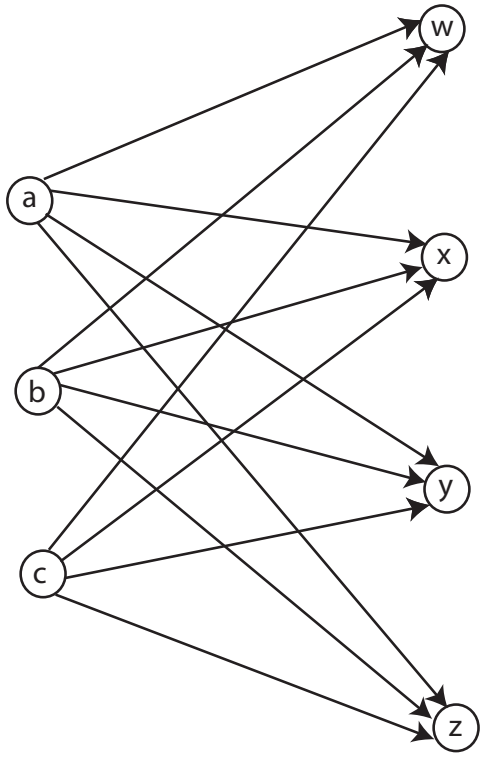


Figure 2.2: A Bipartite Clique

$V_3 = \{6\}$, $V_4 = \{7\}$ and $V_5 = \{8\}$.

A ***weakly connected component*** is a strongly connected component of the digraph formed by adding in arc (x, y) whenever $(x, y) \notin E$ and $(y, x) \in E$. Equivalently, the weakly connected component is a connected component in the underlying undirected graph G' of G formed by replacing each edge (x, y) by $\{x, y\}$. Thus the direction of the arcs is not taken into account.

In an undirected graph $G = (V, E)$ a ***clique*** can be defined as a subgraph $\mathbf{G'} = (\mathbf{V'}, \mathbf{E'})$ such that for each pair of vertices in the subgraph there is an edge connecting them that is, it a complete subgraph of G. The size of the clique is the number of vertices in the subgraph.

A ***bipartite clique*** is a subgraph $K_{i,j} = (V', E')$ where V comprises of two sets of vertices V_1 containing i vertices and V_2 containing j vertices and E represents the set of all possible edges from V_1 to V_2. Each of the vertices in the set V_1 is connected to all the vertices in V_2. Figure 2.2 is an example of a bipartite clique $K_{3,4}$ where the set $V = \{a, b, c\}$ connects to each of the vertices in $V_2 = \{w, x, y, z\}$. The first set V_1 is called the *authorities* while V_2 represents the *hubs*. Please refer to [15] for further

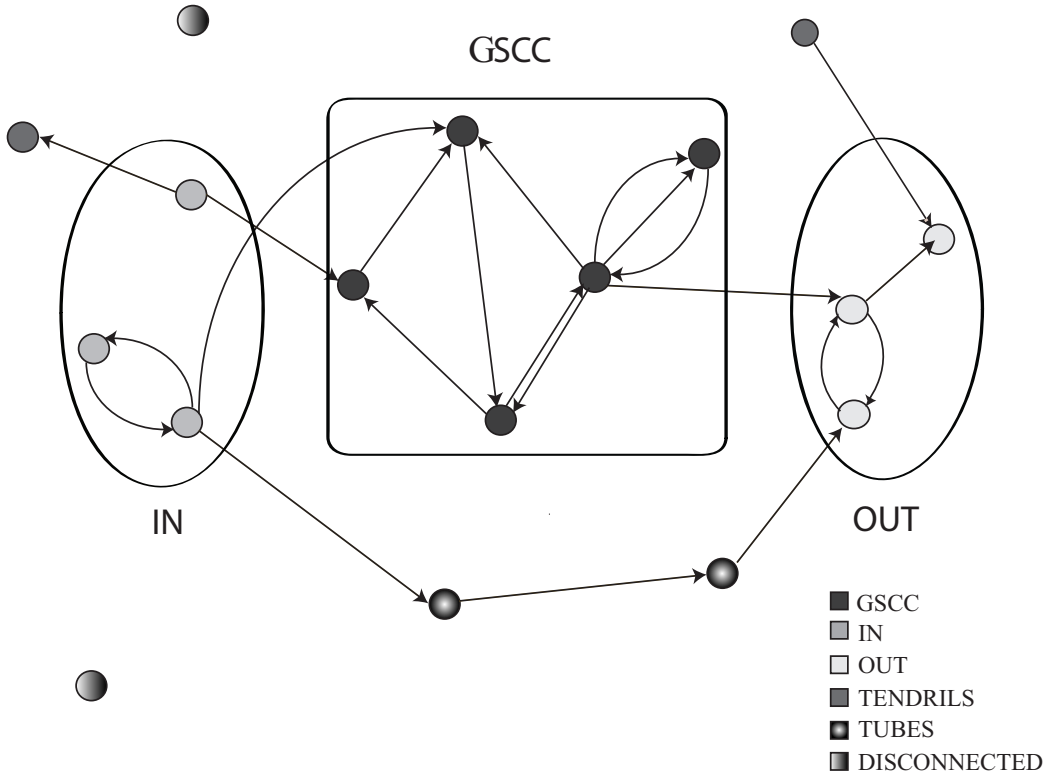


Figure 2.3: A sample webgraph

details on the graph theoretic terms.

The ***webgraph*** $W = (P, H)$ is a digraph where P is the set of web pages and H the set of hyperlinks that connect the webpages in P. W, like every digraph, is known to break into four disjoint pieces of which the $GSCC = (P_{GSCC}, H_{GSCC})$ forms the core of W flanked by two subgraphs $IN = (P_{IN}, H_{IN})$ and $OUT = (P_{OUT}, H_{OUT})$, the subgraphs of vertices that can reach the GSCC and can be reached from the GSCC respectively, and the subset TENDRILS containing pages that can neither reach nor be reached by GSCC. The digraph $B = (P_B, H_B)$, comprising of IN, $GSCC$ and OUT where $P_B = P_{IN} \bigcup P_{GSCC} \bigcup P_{OUT}$ and H_B the set of hyperlinks that provide the links within the individual components as well as the hyperlinks from IN to GSCC and that from GSCC to OUT is referred to as the **bow tie B**. In the sample webgraph is given in Figure 2.3, IN and OUT are marked by two blue ellipsoid regions and P_{IN} and P_{OUT} each have 3 pages in them them. The GSCC is shown in the square region and the collection of pages in IN, OUT and GSCC taken together would represent P_B. A detailed description of the components of the webgraph and its structure at

both the microscopic and the macroscopic level is given in Section 3.3.

2.2 Data Structures for Graphs

There are two main data structures that are usually used to represent graphs, adjacency lists and adjacency matrices. It is important to know which of these representations would be better suited for a graph of particular size and density. The description of adjacency lists and adjacency matrices follows. ***Adjacency Lists***: An adjacency list is an array of linked lists that represents the link structure present in a graph. In order to represent a graph G with V vertices and E edges, an array $A[\,]$ of size $|V|$ is created and each element of the array $A[u]$ is a linked list that contains pointers to all vertices $v \in V$, for which there is an edge from vertex u to v. For undirected graphs an edge between vertices v and u occurs in both the linked lists of u and v. An edge $v \rightarrow u$ in a directed graph is represented by a pointer in $A[v]$ to vertex u. Figure 2.4 shows the adjacency list of the directed graph shown in Figure 2.1. An edge from vertex 2 to 1 is represented by a pointer in the $A[2]$ to vertex 1. If a vertex has no outgoing links to any other node the linked list for the corresponding array remains empty (NULL). In order to represent multiple edges from vertex v to u, pointers are created to the vertex u more than once. This is seen in $A[2]$ where there are two references to the node 4, representing two edges $2 \rightarrow 4$. If a new edge needs to be created from $2 \rightarrow 0$ in Figure 2.1, the linked list has to be traversed until the end of the list for vertex 2 before a pointer to 0 can be added to the tail of the linked list. Self loops from vertex v can be represented by a pointer to itself in $A[v]$. This is illustrated in Figure 2.4 where a self loop on vertex 5 is represented by a pointer to 5.

Linked lists save a lot of memory and are usually ideal for sparse graphs. The amount of memory required for a linked list is of the order of $O(M + N)$ since the edges are represented by the linked lists and the sum of all the lengths of the linked list is equal to $|E|$ for a digraph and $|2E|$ for an undirected graph. This makes linked lists much more suitable if a traversal of the whole graph is required. Insertions or deletions of nodes in adjacency lists require traversal of the corresponding linked list. Visiting all the nodes in a linked list by starting at the root node and following the pointers from one node to the next until the end is reached constitutes a traversal.

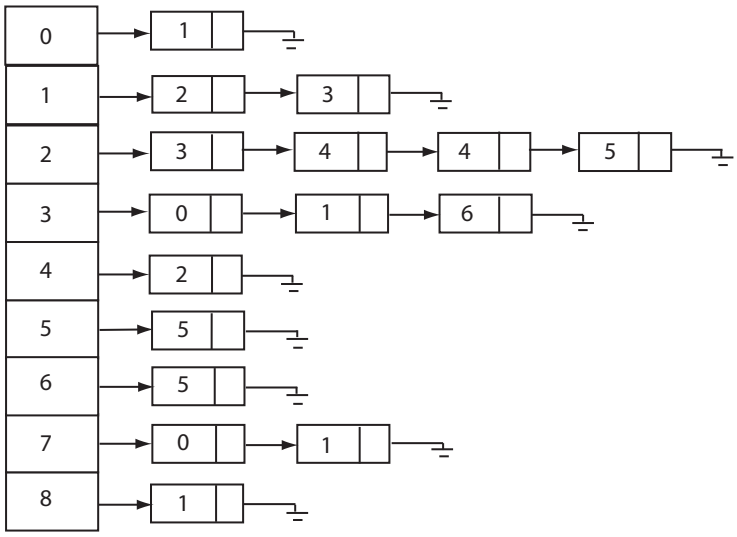


Figure 2.4: An adjacency list representation of the directed graph in Figure 2.1

The traversal of the linked list takes roughly $O(d)$ time where d is the outdegree of the vertex. Computing the degree of a vertex in a linked list requires the same amount of time. The main drawback of the adjacency list lies in its complicated structure and representation. Simplicity has to be sacrificed in order to achieve faster traversals and more efficient space usage for the whole graph. Edge connection lookups are not as simple as in adjacency matrices. A traversal of the corresponding linked list is required before the presence of a link can be ascertained or if a link needs to be deleted.

Adjacency Matrices: An adjacency matrix is a $N \times N$ matrix, $A[\ ][\ ]$, used to represent the link structure in a graph. An edge between a vertex i and j is represented by a 1 in $A[i][j]$ while a 0 signifies the absence of any edges. For undirected graphs the adjacency matrix is symmetric with both $A[i][j]$ and $A[j][i]$ storing the same values. In the directed case an entry of 1 in $A[i][j]$ corresponds to a connection from $i \rightarrow j$ and in $A[j][i]$ represents the arc $j \rightarrow i$. The diagonal of the matrix is usually 0 for loopless graphs. An adjacency matrix representation of the graph in Figure 2.1 is given below. In the adjacency matrix shown, the entry $A[2][4] = 2$ represent two edges from vertex $2 \rightarrow 4$. The self loop on vertex 5 is shown by a 1 in $A[5][5]$.

$$\begin{vmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 2 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{vmatrix}.$$

Adjacency matrix representation of the graph in Figure 2.1

Adjacency matrices are ideal for dense graphs. The representation of graphs by a matrix makes it easier to look up edges. Creating an edge or deleting an edge is as easy as assigning the corresponding entry in the matrix to 1 or 0. In order to ascertain the degree of a vertex it requires $O(N)$ amount of time since all the entries of a row corresponding to the vertex in question have to be checked. However, the space required for a graph is of the order of $O(N^2)$ which takes up quite a lot of memory when the number of vertices is large. Traversals of the whole graph may require $O(N^2)$ time. Thus by comparison linked lists are much more efficient for most cases.

