Modeling and Thematic Analysis of Neighborhood Structures in the Web and Hierarchical Identification of Web Communities

by

© Isheeta Nargis

A thesis submitted to the
School of Graduate Studies
in partial fulfilment of the
requirements for the degree of
M.Sc.

Department of Computer Science
Memorial University of Newfoundland

July 2007

St. John's Newfoundland and Labrador

Abstract

The web graph represents the structure of the World Wide Web by denoting each web page as a vertex and each hyperlink as an arc. The motivating goal behind the research constituting this thesis is twofold – firstly, to model the local structure of the web graph, and secondly, to discover communities of related web pages.

We study the concept of the neighborhood graph to model the local structure surrounding a particular web page. We analyze some structural and statistical properties of the neighborhood graphs and perform a comparison with the corresponding properties of the whole web graph. In several aspects these neighborhood graphs show a similar characteristic to the entire web graph. Both the indegree and outdegree distribution of a sufficiently large neighborhood graph follow the power law phenomenon. We perform a thematic analysis of the local structure of the Web by discovering authorities and hubs in neighborhood graphs, where the set of authorities and hubs gives a representative flavor on the theme of the page in question. We also analyze temporal evolution of Hyperlinked communities and identify Core communities in neighborhood graphs.

We devise an algorithm to extract communities solely based on the topology of the Web. Central to our approach is the innovative idea of Iterative Cycle Contraction to discover Web communities comprised of related web pages. The intuition behind this algorithm is that if two pages link to each other then they are thematically related. Successive iterations yield a hierarchical structure of communities and allow us to define a similarity measure between two web pages in the same community by noting at which iteration their corresponding vertices are first grouped into a single vertex. We apply the algorithm to some focused subgraphs of the web graph and evaluate its effectiveness by performing an investigation into the theme(s) of the

putative communities that it finds. We find that the algorithm is successful at identifying communities and distinguishing communities with varying thematic content in neighborhood graphs. An examination of the distribution of community sizes in a particular iteration of the algorithm reveals that for sufficiently large neighborhood graphs a power law is observed.

Acknowledgments

First of all, I would like to express my immense gratitude to my supervisor, Dr. David A. Pike, without whose continuous support and guidance this work could not reach any destination. I particularly appreciate the amount of time and effort that he spent for guiding me. His support was not only limited to research works; he gave me advices on other aspects of academic life as well. He sent me to AARMS Summer School to have a solid background in my research area. Whenever I faced any problem in the preparation of materials in latex, he provided me a solution. We have gone through several corrections on my writing, he corrected many grammatical mistakes and more importantly, many instances where I could not express in writing what I was trying to convey. He commented to fill the gaps in my writing. He helped me in all ways to improve the skills in scientific writing.

I am grateful to my co-supervisor, Dr. Wolfgang Banzhaf, who gave me many suggestions to improve my thesis and encouraged me by his inspiring comments.

I would like to express my gratitude to the examiners Dr. Anthony Bonato and Dr. Manrique Mata-Montero for the feedback and suggestions they provided me.

I would like to express my thanks to Neil A. McKay, who worked on the construction of the Neighborhood Graphs in Summer 2005 with David. At the very start of my work he explained his works to me. I adapted his programs to suit my specific needs for the research. He also gave valuable feedbacks on the draft of one of my papers of which he is a co-author, and I have incorporated some of those feedbacks to my thesis as well.

I feel myself really very lucky since I had the chance to learn about various mathematical models of the web graph when I took a course named Massive Networks and Internet Mathematics in the AARMS Summer School 2006 held in Dalhousie Uni-

versity, Halifax, Canada. I would like to express gratitude to the course teacher Dr. Anthony Bonato. I learnt about recent developments in this area when I attended the MoMiNIS Winter School and the Fourth WAW held in Banff, Canada.

I would like to thank Dr. Jeannette Janssen for providing me feedback and suggestions in my early stage of research while I was in Halifax.

This research has been supported by funding from MITACS (Mathematics of Information Technology and Complex Systems) and NSERC (Natural Sciences and Engineering Research Council of Canada). I would like to thank MITACS and NSERC for providing us the grants.

I have learnt many things about the academic life while in conversation with my friends Momotaz and Rajibul. In many stages whenever I felt frustrated they gave me the best suggestions. It has been a very pleasant time with them in St. John's.