2.3 Some Graph Algorithms

Depth First Search

The Depth First Search (DFS) Algorithm (see for example [15]) proceeds by selecting a root vertex and then traverses the graph along the child nodes of the search tree until it reaches a node that has no descendants. At this point, it backtracks to the most recent node and operates in a similar manner, until all its adjacent nodes have been explored. For undirected graphs that are connected, the output of the algorithm is a depth first search tree. However, running the algorithm on a directed graph results in a depth first search rooted directed forest, comprising of depth first search rooted directed trees. The output of a depth first search algorithm varies

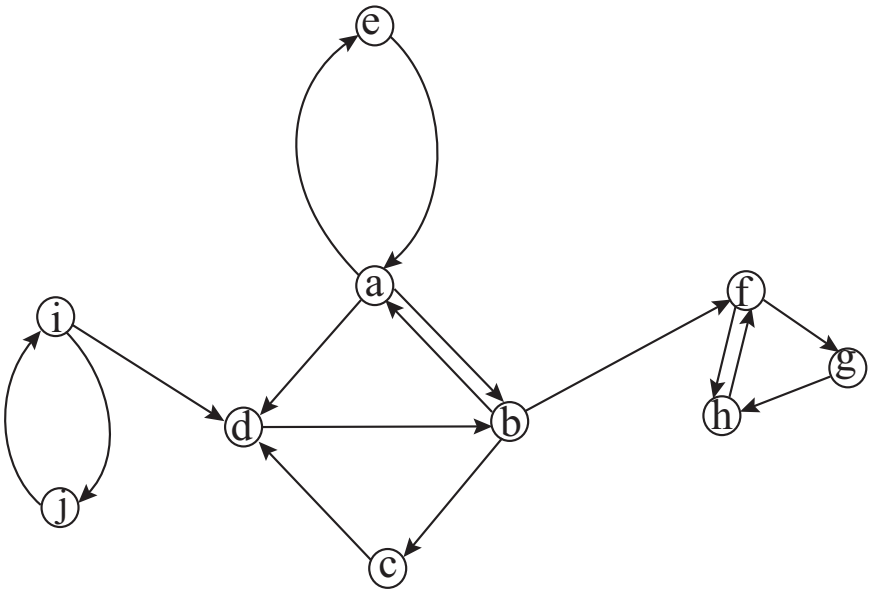


Figure 2.5: A sample graph to illustrate the various graph algorithms

depending on the root node chosen. Depth First algorithms are widely used to check reachability. Other uses of DFS are finding the components of a graph, finding cycles in graphs and so on.

In the DFS algorithm each of the vertices can have three states: undiscovered, discovered or explored. As the name suggests, undiscovered vertices are those that the algorithm has not encountered in its traversal yet. A vertex that has been traversed at least once but its child nodes have not been examined yet is in a discovered state. The final state of every vertex in the graph is explored where all the child nodes of the vertex have been explored. The Depth First Search algorithm has as its input a graph G whose nodes are marked as undiscovered. Initially a root node is selected and marked as discovered. On discovery of each vertex in the graph it is assigned a depth first number. The algorithm then traverses along the child node of the root which in turn explores its descendant. This process continues until a leaf (a vertex with degree 1) is found. If the algorithm finds a vertex that has already been discovered the edge is ignored since it has already been discovered and therefore included in the search tree. At each step, a descendant is found it is marked discovered and added to the predecessor subgraph. All the child nodes are recursively examined until a leaf node is found. The leaf node is marked as explored and since there are no other nodes adjacent to it the leaf is assigned a finish time. The algorithm then backtracks

to the recently visited node which is expanded before the control is returned to its predecessor. The pseudocode for the algorithm is given below and the algorithm is explained with an illustration of the depth first tree obtained after running DFS on the directed graph in Figure 2.5.

DFS(Graph G)
$\forall v \in V(G)$
Mark v as undiscovered;
$time \leftarrow 0$
$T[v] = NULL$;
$\forall v \in V(G)$
Traverse(v);
end.

Traverse(vertex v)
$\forall u \in adj(v)$
If u is not discovered
Mark u as discovered;
$time \leftarrow time + 1$
$depth[u] \leftarrow time$
$T[v] \leftarrow u$;
Traverse(u);
end if.
Mark u as explored;
$time \leftarrow time + 1$
$f[u] \leftarrow time$
end .

Suppose that DFS is run on the directed graph given in Figure 2.5 with the root vertex chosen to be i. First of all, the vertices in the graph are all marked as undiscovered and the time set to 1. Then since the starting vertex is i, it is marked discovered and given a depth first number of 1. Now, the vertices adjacent to i are d and j and neither of the vertices have been discovered as yet. Let us suppose that d

is chosen and id is added to the depth first tree. d is assigned a depth first number of 2 and i is set as the predecessor of d. Vertices adjacent to d are then traversed. Incidentally, d has only one adjacent vertex b and since it is still in an undiscovered state it is chosen and the edge db added to the depth first tree. Next, c is chosen and bc added to the tree. When the algorithm reaches the vertex c, it finds that c has only one adjacent vertex d but d has already been discovered. The edge connecting a vertex that has been discovered later in the depth first traversal to one that comes earlier in the depth first tree of a graph is also known as a *back edge*. Thus cd is a back edge as shown by a thin arc in figure 2.6 (a).

Since all the vertices adjacent to c has been discovered, c is marked explored and a finishing time of 5 assigned to it. At this point the control goes back to its immediate predecessor b and the unexplored edge ba is chosen. Vertex a is added to the depth first tree and its adjacent vertices traversed. At a, the algorithm finds two back edges ad and ab. Thus, e is traversed and since it has no adjacent vertices that are not discovered as yet, it is marked discovered. ae is identified as a back edge. The control goes back to a when it is marked discovered, assigned a finishing time and in turn the control goes to its predecessor, b. At b, the only adjacent vertex that has not been discovered yet is f. Thus, bf is added o the depth first tree and the adjacent vertices of f are explored. Let's suppose that g is chosen as the next vertex to be explored. fg is added to the depth first tree. The only directed arc from vertex g is to that of h which is then traversed. At h, there is an arc to f but it is a back edge and it is therefore marked as discovered and the control returned to its predecessor g which is then marked discovered as well.

Next, the algorithm keeps on returning the control to its immediate predecessors from f to b, then onto d and from d to i. The vertices g, f, b and d are all marked discovered and the control finally goes to i where it finds an unexplored vertex in j. j is then explored and marked discovered and finally the algorithm terminates in i. The flow of the algorithm is obvious from the depth first numbers assigned after each vertex and the finishing times signify the order in which each of the vertices are explored. In case of a graph where all the vertices are not connected, the algorithm begins exploring the vertices in a different subgraph after the vertices in the connected subgraph are explored.

The algorithm examines all the vertices in the graph. The vertices are assigned an initial state, undiscovered and each of the vertices are then traversed. The time required for the initial assignment is $O(N)$. Each of the edges incident upon a vertex are examined. Thus if we take the sum of the all the edges incident upon a vertex,for all vertices in the graph we have the set E. Therefore the time required for traversing all the edges is $O(M)$. The time complexity of the algorithm can be summed is therefore $O(N + M)$.

Breadth First Search

The Breadth First Search (bfs) Algorithm (see for example, [9]) is another search algorithm that uses local information of the nodes to traverse the whole graph. It is used as a basis for many important graph algorithms. BFS takes a graph G comprising of a set vertices V and set of edges E as an input. A root node s is selected as the starting point of the traversal and the bfs algorithm proceeds to create a bfs search tree that contains all the vertices reachable from the root node s. Unlike the Depth First Search algorithm, BFS discovers all the adjacent nodes of a vertex before moving on to the rest of the graph. It operates on a layer by layer basis, each layer being characterized by the distance to the source node. The nodes in layer $k + 1$ are those that are reachable from the nodes in layer k and have not been discovered yet. Thus any node that is at distance k from the source is discovered before the nodes at distance $k + 1$. The output of the algorithm is a breadth first search tree containing nodes that are at a particular distance from the source. The index k of the layer that a node belongs to, represents the distance from the source node.

Like the DFS algorithm, each node can have three states - undiscovered, discovered and explored as described above. In addition to the state each node has two other attributes associate with it - the distance from the source and the parent of the node. The Breadth First Search Algorithm uses a queue as a data structure to store all the nodes that have been explored. A brief description of the algorithm is given below. BFS starts with the source node s which is then marked as explored and inserted into a queue. The adjacency list of the source node s is examined and all the neighboring nodes are inserted into the queue. A bfs tree is created with s as the root node and distance from the source to itself is set to 0. The parent of each of the neighboring nodes of s is set to s. The distance associated with these nodes is set to one more

than that of the source node. The source node is then set to the explored state and removed from the queue. These nodes make up the next layer of the bfs tree. The algorithm then runs by examining each of the new nodes in the layer to find the nodes that are reachable from them. The distance of the new nodes are set and the nodes are marked discovered. This process of exploration and discovery continues until the algorithm reaches the end of the graph. The running time of the algorithm is again $\mathbf{O(N + M)}$.

Breadth First Search is widely used to find the shortest path between the nodes in a graph, to find the connected components in a graph, to check reachability from a particular vertex, to test bipartiteness of a graph and for many more purposes. For my project I did not need to use Breadth First Search. However, it could have been used in place of DFS to check the reachability of vertices and thus identify them as belonging to a particular strongly connected component of the graph. BFS frequently finds its use in computing the shortest path length in a graph. It can be used to compute the diameter of the whole graph, the components of the webgraph and also to measure the depth of the components. Since BFS proceeds on a layer by layer basis we can find the number of vertices on each layer. Thus by running BFS from the core, we can study the distribution of the vertices relative to the core.

Strongly Connected Component Algorithm

The strongly connected component algorithm is applied on a directed graph in order to find all the subgraphs such that each subgraph is maximal with respect to being strongly connected. The SCC algorithm partitions the directed graph into its strongly connected components. Depth First Search algorithm can be used to find a rooted directed forest of the digraph. DFS is then run on the transpose of the graph formed by reversing the arcs but considering the vertices in order of their decreasing finish times as computed by the first DFS. This outputs a depth first forest, each subtree constituting a strongly connected component. The time complexity of the algorithm described above is $O(N + M)$. A pseudocode of the algorithm is given below.

SCC(Graph G)

Run DFS(G) to find the finishing times $f[v]$ of all the vertices;

Find(G^T);
Run DFS (G^T) starting with the vertices in the descending order
of their finishing times $f[v]$;
The resulting depth first trees are each output as a
strongly connected component;

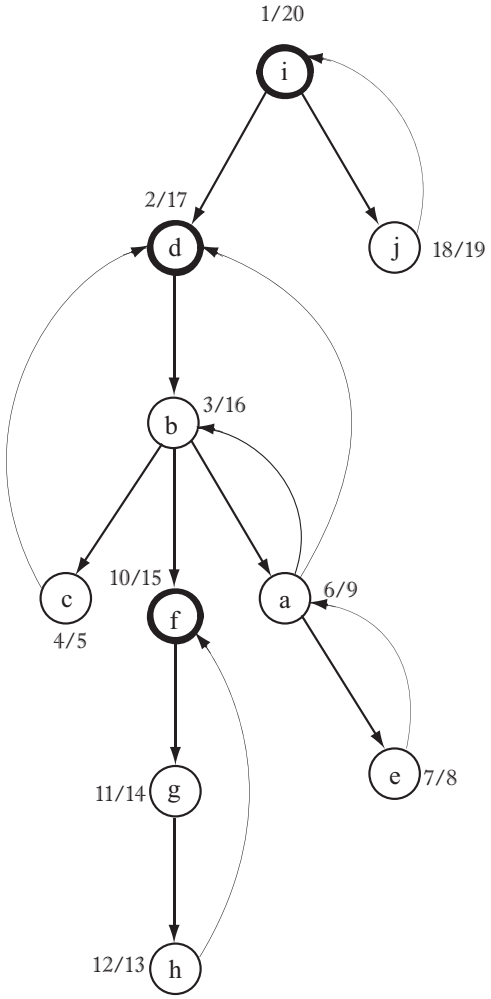
The ***forefather*** $\Phi(a)$ of a vertex x is the vertex y that can be reached by x and the finishing time in the depth first tree is always the highest. A vertex will have a forefather if and only if there is a cycle in the graph or the subgraph concerned. Thus d, i and f qualify as the forefathers of any of the vertices in the respective subgraphs in Figure 2.5 as illustrated by the depth first tree in Figure 2.6 (a). f has a finishing time higher than the rest of vertices in the subgraph $S = \{f, g, h\}$ and can be reached by the other vertices namely g and h. The main idea of the algorithm is to find the forefathers in the graph. By definition, forefathers are vertices with the highest finishing times and all the other vertices in the subgraph can usually reach them. The first step in finding the components of the graph is to run DFS on the directed graph to find the finishing times of the vertices. Now, the presence of a strongly connected component implies that there must be a path from a vertex to every other vertex within the component. The next step in the algorithm is to find the vertices that can reach the forefather. This is done by reversing the direction of the links in the graph and running DFS on the graph starting with the finishing times(forefathers have the highest finishing times). All the vertices that can reach the forefather have a lower finishing time and make up a strongly connected component. Thus once a component is found the Depth First Search Algorithm automatically chooses the forefather of the next component. Figure 2.6 is an illustration of how the strongly connected components of the directed graph in Figure 2.5 are found. i, d and f are the forefathers and running DFS on the directed graph with their links reversed results in the three components illustrated by the depth first trees in Figure 2.6 (b).

Tarjan's algorithm (see [9]) uses a similar idea to that of finding the subtrees using depth first searches. However it uses a stack and recursively examines all its adjacent nodes. The nodes are placed in the stack in the order in which they are visited and

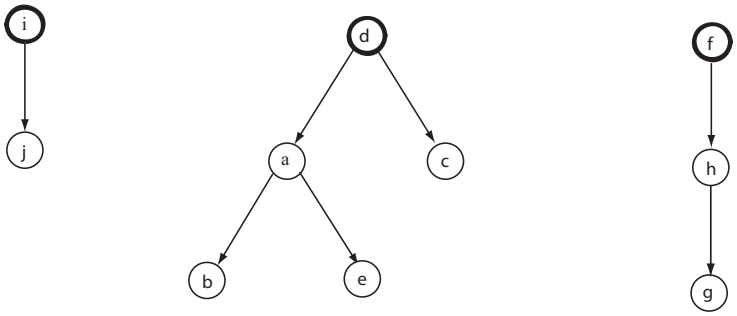
when the subtree is found the stack is emptied thus finding a strongly connected component of the graph. In a similar manner all the subtrees are found along with the root, thus partitioning the graph into its strongly connected components. The complexity of Tarjan's strongly connected component algorithm is also of the order of $O(N + M)$.

The Strongly Connected Component Algorithm is usually used to find all the connected components in a graph. It decomposes a digraph into its several subgraphs that are both disjoint and maximal. If we treat each of these connected components as a single node we obtain a directed acyclic graph. The resulting directed acyclic graph is called the **component graph** or the **condensation graph**. A component graph such as that of the directed graph in Figure 2.3 as shown by Figure 2.7 helps us understand how each of the strongly connected components interact with each other in the directed acyclic graph. The figure shows the interaction between the nine strongly connected components in the digraph. The seven components $C_1 \ldots C_7$ are connected to each other while C_8 and C_9 are totally isolated from the rest of the components in the digraph.

For my project, I used the Tarjan's Strongly Connected Component Algorithm in order to find all the strongly connected components of the whole Webgraph. The largest strongly connected component forms the core or the GSCC (Giant Strongly Connected Component) of the webgraph. The Strongly Connected Component Algorithm was also used on the individual components IN and OUT in order to find out the distribution of the strongly connected components with respect to their size. By treating each of the strongly connected components as a single node a condensation graph was obtained. Various features, including the indegree distribution, the outdegree distribution of the condensation graph were analyzed.



(a) Depth First Tree of the graph in Figure 2.5. The roots are marked in black and the depth first numbers and finishing times are noted on each of the vertices in the tree



(b) Depth first trees generated by running DFS on the transpose of the directed graph in Figure 2.5 starting out with the vertices in descending order of their finishing times. The roots of each of the trees are marked by a thicker border. Each of these trees represent a strongly connected component of the directed graph in Figure 2.5

Figure 2.6: Illustration of Depth First Search and Strongly Connected Component Algorithm on the directed graph in Figure 2.5

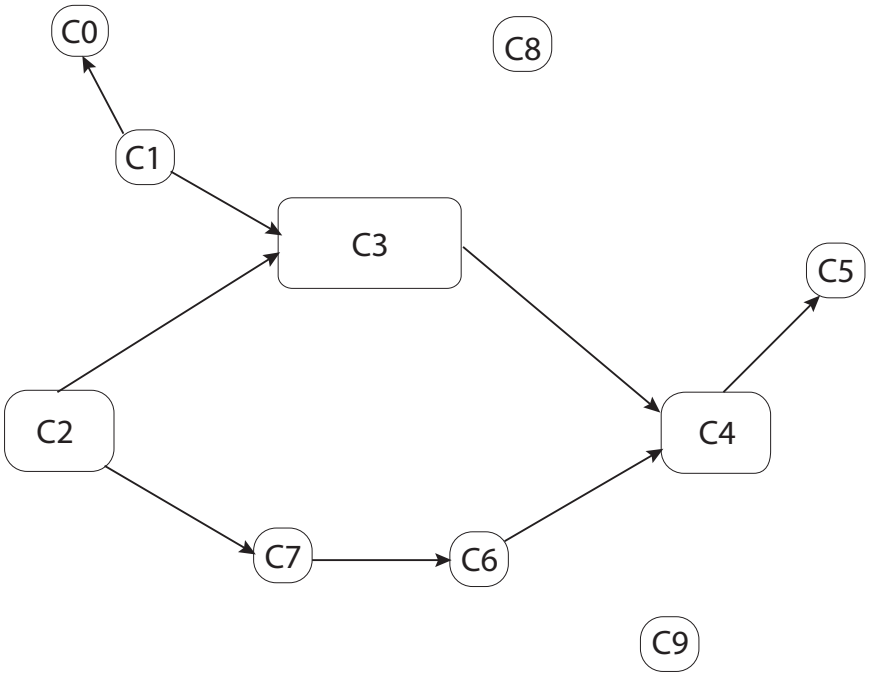


Figure 2.7: A component graph of the directed graph given in Figure 2.3

Chapter 3

Structure and Topological Properties of the Webgraph

The relation between size and the occurrence of any event to its rank is referred to as Zipf's law [7]. A text or a piece of writing can be viewed as a network of words connected to each other. Zipf was a linguistic professor who wanted to find the relation between the frequency of a word used in texts to its rank(referring to its usage). He observed that in English the most common words like "I" and "the" happen to cover most of the text as compared to words of lower rank such as "blasphemy". This observation was summed up mathematically.

$$f \propto r^{-a}$$

where f is the frequency of occurrence of an event, r its rank and $a > 1$ a fixed constant.

A 19th century economist, Vilfred Pareto [34] studied the distribution of wealth in a population. He observed that it was only a few people who contributed to a significant proportion of the income while the population was made up of a large number of people who made a very small contribution. Interestingly, Pareto was interested in the number of people that had income greater than a certain value. It was later observed that 20% of the population contributed to 80% of the income. This became known as the 80 20 rule that is so prevalent in many complex networks.

$$P[X > x] \propto x^{-k}, k > 1$$

where x is the income and X the number of people

3.1 Power Law in Web Graphs

For a long time it was assumed that the distribution of links in a complex network follows a Poisson distribution. This was mainly because of the insights of Erdös and Rényi who had explored random placement of links between the nodes in the network. Thus, it was assumed that the degree distribution of the links were random and uniform with most nodes having values on either side of the mean degree distribution.

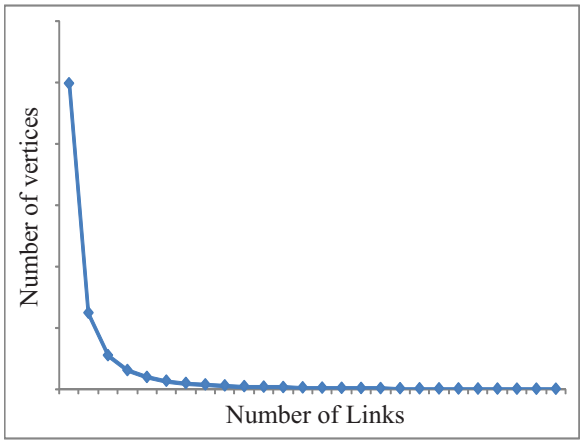


Figure 3.1: Curve showing a power law distribution

However, if we take a closer look at some of these networks in practice we would observe that they contained a high number of nodes with very few links while the main character of the network was provided by a few number of nodes with a lot of connections. A plot of the number of nodes against the number of links in a typical webgraph is given in the figure above. The distribution was a decaying function known as a power law distribution showing that the number of nodes decreased exponentially with the degree. A noticeable feature of this distribution is a long tail or a high number of nodes with a very few links. A double logarithmic plot for such networks revealed a straight line with a negative slope. Networks that contain nodes with a power law degree distribution are called **scale free networks**.

Barabasi et al. [11] pointed out that the ER(Erdös-Rényi) model failed to take into account the dynamic nature of complex networks and the linking pattern evident in such large networks. The World Wide Web, which is the main concern of this project, is a large complex network that evolves with time. Unlike the static random graph model induced by Erdös and Rényi, the number of pages in the world wide web increases and the web authors of newly created pages tend to link to pages that are already popular. This method of linking to vertices with a large number of connections within the network is called **preferential attachment**. Thus the World Wide Web self organizes itself into a scale-free state that can be explained by the preferential attachment mechanism described by Barabasi et al [12]. It will be discussed in subsequent sections but for the time being let's take a look at the various instances of power laws in the World Wide Web.

3.1.1 Indegree Distribution

Power Law for indegree distributions in a web graph: The probability that a page w has $i > 1$ connections from other pages in the webgraph is inversely proportional to i^{-x} for some constant $x > 1$ (x is the slope of the double logarithmic plot for number of pages against the indegree **i** of the page.

$$\mathbf{P}[\mathbf{w} = \mathbf{i}] \propto \mathbf{i}^{-\mathbf{x}},\ \mathbf{x} > \mathbf{1}, \mathbf{i} > \mathbf{1}$$

It was first suggested by Barabasi et al [11] that the indegree distribution of a webgraph follows a power law distribution. They studied the webgraph of 325K pages from the nd.edu domain and the indegree distribution confirmed their assumption. Kumar et al [27] carried out experiments on a 1997 crawl from the WebBase Project at Stanford. Their crawl containing 40 million pages was a much bigger crawl than that of the NotreDame domain and represented a better portion of the web. Surprisingly, the exponent value for the indegree distribution for both the experiments were found to be 2.1. Later experiments by [14] on a much bigger Altavista Crawl of 200 million pages obtained at two different times, May and October of 1999, confirmed the same value as shown in Figure 3.2. The double logarithmic plot of the indegree distribution shows a heavy-tailed distribution that points to the presence of a significant number of pages with a high indegree. The sudden surge shown by the collection of red dots in the graph is due to spammers creating an excessive number of connections to a certain web page. The indegree distribution for webgraphs was also studied at various scales. Ricardo Baeza Yates and Carlos Castillo [10] studied the samples of the web by confining them to national web domains. They studied the web graph of pages that are within the country domains of Brazil, Chile, Greece, Italy, Korea, Spain and the UK and power laws were evident for all the country domains mentioned although the exponent value in such cases were observed to be 1.9 ± 0.1. The size of these crawls varied between 3.3 million to 41.3 million pages, the .it (Italy) domain being the largest and the Chile domain being the smallest. It is worth noting here that another study of the crawl on the .SK (Slovakia domain) [13] produced a power law distribution with an exponent of 1.86 which is consistent with the results obtained in [10]. A study of the webgraph at a finer level was that of the academic web space carried out by Mike Thelwall and David Wilkinson [33] from the university of Wolverhampton. They took crawls on the Australia, New Zealand and the UK web

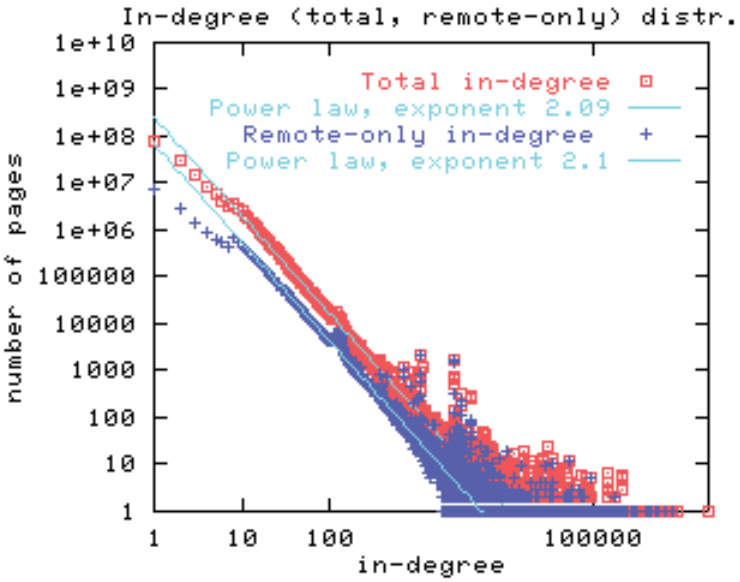


Figure 3.2: In-degree distribution of the Alta Vista crawl done by Border et al [14]

domain and filtered out sites that were related to only academics. The link structure of the academic web spaces thus obtained also showed power laws for the indegree distribution.

3.1.2 Outdegree Distribution

Power Law for outdegree distributions in a web graph: The probability that a page w has o connections to other pages in the webgraph is inversely proportional to o^{-x} such that $x > 1$, $o > 1$.

$$\mathbf{P[w = o] \propto o^{-x},\ x > 1, o > 1}$$

The outdegree distribution was also found to follow the power law by both Barabasi et al [11] and Kumar et al [27]. However, pages with a low outdegree did not seem to fit that well into the powerlaw distribution. Thus the outdegree distribution wasn't a perfect power law. Experiments on an Altavista crawl of 200 million pages by Broder et al [14] showed that the exponent value for both the May 99 and Oct 99 crawls were found to be 2.72, as shown in Figure 3.3. It can be noted here that the slope of the distribution has been calculated by leaving out pages with a very low outdegree. The value below which pages with out degree distribution deviates from the power law distribution is called the cut off point.