I would like to express my gratitude to my family members. My husband Tuhin has been always there for me, whenever I need any type of support. Being a student in the same discipline, he also gave me good feedback on my work. At times of frustrations he is the one to inspire me and cheer me up; at the time of achievements he is the one who seems happier than me. My sisters and my brother gave me good suggestions in all stages of my life. My nephews and nieces are the best gifts in this world, I find immense pleasure passing my time with them. My parents are the ones for whom I have reached where I am now. My mother took care of my studies in my childhood. My father has been the inspiration of my life. I always wanted to be as caring as my mother and as scholar as my father. I always miss them here in Canada since they live in my country, Bangladesh. But whenever I feel obstacles or darkness I think about my parents to get a light of hope.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The Structure of the Web

The structure of the World Wide Web can be represented by a large directed graph in which each vertex corresponds to a web page, and in which there is an arc from one vertex to another if there is a hyperlink from the first web page to the second one. This graph, known as the web graph and denoted throughout this thesis by $\mathcal{W}$, discards all the content and contains only the link information of the Web.

By removing the content information the size of the data structure describing the whole World Wide Web reduces to a great extent. The web graph being itself a massive data structure makes the task of storing and manipulating the whole graph infeasible, so we take a focused approach. Beginning with a specified web page, we determine which other pages are in close proximity to it, and then we construct the subgraph of the web graph that is induced by these pages.

1.2 Motivation

The motivating goal behind the research constituting this thesis is twofold, as described below:

1. To model the local structure of the web graph. We want to analyze some structural and statistical properties of the local structure of the Web and then compare them with the corresponding properties of the entire Web. We also want to perform a thematic analysis of the local structure of the Web.

2. To discover communities of related web pages. Pages with related theme(s) may be identified as a Web community, and a hierarchical structure of Web communities can be defined. We want to detect Web communities using only the structural information of the Web.

1.3 Contributions of this Thesis

The major contributions of this thesis can be summarized as follows:

- We introduce the concept of the neighborhood graph to model the local structure of the World Wide Web, where this local structure surrounds a particular web page.

- We analyze some structural and statistical properties of the neighborhood graphs starting with some web pages and compare to the corresponding properties of the whole web graph, as described below:

  - We analyze the relative size of regions in the Bow-Tie structure.

  - We study the indegree and outdegree distributions.

- We analyze the hyperlinked community consisting of the best Authorities and hubs in the surrounding region of a particular page in the following ways:
  - We investigate the themes of the best hubs and Authorities and compare to the theme of the page in question.
  - We analyze the temporal evolution of hyperlinked communities in the neighborhood graphs.
  - We extract the core hyperlinked communities in the neighborhood graphs.
- We devise an innovative algorithm named the Iterative Cycle Contraction algorithm to extract a hierarchy of Web communities. We define a similarity measure between two web pages in the same community based on the iteration at which their corresponding vertices are first grouped into a single vertex. We perform the following experiments on the Iterative Cycle Contraction algorithm:
  - We investigate the theme(s) of the pages in the communities extracted in some iterations on the same graph.
  - We analyze the similarity measure between web pages.
  - We study the distribution of the size of communities in a particular iteration of the algorithm.

1.4 Organization of this Thesis

The rest of this thesis is organized as follows:

- Chapter 2 presents the basic background and related works. The chapter describes the definition, applications, structural and statistical properties of the

web graph, followed by reviews on the HITS algorithm and the Trawling algorithm. This chapter ends with some definitions and notations in Graph Theory that will be used throughout the rest of the thesis.

- Chapter 3 presents the definition, construction and some structural and statistical properties of the neighborhood graphs. In particular, the relative size of regions in the Bow-Tie structure and the distributions of indegree and outdegree in the neighborhood graphs for some web pages are presented along with a comparison to the corresponding properties in the whole web graph.

- Chapter 4 describes the concepts of discovering Authorities and Hubs in neighborhood graphs, followed by the temporal evolution of hyperlinked communities in neighborhood graphs and the identification of core communities in the neighborhood graphs.

- Chapter 5 presents a new algorithm, Iterative Cycle Contraction, that extracts a hierarchy of Web communities. This chapter starts with the idea of the algorithm, which is followed by the algorithm. Then experimental results obtained from an implementation of the algorithm are presented. Some example communities in some initial iterations are presented to illustrate the hierarchical structure of communities, followed by a discussion on the similarity measures between web pages and the distribution of the size of communities in a single iteration.

- Chapter 6 summarizes the thesis and presents some possible future research directions.