Ricardo Baeza Yates and Carlos Castillo [10] have noted the exponent value of the

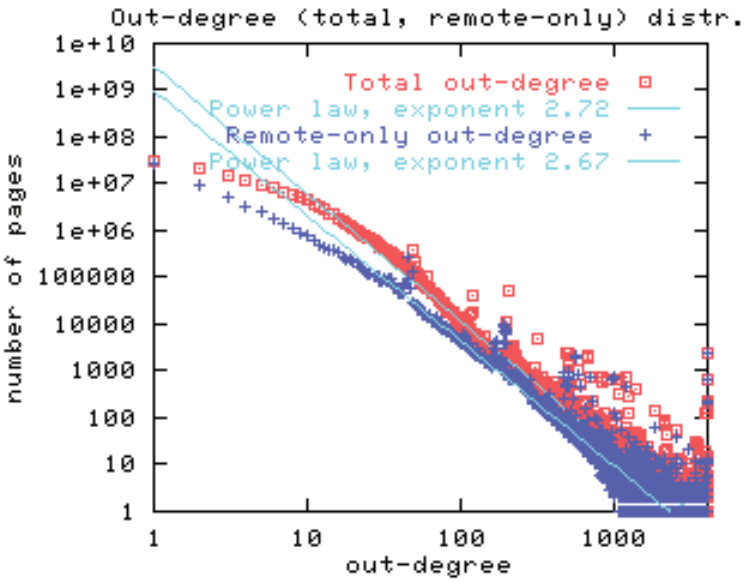


Figure 3.3: Out-degree distribution of the Alta Vista crawl done by Border et al [14]

outdegree distribution of national web domains for both pages with small outdegree and large outdegrees. The exponent for large out degrees which is our main concern has been found to be 2.8 ± 0.8 while that for small out degrees were 0.6 ± 0.2. Another study of the crawl on the national web domain of Slovakia (.sk domain) [13] produced a power law distribution with an exponent of 3.66 for the outdegree distribution.

3.1.3 Distribution of the size of connected components in a web graph

The distribution of various sizes of the connected components of a webgraph was also examined by Broder et al [14]. The sizes of the strongly connected components of the webgraph they studied followed a power law distribution with an exponent of 2.54. By treating the links as having connections in both directions they verified that the weakly connected components also followed a power law distribution with the same exponent. Debora Donato et al. [18] carried out experiments to examine if power laws could be applied to the individual components of the web graph. Their experiments confirmed the presence of power laws for all the components with respect to their indegree distribution, outdegree distribution and the distribution of the size of the strongly connected components. Thus the number of the strongly connected components decays exponentially with the size of the component as can be seen by a log-log plot in Figure 3.4.

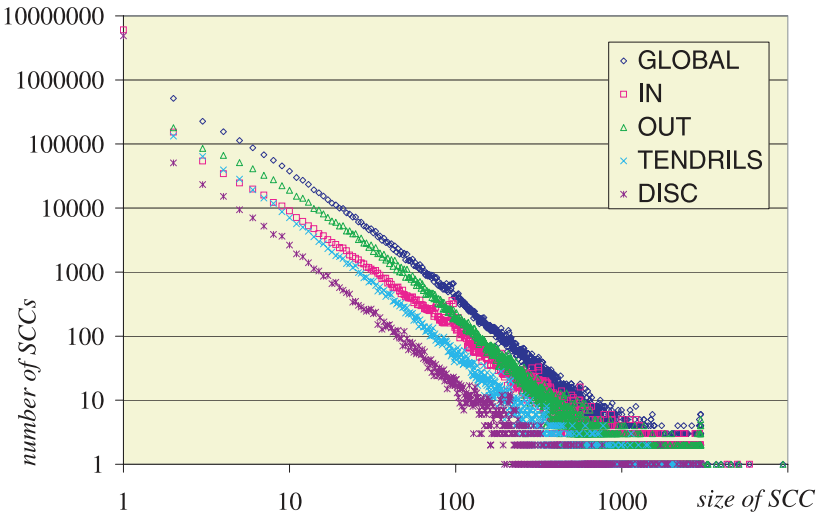


Figure 3.4: SCC distribution of the components of the web graph [18]

The total number of pages in a site is also known to follow a power law. Studies by Adamic and Huberman [23] from an Alexa and Infoseek crawl of sizes over $259K$ and $525K$ sites showed log normal distribution for the number of pages in a site. Thus it can be observed that power laws are really quite widespread when it comes to webgraphs either with respect to the macroscopic structure of the webgraph or the structure of its components.

3.2 Community Structures in the Webgraph

Another common feature observed in webgraphs is the presence of **community structures**. A web community is characterized by very strong linkage pattern between the websites in the community. For example, the presence of a bipartite clique reflects a very strong community structure. The bipartite clique essentially comprises of two sets, the set of **Fan pages** F and a set of **Center Pages** C. The set of F pages comprise of pages that have links to another set of pages. The set of C pages are pages that are referred to by the F pages. The two sets form the bipartite core. New websites are being created everyday and it is up to the creator of the web site to link to already existing websites. Web authors usually link to websites that are popular and relevant to their topic. Thus websites that contain important information on a particular theme or topic and are regarded to be of some importance are referenced

Figure 3.5: An example to illustrate clustering of webpages [4]

often by new websites with a similar theme. These popular pages on a particular theme are known as **authoritative pages**. The subset of pages that reference them constitutes the **hub pages**. The collection of hub pages that reference the authoritative pages form a bipartite core comprising of a dense linkage between the hub pages and the authoritative pages.

A good hub refers to several good authoritative pages while a good authority is usually referenced by several good hubs. These hubs are responsible for holding the whole webgraph together. They connect to a lot of pages who have very few connections (one or two incoming links) and allow them to remain connected to other pages in the webgraph through the hubs.

In fact it has been observed that although webgraphs are resistant to random removal of nodes the webgraph would totally fall apart if most of these hubs are systematically removed. Hubs have outgoing links to many pages. They are also

the only link for many pages and are responsible for providing connections between different sets of pages. If the hub pages are removed major connections between pages in the graph will be lost. Pages whose only connection was a hub page will become totally isolated. Thus the graph will begin to break down into smaller subgraphs if the hubs are removed. This is why virus attacks targeted at hubs can cause the whole webgraph to disintegrate quickly.

Websites based on a common theme or common area of interest tend to group together and form clusters or groups. Study of such linkage pattern of websites clustered by a particular theme allows us to extract the web communities. Kumar [27] first studied algorithms to identify the several thousand communities within webgraphs while Kleinberg [25] has also studied how to extract information from the world wide web to identify the communiites and their authoritative pages. An example of pages grouped by themes is given in the Figure 3.5. The groups in blue fall under the category of travel while those in red are mainly related to newspapers or news media in Bangladesh. The Green Cluster tends to organize into a separate group although they fall into the category of News Media. The pages in this group consist mainly of newspapers in Pakistan and are linked to the red cluster through a page that covers Foreign News.

3.3 The Structure of the Web

The web is an evolving complex system with the number of web pages and the hyperlinks constantly changing. In a recent survey, Gulli and Signorini [22] have found that there are over 11.5 billion pages (indexable) and several billion links while the size of the web was found to be around 200 million pages by an Altavista crawl done by Bharat and Broder in 1997. New web pages are being added all over the world, while some of them are being taken down. New hyperlinks might be added to an existing page or even changed to point to a different page (rewiring) and so forth. In spite of the random way in which the WWW appears to be changing, a study of the web graph at both the macroscopic level and the microscopic level has revealed a specific structure. Even the manner in which the WWW is growing follows a certain pattern. Recently the researchers have taken a great deal of interest in the web graph because an understanding of the web graph will allow us to design efficient algorithms

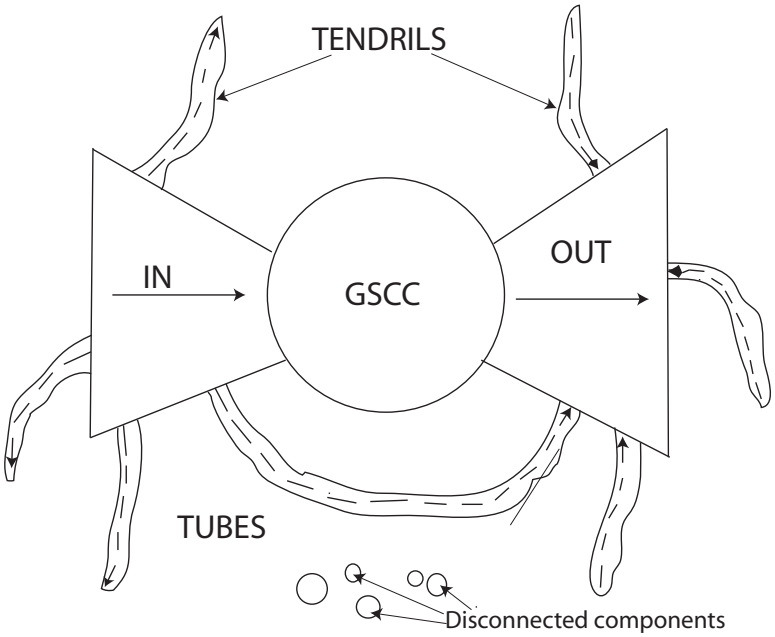


Figure 3.6: The Bow Tie Structure of the Webgraph (adapted from Broder et al [14])

for the search engines. The link structure of the web can be exploited to evaluate the importance of a web page. PageRank is one such algorithm that uses the link structure of the web and the kind of sites that link to a page to rank it. This section describes the structure on the web as a random graph but before proceeding further a few terms that will be used through out the section need to be defined.

The web graph has a "bow-tie structure", as was first depicted by Broder. [14] The webgraph can be broken down into four basic components. It has a giant strongly connected central core called the GSCC(also referred to as SCC in some literature). The nodes of IN are those that can reach the GSCC but cannot be reached from the GSCC. As for the nodes in OUT they can be reached from the GSCC but cannot reach the GSCC. Although GSCC separates IN and OUT there may be paths from IN to OUT that do not intersect the GSCC. These form the TUBES. TENDRILS consist of those nodes that can either reach nodes in IN or be reached by nodes in OUT but are not in (GSCC $\bigcup$ IN $\bigcup$ OUT). The rest of the nodes are classified as disconnected components since they are totally isolated. The nodes in the disconnected components cannot reach any of the three main components (GSCC $\bigcup$ IN $\bigcup$ OUT) and cannot be reached by them. A brief description of the components is given after the following proposition that shows that the whole webgraph breaks up into disjoint components

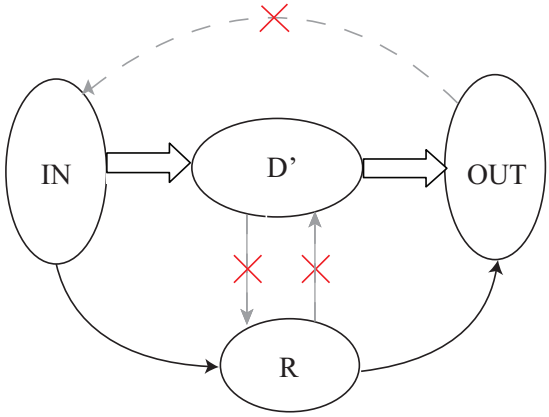


Figure 3.7: Illustration of the interaction between the various components of the digraph

that are linked to each other through various links.

Proposition 1 *A digraph D decomposes into*

$$D' \bigcup IN \bigcup OUT \bigcup R$$

where

- $V(D) = V(D') \bigcup V(IN) \bigcup V(OUT) \bigcup V(R)$,
- D' *is a strongly connected component of* D,
- $IN = \{v \in V(D) - V(D') : \exists$ *a directed path from* v *to a vertex of* $D'\}$,
- $OUT = \{v \in V(D) - V(D') : \exists$ *a directed path from a vertex of* D' *to* $v\}$, *and*
- R *is the subgraph induced by* $V(D) - \{V(D') \bigcup V(IN) \bigcup V(OUT)\}$.

Note that if $v \in IN$, then $v \notin D'$ and $v \notin OUT$ and $v \notin R$. Now if $v \in OUT$, then $v \notin D'$, $v \notin IN$ and $v \notin R$. Also, $v \in D' \rightarrow v \notin IN$, $v \notin OUT$ and $v \notin R$. Finally if $v \in R$ then $v \notin IN, v \notin OUT$ and $v \notin D'$ Therefore we can conclude that $V(D')$, $V(IN)$, $V(OUT)$ and $V(R)$ are disjoint sets and partition $V(D)$.

Every pair of vertices in a strongly connected component is equivalent, that is, there is a path from v to u and u to v for very pair of vertex, be it direct or through

intermediate vertices. Now if we claim that a vertex v belongs to both IN and OUT then a vertex u in IN and w in OUT would be equivalent to v. Thus by the transitive property, u and w would be equivalent making it a single strongly connected component. This is clearly not the case and therefore we show that all digraphs decompose into disjoint subsets. In fact, all random digraphs would break up into these four components [15]. At this point it would be interesting to ask if there is anything that actually affects the size of these components. For our research the size of the bow tie of the webgraph is quite important. In fact, Dorgovtev et al [20] showed that the size of the four main components of a directed graph depends on various properties of the directed graph. From the point of view of our research we would like to look at the various sizes of the bow tie seen in webgraphs later. A brief description of the components of the webgraphs is given below.

GSCC (Giant Strongly Connected Component): The central core or the strongly connected component of most webgraphs comprises of about a quarter of the whole graph. In an experiment on 200 million pages from a web crawl by Alexa, 56 million were found to be in the core of the webgraph. Every page in GSCC can be reached by any other page within the core. The GSCC acts as the bridge between the other components and the pages in the GSCC have a very good connectivity themselves. Pages within the core tend to have a very high connectivity and are referred to as hubs. Popular websites like Yahoo or Google that provide the connectivity to many websites by listing them on their page would fall in this category. The concept of pages acting as hubs will be discussed in more detail in a later section.