  The main results in Chapter 3 and Chapter 4 have been submitted for publication [Nargis et al., 2007]. Also, the main results in Chapter 5 have been

submitted for publication [Nargis and Pike, 2007].

Chapter 2

Background and Related Works

This chapter is dedicated to basic background and related works. We first review the definition of the web graph followed by various applications of the knowledge on the web graph. Then the macroscopic structure of the web graph is described and an overview of the power law phenomenon in the World Wide Web is presented. We describe the concepts of Authorities and Hubs on a search topic and an algorithm to extract them. Then a short review of the extraction algorithm of emerging cyber-communities is presented. The chapter ends with some definitions and notations in Graph Theory that will be used throughout the rest of the thesis.

2.1 The Web Graph

2.1.1 Definition

To model the link structure of the World Wide Web we use the idea of the web graph. The web graph $\mathcal{W}$ is defined as the directed graph where each vertex $v \in V(\mathcal{W})$ represents a web page, where $V(\mathcal{W})$ is the vertex set of $\mathcal{W}$, and the arc set $E(\mathcal{W})$ of $\mathcal{W}$ is defined as $E(\mathcal{W}) = \{(u, v) \ : \ u \text{ has a hyperlink to } v\}$. So each web page is

denoted by a vertex and each hyperlink by an arc in $\mathcal{W}$, respectively.

2.1.2 Applications

There are various applications that may be enhanced by developing the knowledge of the properties of the web graph [Broder et al., 2000]:

1. Designing Web crawl strategies [Cho and Garcia-Molina, 2000, Liu et al., 2004, Pant et al., 2004, Toyoda and Kitsuregawa, 2006].

2. Analysis of the behavior of Web algorithms that use link information in the Web [Botafogo and Shneiderman, 1991, Mendelzon and Wood, 1995, Carrire and Kazman, 1997, Brin and Page, 1998, Kleinberg, 1999, Osthus and Buckley, 2004].

3. Predicting the evolution of Web structures such as bipartite cores and webrings, and designing better algorithms for discovering and organizing them [Kumar et al., 1999].

4. Predicting the emergence of new, yet unexploited phenomena in the web graph [Kumar et al., 1999].

5. Mathematical modeling of the evolution of the World Wide Web [Albert et al., 1999, Kleinberg et al., 1999, Kumar et al., 2000, Fabrikant et al., 2002, Cooper and Frieze, 2003, Bonato, 2004, Chung and Lu, 2004, Cooper et al., 2004, Osthus and Buckley, 2004, Sydow, 2004, Leskovec et al., 2005, Chakrabarti and Faloutsos, 2006].

6. Design and enhancements of Web Search Engines [Brin and Page, 1998, Kleinberg, 1999, Ke et al., 2006].

7. Discovery of Web communities [Kumar et al., 1999, Gibson et al., 2005]

2.2 The Bow-Tie Structure of the Web

Broder et al. [Broder et al., 2000] experimented on a Web crawl of approximately 200 million pages and 1.5 billion hyperlinks and came to the conclusion that the macroscopic structure of the Web has a bow-tie shape, as illustrated in Figure 2.1, composed of five main regions as follows:

SCC The largest strongly connected component, in which any page can be reached from any other page by following hyperlinks.

IN Pages that can reach the SCC, but cannot be reached from it - possibly new sites that people have not yet discovered and linked to.

OUT Pages that are accessible from the SCC, but do not link back to it, such as corporate websites that contain only internal links.

TUBES and TENDRILS TENDRILS is the region containing pages that are reachable from portions of IN but not reachable from any page in the SCC and pages that can reach portions of OUT but cannot reach any page in the SCC. TUBES form a passage from a portion of IN to a portion of OUT without involving SCC.

DISC all other Disconnected Components.

2.3 Power Law

The *Power Law Phenomenon* is described by the relation p_k varies in relation to $k^{-\beta}$, where p_k is the probability that the variable in question has value k and $\beta > 1$ is a constant. Networks with some parameters following such a distribution are called *scale-free* since this distribution has no natural scale.