IN: This portion of the web graph consists of pages that can communicate with those of the core but not in the reverse direction. In other words, there is a path from each vertex of IN to GSCC but there is no path from the pages in the GSCC to IN. Therefore, BFS in the forward direction will explode from a starting point within the IN. Nodes within this component are mainly new pages that have not yet been discovered by the outer world. Broder [14] found these pages to comprise of roughly a quarter of the whole web graph. Ideally, this subset includes the newly created pages that aren't quite known to the outside world and haven't therefore been linked to. These pages do however tend to link to popular websites or sites that we refer to hubs and are known to have a high degree of connectivity.

OUT: Pages that can be reached from those within the core but not in the Reverse direction make up this component of the webgraph. OUT also often makes up about a quarter of the whole web graph and the pages within this component are typically the corporate websites. Company sites like those of the Starbucks or Air Canada have incoming links from hubs like Google or Yahoo. A Breadth first search, run from a starting page within this component will explode along the reverse links. However, a web surfer browsing through a page in OUT, or a crawler starting in a page in OUT is most likely to get stuck since there are no links that go back to the core.

TENDRILS and TUBES: Tendrils consist of pages ($notin(GSCC \bigcup IN \bigcup OUT)$) that are either reachable from IN or reach to pages in the OUT. They do not have any incoming links to or from any of the pages within the giant core. As a result they tend to be isolated. The pages that link to pages in OUT but cannot be reached by those in OUT or pages from any other component make up the OUT_TENDRILS. IN_TENDRILS on the other hand are those pages that can be reached by some vertices within IN but cannot reach any vertex in IN. Thus we notice that the direction in which the two different TENDRILS link are exactly opposite to each other. TUBES, on the other hand comprise of vertices that form a pathway between IN and OUT. Therefore the pages within the TUBES can be reached from some vertices within IN and they in turn can reach some vertices within OUT.

DISCONNECTED Components(R) : Disconnected Components is a collection of vertices of the webgraph that can neither be reached by any vertex in any of the other components (GSCC, IN, OUT) of the webgraph, nor do they link to any vertex present in the other components of the webgraph. This subset of vertices are the ones that do not belong to the giant weakly connected component of the webgraph. DISCONNECTED Components might consist of pages within a site. Although the pages within the site do not have any incoming links or outgoing links from and to the rest of the webgraph, they can actually have interconnections between themselves. Isolated groups would usually fall in this category. In fact a study of the Chilean webgraph revealed that 50% of the pages within the .cl domain fall into this subset. Some examples of disconnected components have been given in [17]. They used a spectral method to separate disconnected and nearly disconnected components of the webgraph.

Proposition 2 *Let $p \in (0,1)$ be fixed and $n \geq d$. Let $D_{n,p}$ be the probability space of all simple digraphs on $\{1....n\}$. Then $\lim_{n\to\infty} Prob(D \in D_{n,p}$ is strongly connected$) = 1$.*

Proof: Let $D_{n,p}$ be the probability space of all simple digraphs on $1 \ldots n$ and let u and v be distinct vertices of D. Let $E_{u,v}$ be the event that there is no vertex z such that (u,z) and (z,v) are arcs. Since the probability of both the arcs being present is p^2, the probability of the event $E_{u,v}$ occurring is given by

$$\text{Prob}(E_{u,v}) = (1-p^2)^{n-2}.$$

Now if D is not strongly connected then there are distinct vertices u and v such that u and v are not joined by a path of length 2.

$$\begin{aligned}
\text{Prob}(D \text{ is not strongly connected}) &\leq \text{Prob}(\bigcup_{u,v} E_{u,v}) \\
&\leq \sum_{u,v} \text{Prob}(E_{u,v}) \\
&= \sum_{u,v} (1-p^2)^{n-2} \\
&= n(n-1)(1-p^2)^{n-2} \\
&\leq n^2(1-p^2)^{n-2}.
\end{aligned}$$

Now, if we take the natural logarithm of the last expression above we have

$$\ln\left(n^2(1-p^2)^{n-2}\right) = 2\ln(n) + (n-2)\ln(1-p^2).$$

The second term in the expression grows is negative and grows faster than ln(n) as $n \to \infty$, so we see that

$$\text{Prob}(D \text{ is not strongly connected}) \to 0 \text{ as } n \to \infty.$$

Therefore Prob(D is strongly connected) $\to 1$ *as* $n \to \infty$.

Thus a directed graph is almost always strongly connected, so the bow tie structure of a random digraph is almost always just the GSCC.

We have already showed that a webgraph can be decomposed into its disjoint components of which we have the bowtie comprising of the three most important

sets IN, OUT and SCC and the rest of the components, namely Tendrils, Tubes and the disconnected components. Dorgovtev et al [20] showed that the size of the components of a directed random graph depends on various properties of the directed graph that would determine their size distribution. It would be interesting to take a look at the various distribution of the size of components in a webgraph. In Table 3.1 we will take a look at the various sizes of the bowties and its components with respect to the whole webgraph The crawls examined vary with respect to the size and scope of the webgraph, the crawl methods and the time the experiments were undertaken. It can be noticed that irrespective of the size of the crawl, the bowtie comprises of a major portion of the total webgraph as can be expected. For all crawls that are confined to specific country domains like Italy, Slovakia, UK or the IndoChina, the bowtie comprises of over 98% of the total graphs while that for the Altavista crawl on 200M vertices studied by Broder et al [14] stands at the smallest at 70% in comparison to the rest. Although experiments on the Altavista crawl had revealed that the bowtie decomposed into its constituents of roughly the same size, it was seen to vary in a great deal in other crawls. The core or the GSCC component of the webgraph has been consistently observed to comprise of the larger portion of the web with an exception in the WebBase crawl where the OUT component dominates the webgraph slightly over the core. However, it is worth noting here that the IN component has been found to be the smaller of the components of the bowtie in size with the smallest as low as a mere 0.01% of the webgraph in the Slovakia webcrawl to the largest as high as 21% in the Altavista crawl. The OUT components seem to comprise of a fair share of the webgraph and is in fact is found to be even bigger than the core in the WebBase crawl. Thus we can observe bowties of all shapes and sizes although most of them are characterized by a large core.

Although Broder et al. [14] looked at the macroscopic structure of the web graph and presented the bow tie structure comprising of four main components, the inner structure of the components were not considered. They left it as an open question for future research and it was eventually taken up by Donato et al [18]. They studied the inner structure of the components IN, OUT and the GSCC. Since it is very difficult to retain web graphs in memory that are of the order of gigabytes in size it is essential to use external memory algorithms. They implemented a number of external

and semi-external memory algorithms to carry out experiments on four crawls from different domains. They showed that power laws were prevalent in both IN and OUT for the indegree and the outdegree distribution. However the components themselves did not break up into the usual bow tie structure. The size of the bowtie varied from crawl to crawl (please refer to Table 3.1) and moreover, there was no significant giant component present in any of its components. On the contrary, these components turned out to be quite shallow and fragmented. A brief description of their experiments and findings will be described next. The core was condensed to a node and breadth first search was run from the core both along the forward links and the backward links. The total number of layers obtained by running BFS until all the nodes in the component have been reached is the depth of the component. Experiments along the forward links revealed that most of the nodes of OUT were actually concentrated closer to the giant component. About 90% of the nodes of the components could be found in the first 5 layers and although the depth of the subset OUT for the WebBase webgraph was found to be 580, due to a single chain of vertices. Similar results were obtained for IN, thus confirming that both the components were shallow. Furthermore there was no sizable strongly connected component present in either of the components that could make up the core of a bowtie. A daisy chain structure of the web was suggested by Debora et al [19] to describe the finer structure of the components of the web graph. The core was studied in relation to its connectivity to IN and OUT. A large proportion (about 80% for the WebBase Graph) of the nodes was found to be connected to the components IN and OUT. The core was also found to be resilient to targeted attacks. Removal of many nodes of very low indegree was required to be removed before they were reduced to a very small size thus confirming it to be well interconnected.

3.4 Diameter of the Web Graph, Small World Phenomenon and Growth Dynamics

The Small World Phenomenon [24] states that the diameter of the graph is small and any pair of nodes in a graph is connected by a short path. It is observed in various different systems like the biological systems, social networks, and various technological networks (citation networks, collaboration graphs, the Internet and the World Wide

Web). It first came up in the context of social networks. Stanley Milgram [30] chose two random individuals A and B, in Nebraska and Massachusetts respectively. A was asked to send out letters to all his contacts who in turn would send out letters to their acquaintances. The results were surprising and it was noticed that A and B were separated by at most six other people and thus it came to known as the **Six Degrees of Separation**. Watts and Strogatz put forward their own model which started with a lattice containing vertices in a ringlike structure. The vertices are then rewired to $2m$ neighbors where m is the average degree of the vertices. It showed a small world phenomenon whose clustering coefficient was larger than that of the random graph.

In this section, we are interested in the small world phenomenon produced by the web graph. The topology of the web graph is such that not only is any pair of page a few clicks of each other but it also exhibits a high clustering coefficient. The growth of the web is unabated and the web graphs keep on growing at an alarming rate. Thousands of pages are added everyday but the web self organizes itself. Had it not been for the small world property, it would have been difficult for any user to navigate from an arbitrary page to reach any other page within the web. The size of the web was estimated to be 8 billion pages by Kleinberg et al, [25] at the time of their experiments. They used a robot to extract information about the local connectivity of the pages. The next layer's connectivity information was retrieved recursively and added to a database. In this way, information about all the indexable websites(that is sites that can be reached by the crawlers) was recorded.

The average distance between any pair of pages in the graph, *average d* was found to be 18.59 (19 clicks). From the data obtained they plotted values for average d and established a relationship between the average distance for any pair of pages and the size N of the graph.

$$average\ d = 0.35 + 2.06 * \log N$$

Due to the logarithmic dependence of d on the size of the web graph, the increase in the average distance between the pages is hardly noticeable. Broder et al [14] carried out experiments on an Altavista crawl in 1999. The other key feature of web graphs is a high clustering coefficient. Adamic [8] studied web graphs at the site level. He used data from an Alexa crawl on $259,794$ sites for which *average d* was found

	Altavista 1999	*WB* 2001	*WB* 2003	*Italy*	*Indo China*	*UK*	*SK*	*TREC*
Vertices	$200M$	$135.7M$	$49.3M$	$41.3M$	$7.4M$	$18.5M$	$50.6M$	$1.22M$
Links	$1.5G$	$1.18G$	$1.19G$	$1.15G$	$194.1M$	$298.1M$	$1.9G$	$11.16M$
GSCC	28%	32.9%	85.87%	72.3%	51.4%	65.3%	70.8%	74.4%
IN	21%	10.6%	2.28%	0.03%	0.66%	1.7%	0.01%	1.79%
OUT	21%	39.3%	11.26%	27.6%	45.9%	31.8%	29%	12.37%
BOW	70%	82.8%	99.41%	99.93%	97.93%	98.8%	99.81%	98.56%

Table 3.1: A study on various bowties with respect to the size of its components. Data has been obtained from webcrawls varying in sizes and the year the crawl was done [18] [31] [32] [13]

to be only 3.1 and the clustering coefficient $c = 0.1078$ as compared to 2.3×10^{-4} for random graphs. The largest SCC exhibited a small world phenomenon. They noted that the SCC that contained pages from several several sites were more useful than those belonging to a single site.

Netcraft is a company that carries out research on various aspects of the Internet Sites. A survey reported on August 2008 found the total number of sites to be 176,748,506. [3]. The number of sites have really increased at a rapid rate over the years. The total number of sites at the end of 2002 was somewhere around 35 million which is about one fifth of the total number of sites now. The graph in the figure **??** shows the exponential growth of the number of sites over a span of 5years The number of sites more than doubled to about 74M sites from 2002 to 2005, a span of *three* years but in the next *two* years the number almost doubled from 74M to 149M. Thus we can

clearly see the rate of increase in sites is constantly increasing as can be depicted from the graph in the figure shown. However, it should be noted here that this increase only represents that at the site level and not the actual increase in the total number of webpages. A rough estimate of the total number of webpages is given below.

Maurice de Kunder [6] devised a mechanism for estimating the size of the World Wide Web. He used four search engines, namely Google, Yahoo, Windows Live Search and Ask to find the number of indexable pages within the web. Since different search engines use different strategies to index the web, they come up with varying information on the number of sites that could be reached. Now, first of all, 50 words were chosen from an offline text collection in DMOZ [2].The words chosen are such that it covers all the intervals available. The 50 words are then sent to each of the 4 search engines. The percentage of documents in which a particular word was found helps us to estimate the size of the total web. The results obtained for each of the words are then averaged for a particular search engine. In this manner the estimations are calculated for each of the search engines mentioned. Once the results for the estimations are computed the overlap of the sites has to be subtracted from the sum of all the estimations. Since the overlap is overestimated, the estimate for the total size of the World Wide Web turns out to be an underestimate. There are several orderings available as the overlap has to be subtracted in sequence. The size of the indexed web obtained as of November 12, 2007 is 24.45 billion using the order YGWA for the search engines. The total number of pages indexed by Yahoo, Google, Windows Live Search and Ask as of 12th November is roughly 23 billion, 8.5 billion, 13.8 billion and 5.9 billion respectively. Thus this result for the total indexable web size is estimated to fall between 15 billion and 30 billion documents. It was only in 2005 that Yahoo announced that they were able to index as many as 19.2 billion documents [29] which was more than twice the size indexed by Google at that time (8.1 billion documents). However it is not possible for any search engine to index all the webpages present in the web. Therefore, there are many webpages present in the world wide web that are not taken into account. In addition, due to the dynamic nature of the web, pages are continually being added and some are being deleted as well. It takes a while for search engines like the Google and Yahoo to update their records and the old information about webpages is usually retrieved from the cache

which is not up to date.

3.5 Self Similarity in the Web

The Web graph has a fractal structure which is exhibited under different contexts and at various scales. By breaking down the whole webgraph into smaller fractions we notice a very similar structure to that of the whole webgraph. The self similar nature of the web graph was recently studied by Dill et al [16] and they showed that statistical dependence was evident in subgraphs containing sites that were cohesive. Such collection of websites contained pages that shared a common context, be it content or sites they belonged to. They formed meaningful units, called **Thematically Unified Clusters**. Experiments on the link structures of the TUCs revealed a bow tie structure which is just similar to that of the whole web. Moreover, they were connected by a strong backbone that was robust and highly connected. The web graphs were grouped under various contexts – content, location and geographical location.For each of the contexts the basic characteristics of the web graph, namely the in-degree/out-degree distribution, strongly connected component /weakly connected component size/ bow tie and community structure were examined. Results showed amazing statistical similarities for the units classified by different contexts.

Content: In order to find the collection of websites that shared the same topic, a set of keywords was used. The sites that contained all the keywords were combined into one subset. The sub graphs generated had websites of the order of 105. Experiments on the link structure of the websites in the sub graphs confirmed the basic characteristics found in a web graph. However, sub graphs formed by using comparatively new topics lacked a well defined bow tie structure found in popular keyword set graphs.

Location: Large sites, sub domains and intranets happen to have a lot of web pages that share a common interest and it makes sense to study the link structures of such intranets. A crawl on such intranets can be studied in order to verify the fractal nature of the web. Dill et al [16] used the IBM intranet and several sub domains each containing about 10000 pages. 82% of the nodes in the IBM intranet were inside the SCC while for the subdomains it was about 40%. Larger sites showed better bow tie structures than the smaller ones.

Geographical Location: Local information could be very important to businesses.

Websites with a common geographical location tend to share a common context. The Sites that have geographical cues like ZIP, Phone numbers and Addresses can be grouped together to form a TUC by using databases of latitudes and longitudes.Dill studied the structure of the SCCs that showed that the TUC s were themselves very well connected to each other through a navigational backbone.Thus the web graph can be broken down into several TUCs, each of which shows characteristics similar to that of the web. These TUC are in turn bound by a navigational backbone that overlap the clusters and provide robustness and high connectivity between the TUCs.

Chapter 4

Webgraph Models

The Web is already a huge repository of information. Several billions of hyperlinked documents are already on the web and several millions more are being added every month. It is growing at a phenomenal rate and scientists are finding it extremely difficult to map the whole web. Though the web began its journey with the objective of share documents with the scientific community it has found its uses in the commercial arena as well as in that of social interactions. Current technology is finding it difficult to cope with the rapid rate at which it is expanding. Crawlers are finding it exceedingly difficult to gather information about the whole world wide web. To date there aren't any search engines that can map the whole world. On top of that, the sheer size of the web is difficult to manage and store. Efficient compression algorithms need to be developed in order to study the link structure of the web. In order to get around these problems we need to develop a model that can represent the web as accurately as possible with respect to its structural and topological properties as well as the evolving nature of the web. Some of the motivations for building models that are representative of webgraphs are given below:

- Understand the evolution of the web better.
- Develop efficient search algorithms to retrieve information.
- Discover hidden characteristics and patterns in the webgraph.
- Use the the webgraph models for computationally difficult problems.
- Testing web applications on a smaller scale.

The use of an accurate webgraph model that incorporates the basic structural and topological properties of a webgraph would help us understand the evoloution of the web. We would be able to grasp why certain web authors link to particular sites and why webpages on a particular topic tend to cluster together or why certain pages tend

to link to many other pages while the majority of the pages in the webgraph have low connectivity. A good model will help us understand all the intricacies present in the linkage pattern in a webgraph and its growth dynamics. Models will help design more efficient search strategies to find sites that are relevant to the topic. In addition, they will help understand the link structure of the web and thus enable us to list more websites on a particular topic. Information retrieval would therefore be more efficient and thus help the scientific community even better.

The web exhibits several interesting topological properties both at the microscopic and macroscopic level. There might be many such interesting patterns that can be revealed if a good model for the webgraph can studied in detail. Besides, the phenomenal size of the webgraph limits us to test many graph theoretic problems that are difficult to compute for large graphs. Models of webgraphs would be very helpful in this context. There are also many web applications that can be tested on a smaller scale using webgraph models. This would allow us to develop efficient applications and therefore enhance their performance. Now let us look at what would make a good model for the webgraph.

A webgraph model will be a good representative if it can capture the dynamic nature of the web. The Web is continuously changing both in size and content. new links are constantly being added between the vertices already present in the webgraph. Some web authors might stop linking to a site and link to another site that is already present. In other words rewiring of links is also a possibility with webgraphs. Web authors tend to link to pages that are already popular, relevant to their theme and are rich in information. Thus there is a pattern in which in the links are being created between new vertices and the existing ones that should be taken into account. The model should include both growth and deletion of the vertices.

4.1 Random Graph Model

The classic theory of random graphs, studied in detail by Erdös and Rényi [21] in 1959 has served as the foundation for many real world networks in domains ranging from biological networks to information networks. They proposed a probabilistic model by which a random graph could be created. $G(n, p)$ and $G(n, m)$ are two different ways in which a random graph can be created. $G(n, p)$ refers to a graph of size n where

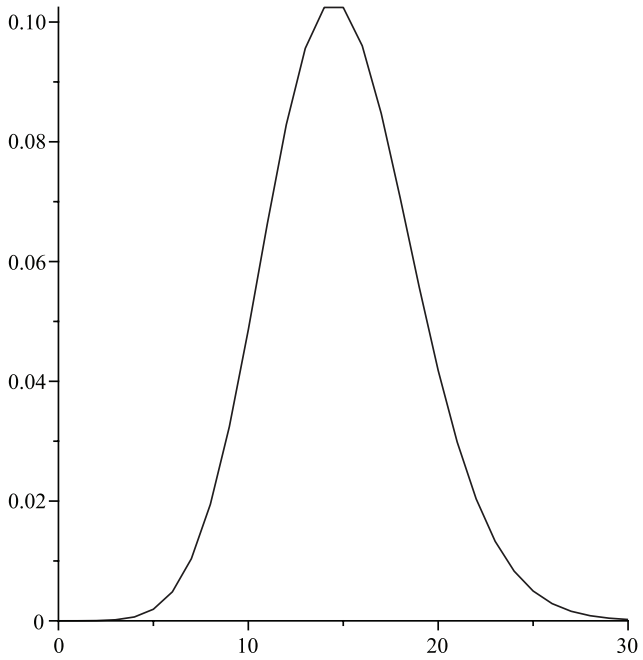


Figure 4.1: Degree distribution for a random graph model

the probability of having an edge between any of the two vertices is p. $G(n, m)$ on the other hand is graph with n vertices and m edges are created at random between the vertices.

In this section we will discuss the more popular $G(n, p)$ model. Now let us suppose we have a graph of size n. The number of possible edges between the n vertices is $\frac{n(n-1)}{2}$ and each exists with a probability p . Now let's look at the degree distribution of random graphs. Let us suppose that a vertex is connected to k other vertices. Now choosing k vertices to connect to from $N-1$ other vertices we have k connections are made with probability p while the rest $N-1-k$ remain unconnected with probability $(1-p)$. Now the number of possible combinations to choose k connections from $N-1$ vertices is $\binom{N-1}{k}$. Therefore the probability distribution obtained can be given by the following equation.

$$\text{Prob(vertex } v \text{ has degree } k) = \binom{N-1}{k} p^k (1-p)^{N-1-k}.$$

The degree distribution for traditional random graphs result in a binomial distribution such that the vertices with degree k are uniformly distributed on either side of the mean degree. Unfortunately this fails to meet the criterion of power law graphs

to which webgraphs belong, which are scale free and tend to decay exponentially as k rises. Besides, note that traditional random graphs are static in nature. In other words, the number of vertices in the graph is fixed to begin with and therefore do not account for the evolving nature of the web. Random graphs created from random pairings are undirected in nature and do not incorporate any of the topologies usually seen in a webgraph. The small world effect is, however, observed in random graphs. The vertices in the random graph model are also separated by a very short distance between them.

4.2 The Preferential Attachment Model

Since the Random Graph Models failed to meet most of the desirable properties of a complex network like the webgraph, a new model was proposed by Barabasi and Albert [11] in 1999. With a focus on the dynamic nature of the webgraph, Barabasi and Albert tried to design a model that took the dynamics of such networks into account. **The Preferential Attachment Model** is an evolving network model of the webgraph that includes both the growth of the webgraph and the mechanism by which new links are created between existing web pages and new ones. The number of this vertices of this model grows with time and so does the number of edges. It's far more realistic because it takes into account the nature of the WWW and the manner in which a user links his site to other websites available on the network. It is quite likely for a new site to be linked to popular sites or corporate sites that are already well connected, that have high indegree. The probability $\Pi(k_i)$ of a vertex i having an indegree k_i to get a new edge is proportional to the k_i of the vertex as shown by the equation below.

$$\Pi(k_i) = \frac{k_i}{\sum_j k_j}$$

The sum of the indegree of all the vertices in the denominator provide the normalizing factor. Thus higher the indegree k_i of the vertex, the higher is the likelihood of the vertex to acquire a new edge. This bias of connecting to vertices with higher indegree is sometimes referred to as the preferential attachment or the "the rich gets richer". We notice in the equation above that a new node x will link to an existing node y is directly proportional to the indegree i of the vertex y. Similar examples

also apply to other complex networks like the internet model, the citation network where an author is most likely to cite a paper with many references or even the biological networks. This model meets two main features of the webgraphs - the power law degree distribution and the properties of small world networks. It can be shown that the indegree distribution of the vertices for such models follow the power law thus depicting their scale free nature . The average path length between any pair of vertices for this model is found to be quite small. Although, this model meets the power degree distribution and the small world phenomenon, it fails to provide us with a directed graph that has cycles or cliques present in webgraphs.

4.3 Copying Models

In a quest for a model that incorporates all the properties of webgraph, Kumar et al. [26] came up with the **copy model**. Before creating a web page, an author is quite likely to look for existing pages similar to his theme. Of the existing pages some websites tend to carry more information and are therefore linked to by many other webpages with similar topic. These webpages usually have a lot of hyperlinks to other pages. Many of these hyperlinks to which the popular webpage has linked are of interest and might be copied by the creator. The author will also include pages of his own choice as well in addition to the hyperlinks copied from popular pages. Therefore we see a fraction of the hyperlinks is usually copied from webpages that bear resemblance in content.

Dynamic in nature, the copying model allows an increase in the number of vertices and edges over time. Indegree distributions for this model follow the power law, usually seen in web graphs. Another feature of the web graph which was absent in earlier models discussed is presence of bipartite cliques. The copying model happens to generate a large number of cliques. In addition, the small world phenomenon is also observed in such models. There are two types of copying models – the linear growth copying model and the exponential growth copying model.

4.3.1 The Linear Growth Copying Model

The **linear growth copying model** [25] is an evolving model of the web graph in which one vertex is added at every timestep to the existing graph V_t. An outdegree

of d is kept constant. It is assumed that each new vertex can link to all the vertices in the web graph. In other words, a vertex added at time step $t+1$ can link to the vertices available from time step t. A constant out degree of d is maintained and a copy factor $\alpha \in (0,1)$ is used to choose the proportion of the links that will be copied from the chosen vertex to the new vertex.

4.3.2 The Exponential Copying Model

The exponential growth copying model was studied by Kumar et al in 2000 [26]. It is also an evolving model but unlike its linear version, the exponential growth copying model allows self loops and the growth in the number of vertices at each time step is a proportion of the number of vertices already existing in the webgraph. As new vertices are added the creation of new edges follows a particular criteria. Since multiple vertices are being added at every time step, edges can be created between the new vertices and the existing vertices as well that between the vertices that are being added. Thus if a particular probability edges are created between the new vertices and the existing ones chosen at random with probability p then edges are also created between the new vertices chosen at random with probability $1-p$. In addition, a proportion of the vertices will also have self loops.

4.4 Multi Layer Model

Traditional random graph models fail to take into account the self similar nature of the web graphs. All the models discussed so far have represented the web graph in one layer, allowing each vertex to connect to all the vertices in the graphs. However, these models cannot explain the presence of dense subgraphs in the web graph. In a recent paper [16] Dill et al. showed that the Web Graph is a fractal with several regions or layers that are generated by independent processes. Each of these regions contain a cohesive collection of web pages that share the same theme. Each collection of such pages is called Thematically Unified Clusters (TUC). These Thematically Unified Clusters exhibit characteristics that are similar to the Web which explains the fractal nature of webgraphs. The clusters are connected to each other through a navigational backbone. In order to account for the self similar nature of the web graph Laura et al [28] came up with "The Multi Layer Model" in 2002. The multi

layer model is a hybrid of both the evolving network model as well as the copying model.

Chapter 5

Experiments and Results

The main objective of my experiments was to study the link structure of a collection of websites obtained from a crawl of .GOV sites done by University of Glasgow. The webgraph contained a total of 1,247,753 documents. It was studied both at the microscopic and macroscopic level. Some of the salient features of the webgraph, namely the power law distribution for indegrees, outdegrees, and the distribution of the size of the strongly connected components was examined. Experiments were performed to identify each of the main components of the web graph, namely the Giant Strongly Connected Component, IN, OUT, TENDRILS and TUBES and Disconnected Components. The structure of some of the components were studied and their indegree distribution and out-degree distribution over the vertices showed power laws. In this section, I will describe the datasets for my experiments, data structures used, algorithms and experiments performed on the graph, the methodology for my experiments and the results obtained. ***Datasets, Data Structures and Link Information***

For my experiments, I used datasets from a University of Glasgow crawl [1] done on .gov websites. The crawl was stopped after it reached one million documents specifically html files and text files. However, the crawl includes documents of other file types like images, postscript files, pdf files and other application files as well. The datasets were preprocessed to leave out pages that were duplicates. Duplicate pages are basically pages with the same content that have been identified by crawlers as different pages due to differences in their canonical form. For example, the crawler might identify http://www.abc.gov and www.abc.gov/index.html as different pages due to the difference in the html tag although they are basically the same page. Such pages show up as duplicates and need to be removed. These duplicates were removed and all links to the web page with different canonical forms were incorporated into one page. Another problem that arises with crawlers when there is a redirection. Where the crawler encountered a redirection from one webpage to another the final

$Id1$	$Id2$	$Forward\ Links$	$Backward\ links$
$G05-27-2210295$	$G13-41-0507968$	2	0
$G02-29-0959755$	$G02-83-2257767$	0	1
$G03-63-2538302$	$G22-94-1660421$	1	0
$G02-78-2945139$	$G05-54-2938760$	1	1
$G41-63-1569388$	$G41-71-0723473$	1	0
$G01-68-2524035$	$G15-35-2304333$	0	1
$G09-26-2455494$	$G40-48-2352462$	1	0
$G01-43-2031819$	$G08-98-1347644$	0	1
$G09-28-2646499$	$G41-16-1615589$	0	1
$G00-15-3335359$	$G36-88-2456389$	0	1

Table 5.1: A section of the Links resolved file with the linkage Information of the webgraph

destination was considered. In other words, any links that showed up in the dataset because the user was being redirected to a destination page were processed so as to incorporate it as a link to the final destination page. Each of the webpages in the webgraph were given an id to uniquely identify them. The link information of the web sites was stored in a file. A diagram of a portion of the file, containing the link information is shown below. The first two columns in the diagram below represent the vertices that are connected to each other, either in one direction or both. A forward link can be defined as the link from vertex id1 to vertex id2, while a backward link connects vertex id2 to id1. The third column represents the number of forward links,and the last column represents the number of backward links. Thus, there is a link from G14-15-3079981 to G14-21-2349280 in either direction, as shown in table 5.1. Multiple links between two vertices in a particular direction were found as can be observed from in the first row of the dataset sample. The 2 in the 3rd column represents double links from vertex G05-27-2210295 to G13-41-0507968 while there are no links connecting G13-41-0507968 to G05-27-2210295.

In order to store the link structure of the web graph, adjacency lists were used as the data structure. Adjacency lists are arrays of linked lists and the space complexity for this data structure is of the order of $O(M+N)$ compared to $O(N^2)$ for adjacency matrices. The sheer size of webgraphs makes it very difficult to be stored in main memory specially using adjacency matrices. Besides, it is much easier to find

neighbors of a vertex ($O(d)$) that belong to large graphs. These two properties make adjacency list an ideal data structure for the web graph.

The file for the link structure was scanned and each time a new vertex is found, a new node is created. The node information is stored along with its in degree and out degree values in a data structure that is gradually updated. Let's call this data structure the symbol table, for convenience. On every scan of the information on a link, the vertices are compared with those already present in the symbol table. Two vertices id1 and id2 are examined for every link and links are created either from id1 to id 2 or from id2 to id1 or both . If the vertex id1 has already been registered, and a forward link has to be created between id1 and id2, id2 is attached to the tail of the linked list for id1 and a new list is created for id2. For backward links between id1 and id2, id1 is attached to the tail of the linked list for id2. The file contained over 9747K lines or pairs of nodes. Since both the vertices were matched against those already in the symbol table to check if they were already registered it was extremely time consuming to check for 9 million pairs of nodes as we will see later. The number of disk accesses required to read the whole webgraph into memory was extremely expensive. This is mainly because of the size of the webgraph and a File I/O is also much more expensive than that of accessing any data in memory. Once the link information for the web graph has been read from the file, the next step is to identify the different components of the web graph. The graph contained 11,067,049 links and 1,213,307 pages.

Finding the Components of the Web Graph

Once we have stored the linkage information in the form of adjacency lists the Strongly Connected Component Algorithm (SCC) is first run on the the web graph. Tarjan's algorithm [9] for the Strongly Connected Component algorithm was implemented since it is more efficient than the other known SCC algorithms. The algorithm finds all the strongly connected components in the web graph in linear time, (O(N) for time complexity) but it is worth mentioning here that webgraphs of sizes over 11 million links like the one above are extremely difficult to process using conventional memory algorithms like the one above. The information on all the strongly connected components is stored. Among these components, one of them is expected to stand out in size and it forms the core or the GSCC of the web graph. In my experiment,

the size of the core or the Giant SCC was 898,500 vertices and the total number of strongly connected components obtained was 299,618 (it includes many components of size 1). 297,547 of the 299,618 components were found to be singletons which is roughly over 97% of the total number of components.

Thus, the core of the web graph comprises approximately 74% of the whole web graph which deviates significantly from results obtained by Kleinberg et al. [25] and Kumar et al [27]. An index of all the vertices belonging to GSCC are stored in an array. In the next step, Depth First Search algorithm was run from each of the vertices belonging to SCC. Running DFS from the vertices of the GSCC provides with the set of vertices that can be reached from GSCC. The vertices that can be reached from vertices in the core, but don't belong to the GSCC, are by definition vertices in OUT. Let X be the set of vertices such that it contains vertices belonging to the GSCC and OUT. The other major component IN comprises of vertices that can reach those in GSCC but cannot be reached by vertices of the core. Now since we have already identified the core we can easily finding the vertices of IN $\bigcup$ GSCC by reversing the links and find the set of vertices that are reachable from those in GSCC along the reverse links. Let Y be the set of vertices such that it comprises of vertices reachable from GSCC along the reverse links and are found by running Depth First Search along the reverse links.

$$X = \{v \in V(\text{OUT}) \bigcup V(\text{GSCC}) : \exists \text{ a directed path from any vertex in GSCC to } v\}$$

$$Y = \{v \in V(\text{IN}) \bigcup V(\text{GSCC}) : \exists \text{ a directed path from any vertex } v \text{ to those in GSCC}\}$$

At this point, having found a set of vertices X and Y the components IN can found leaving out GSCC from Y ($\mathbf{V(IN) = V(Y) - V(GSCC)}$) while OUT can be found by leaving out vertices belonging to GSCC from the set X ($\mathbf{V(OUT) = V(X) - V(GSCC)}$). Thus we have identified the components IN and OUT and have the full bowtie B at this point.

The next step is to identify the set of vertices belonging to the Tendrils. Now Tendrils would consist of IN_Tendrils and the OUT_Tendrils. IN_Tendrils are vertices reachable from IN but not viceversa. DFS, run on the vertices belonging to

IN along the forward links results in a set of vertices F ($\mathbf{F = IN} \bigcup$ IN_Tendrils $\bigcup \mathbf{GSCC} \bigcup \mathbf{OUT} \bigcup \mathbf{Tubes}$ that include IN_Tendrils, IN, GSCC, OUT and the Tubes. Let us call this set F. On the other hand, DFS run on the set of vertices in OUT along the reverse links would actually explode to give us the set of vertices J($\mathbf{J = OUT} \bigcup$ OUT_Tendrils $\bigcup \mathbf{GSCC} \bigcup \mathbf{IN} \bigcup \mathbf{Tubes}$) belonging to OUT_Tendrils, Tubes, OUT, GSCC and IN. Now the Bowtie B comprises of vertices belonging to IN, GSCC, and OUT we can write F and J in terms of B. The set of vertices F and J both have the B and Tubes in common. Thus the intersection of sets F and J would give us B and Tendrils. IF we leave out vertices belonging to the bowtie from the intersection of the sets F and J we would obtain the set of vertices belonging to Tubes. Now by excluding vertices that belong to the bowtie B and the Tubes from the set F gives us IN _Tendrils while by excluding vertices belonging to either the bowtie B and the Tubes from the set J results in OUT_Tendrils. Thus we have both portions of the Tendrils and the union of IN-Tendrils and the OUT_Tendrils would give us the component TENDRILS of the webgraph. Thus we have all the components of the webgraph, barring the DISCONNECTED Components.

$$
\begin{aligned}
F &= B \bigcup \text{Tubes} \bigcup \text{IN\_Tendrils} \\
J &= B \bigcup \text{Tubes} \bigcup \text{OUT\_Tendrils} \\
F \bigcap J &= \text{Tubes} \bigcup B \\
Tubes &= (F \bigcap J) - B \\
\text{IN\_Tendrils} &= F - B - \text{Tubes} \\
\text{OUT\_Tendrils} &= J - B - \text{Tubes}
\end{aligned}
$$

While running DFS from the vertices in the component IN the vertices v such that there is a edge $\{u, v\}$ where $u \in$ IN and $v \in$ GSCC are the **entrypoints** of the Giant Strongly Connected component. Similarly the **exitpoints** of the GSCC are identified by separating the vertices x such that there is an edge $\{x, y\}$ where $x \in$ GSCC and $y \in$ OUT. Thus by running DFS along the reverse links from OUT we identified all the exitpoints of the core, GSCC. Having found the components we will take a look at the different sizes of the various components, their degree distributions

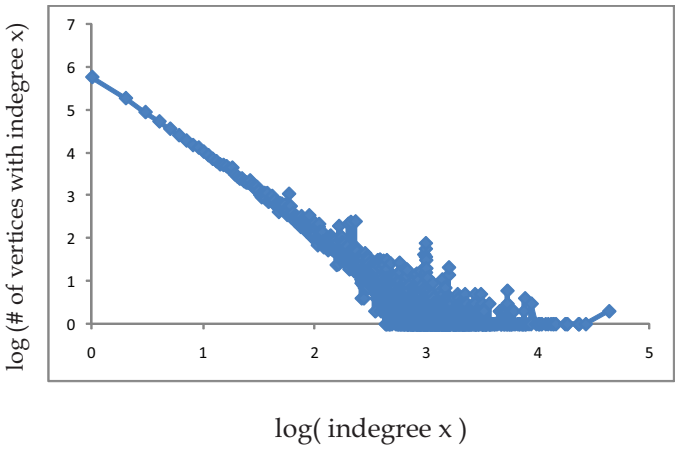


Figure 5.1: Indegree Distribution for the webgraph

and other structural properties. Before we go into that we will study the various degree distributions of the whole webgraph.

Power Laws on Degree Distributions and Size of the Components

The degree distribution of the web graph follows power laws for both the indegree and the outdegree of the vertices in the web graph. The frequency distribution for the indegree and the out degree distribution of the graphs were computed. The maximum indegree of a vertex in the web graph has been found to be $44,347$ and the maximum outdegree of the vertex in the web graph was 653. The number of nodes having an indegree x is calculated for the vertices with indegree x and a log-log plot is done. It clearly shows a heavy-tailed distribution and verifies the power laws for the in-degree distribution. The slope of the first graph gives the value of the exponent for the in-degree distribution. The slope is calculated by using a regression model that best fits the graph. From my data, the exponent for the in-degree was found to be 2.0281. The exponent of the in-degree distribution however deviated slightly from that of 2.1, found by both Barabasi and Albert et al [12] and Kumar et al [27]. in their experiments.

Similarly the frequency distribution for the out-degree of all the vertices in the web-graph is computed and a log- log plot is done for the number of pages with out-degree x against the out degree of the page. The graph for the out degree distribution shows a deviation from the power laws for low values of x but as the values of x

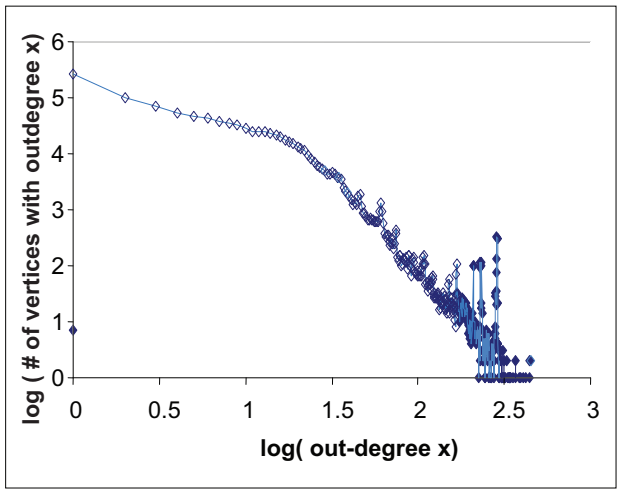


Figure 5.2: Outdegree Distribution of the Webgraph

gradually increases the distribution exhibits power laws. The exponent for the out-degree distribution is calculated in a similar manner and it is found to be 2.68194. It is worth mentioning here that the exponent calculated by Albert and Barabasi was 2.71.

The distribution of the strongly connected components was also observed with respect to their sizes. A double logarithmic plot was done for the scc distribution. However, is should be mentioned here that the largest component (898500)vertices and the smallest component (1) were left out in order to fit the distribution into a power law.The exponent for the scc distribution was found to be **2.145**

The component OUT had $155,103$ vertices which is roughly about 12.78% of the whole webgraph studied. The indegree distributions and the outdegree distributions of the vertices within OUT were studied to verify their scale free properties within a component. The indegree distribution as seen in Figure 5.4 clearly exhibit power law distributions. Although the outdegree distribution of OUT shows power law distributions for vertices with a higher outdegree, it deviates from the powerlaw distribution near the tail of the graph. The heavy tailed distribution is missing.

The results were obtained by running the jobs on nodes in the cluster at Math and Stat. The compute node used for the purpose of the experiments had a dual core processor with a speed of 3GHZ speed and a memory of 32GB RAM. Please check [5] for details on the configuration of the compute nodes.

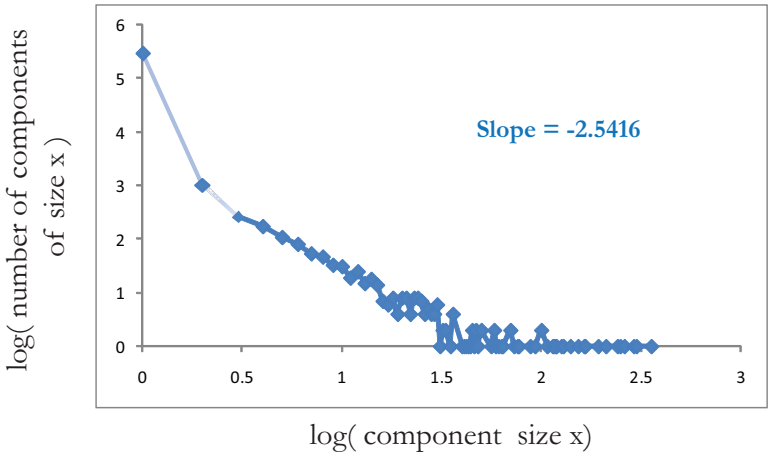


Figure 5.3: Strongly Connected Component Distribution

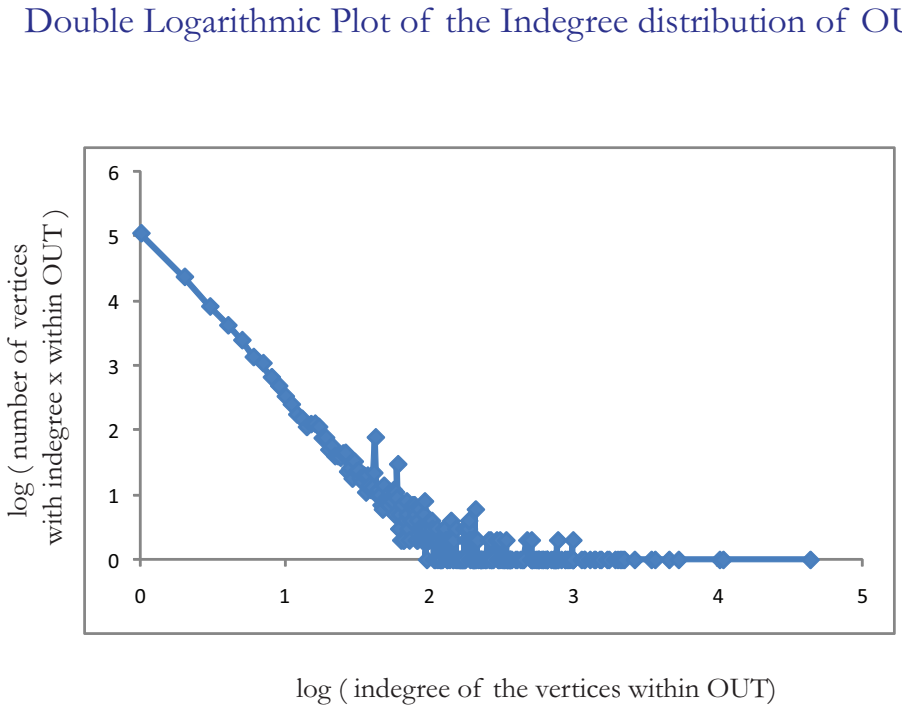


Figure 5.4: Indegree Distribution of the vertices in OUT component of the Webgraph

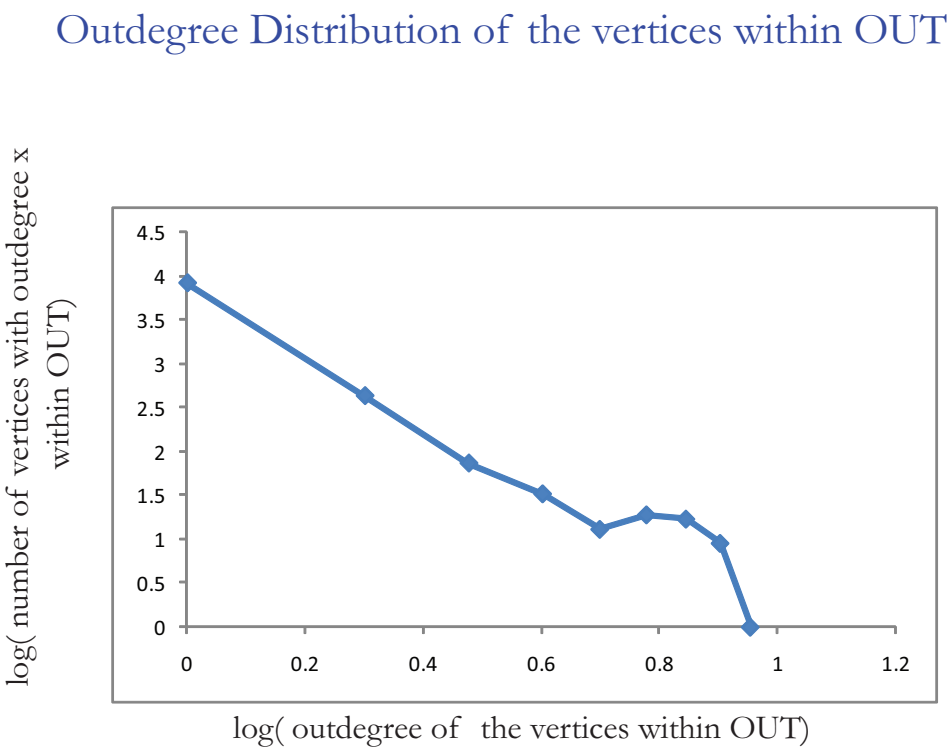


Figure 5.5: Outdegree Distribution of the vertices in the OUT component

We have looked at a dataset to closely examine and compare it to a webgraph. We wanted to verify if a smaller dataset obtained from a random crawl would exhibit scale free characteristics seen in webgraphs. The scale free characteristics namely power laws with respect to indegree distribution, outdegree distribution and the strongly connected component distribution with respect to size were consistent with results showed by larger graphs like the Altavista crawl on 200 million nodes done by Broder [14]. Close examination of the link structure of the dataset revealed a giant component that comprised of about **74.21**% of the whole graph. Thus, most of the nodes in the graph were concentrated in the core making the graph highly connected. The IN and OUT were roughly only **12**% each of the whole graph. We wanted to see if there were bowties inside the components. Our studies revealed that unlike the findings with the whole webgraph there were hardly any significantly large strongly connected component inside the components. The largest scc in **OUT** consisted of only 3 vertices thus revealing a very fragmented and loosely held component. The indegree distribution, outdegree distribution and the scc distribution were also studied at the component level and it exhibited power laws.

For future studies, the depth of each of the components can be studied. We would like to recommend the study of the reliability of the webgraph. It would be very useful if we can find out how the connectivity of the webgraph would be affected if nodes were removed at random. It is well known that since webgraphs contain hubs residing in the core of the webgraph that are responsible for holding the whole webgraph together, removal of the major hubs would lead to the webgraph breaking up. This can be examined by removing vertices with the highest outdegree in the webgraph. It is quite plausible that the same should hold true for the dataset although it exhibited a higher connectivity due to the densely connected giant core that made up a significant portion of the webgraph.

Bibliography

[1] The .gov test collection. `http://ir.dcs.gla.ac.uk/test_collections/govinfo.html`, November 2002.

[2] Dmoz, open directory project. `http://dmoz.org/`, March 2008.

[3] Netcraft, November 2008. `http://netcraft.com`.

[4] Touchgraph, November 2008. `http://touchgraph.com`.

[5] Mathstat cluster, March 2009. `http://www.mathstat.dal.ca/cluster/index.php/Site/Hardware/`.

[6] World wide web size, March 2009. `http://www.worldwidewebsize.com/`.

[7] Lada A. Adamic. Zipf, power-law, pareto - a ranking tutorial.

[8] Lada A. Adamic. The small world web. pages 443–452. Springer, 1999.

[9] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley Publishing Company, Reading, MA, 1974.

[10] Ricardo Baeza-Yates and Carlos Castillo. Link analysis in national Web domains. In *Workshop on Open Source Web Information Retrieval (OSWIR)*, 2005.

[11] A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 1999.

[12] Albert-Laszlo Barabasi, Reka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281, 2000.

[13] Luca Becchetti, Carlos Castillo, Debora Donato, and Adriano Fazzone. A comparison of sampling techniques for web characterization. In *Workshop on Link Analysis (LinkKDD)*, 2006.

[14] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks*, 2000.

[15] Thomas H. Cormen, Charles E Leiserson, and Ronald. L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, MA, 1990.

[16] Stephen Dill, S. Ravi Kumar, Kevin S. McCurley, Sridhar Rajagopalan, D. Sivakumar, and Andrew Tomkins. Self-similarity in the web. In *The VLDB Journal*, pages 69–78, 2001.

[17] Chris H. Q. Ding, Xiaofeng He, and Hongyuan Zha. A spectral method to separate disconnected and nearly-disconnected web graph components. In *KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 275–280, New York, NY, USA, 2001. ACM.

[18] D. Donato, L. Laura, S. Leonardi, and S. Millozzi. Large scale properties of the webgraph. *European Physical Journal B*, 2004.

[19] Debora Donato, Stefano Leonardi, Stefano Millozzi, and Panayiotis Tsaparas. Mining the inner structure of the web graph. 41, May 2008.

[20] S. N. Dorogovtsev, J. F. F. Mendes, and A. N. Samukhin. Giant strongly connected component of directed networks. *Physical Review E*, 64:025101, 2001.

[21] P. Erd

os and A. Réyi. On random graphs. *Science*, 6(5439):290–298, October 1959.

[22] A. Gulli and A. Signorini. The indexable web is more than 11.5 billion pages. In *WWW '05: Special interest tracks and posters of the 14th international conference on World Wide Web*, New York, NY, USA, 2005. ACM Press.

[23] Bernardo A. Huberman and Lada A. Adamic. Evolutionary dynamics of the world wide web, 1999.

[24] Jon Kleinberg. The small-world phenomenon: An algorithmic perspective. In *in Proceedings of the 32nd ACM Symposium on Theory of Computing*, pages 163–170, 2000.

[25] Jon M. Kleinberg, Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew S. Tomkins. The web as a graph: Measurements, models, and methods. 1999.

[26] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. Stochastic models for the web graph. *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, 2000.

[27] Ravi Kumar, Prabhakar Raghavan, Sridhar Rajagopalan, and Andrew Tomkins. Trawling the web for emerging cyber-communities. *Comput. Netw.*, 1999.

[28] L. Laura, S. Leonardi, G. Caldarelli, P. De, and Los Rios. A multi-layer model for the webgraph. 2002.

[29] John Markoff. Debating the size of the web. August 2005. `http://www.iht.com/articles/2005/08/15/business/web.php`.

[30] Stanley Milgram. The small world problem. *Psychology Today*, 2:60–67, 1967.

[31] M. Ángeles Serrano, Ana Maguitman, Marián Bogu ná, Santo Fortunato, and Alessandro Vespignani. Decoding the structure of the www: A comparative analysis of web crawls. *ACM Trans. Web*, 1(2):10, 2007.

[32] Ian Soboroff. Do trec collections look like the web? *SIGIR Forum*, 36(2):23–31, 2002.

[33] Mike Thelwall and David Wilkinson. Graph structure in three national academic webs: power laws with anomalies. *J. Am. Soc. Inf. Sci. Technol.*, 54(8):706–712, 2003.

[34] the free encyclopedia Wikipedia. Pareto principle, March 2009. `http://en.wikipedia.org/wiki/Pareto_principle`.