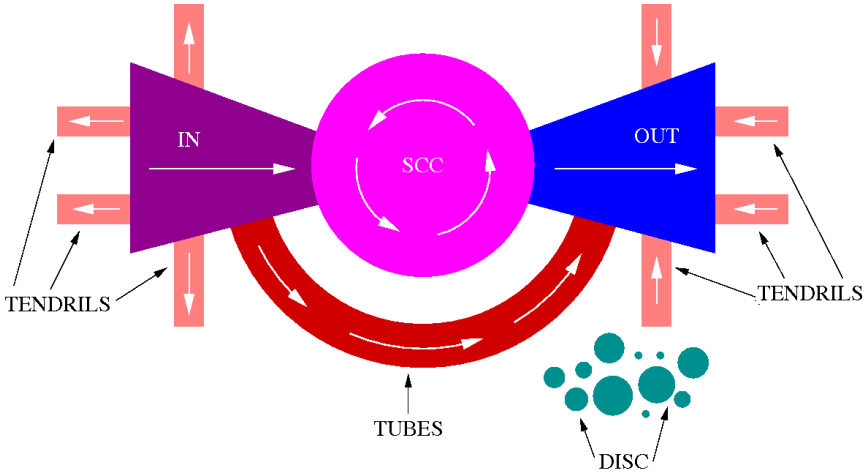

Figure 2.1: The macroscopic structure of the World Wide Web [Broder et al., 2000].

Many real networks have properties that follow the power law. Redner [Redner, 1998] studied the distribution of the citation of papers in two databases and reached the conclusion that the asymptotic tails of both of the distributions follow a power law. In a study of the Internet topology, the power law distribution was observed to explain the degree distribution in two different levels, namely in the Autonomous System level, and the Router level [Faloutsos et al., 1999].

The indegree distribution of vertices in the web graph has been observed to follow a power law. In both [Broder et al., 2000] and [Donato et al., 2004], the power law exponent β for the indegree distribution in $\mathcal{W}$ was observed to be 2.1, which is also consistent with previous results reported in [Barabási and Albert, 1999, Kumar et al., 1999]. About the outdegree distribution, several researchers have come to the conclusion that it also follows a power law, albeit with a significant deviation in the initial portion [Barabási and Albert, 1999, Kumar et al., 1999, Broder et al., 2000]; in [Broder et al., 2000] this power law exponent was observed to be 2.72. However, in a more recent experiment Donato et al. [Donato et al., 2004] raised doubt about the existence of a power

law in the outdegree distribution.

2.4 Self-similarity of the Web

Dill et al. [Dill et al., 2002] experimented on various subgraphs of the web graph induced by collections of web pages that share a common attribute (for example, domain, keyword, geographic location). They reached the conclusion that self-similarity exists in the Web for several parameters, where the self-similarity holds in the sense that these subgraphs show similar characteristics to the whole web graph. They analyzed the bow-tie region structure as well as the indegree and outdegree distributions in their subgraphs. The indegree and outdegree distributions of these subgraphs follow the power law phenomenon. In each of these subgraphs there exists a bow-tie structure with a large SCC.

2.5 Authorities and Hubs

2.5.1 Overview

Kleinberg [Kleinberg, 1999] devised the HITS (Hyperlink Induced Topic Search) algorithm to identify the best Hubs and Authorities for a particular search topic. The idea of 'Authority' comes from the intuition that a page linking to another page implies that the creator of the first page infers a notion of 'Authority' to the second page. Gibson et al. [Gibson et al., 1998] define a 'hyperlinked community' as a community on a topic that consists of two interrelated sets of pages. The first set contains the 'Authorities' on that topic which are highly referenced pages and the second set consists of 'Hub' pages that link to many Authorities on the topic.

For a particular search query q, HITS operates on a focused subgraph $\mathcal{W}_q$ of $\mathcal{W}$

constructed from the search results on query string q. HITS returns the topmost c Authorities and topmost c Hubs on the topic query q, where c is a fixed parameter.

2.5.2 The Graph on which HITS operates

HITS aims to focus the computational effort on pages relevant to q. As a first try, the set P_q of all pages containing q might be considered; but P_q has the following problems [Kleinberg, 1999]:

1. P_q may contain a huge number of pages; as a result high computational cost will be incurred to find the best Authorities amongst them.

2. Some or most of the best Authorities related to q may not be included in P_q.

Therefore, HITS attempts to seek a collection S_q of pages with the following criteria [Kleinberg, 1999]:

1. S_q is relatively small in size so that the associated computational cost is minimized.

2. S_q is rich in relevant pages to q, implying that finding good Authorities in S_q will not be a very difficult task. The intuition behind this implication is that the good Authorities are likely to be frequently referenced within S_q.

3. S_q contains most (or many) of the best Authorities.

As a candidate for S_q, the *Root Set* R_q may be considered, where R_q is defined as the set of top t ranked web pages for the search query q from any conventional text-based search engine; here t is a parameter, typically set to 200. R_q satisfies criteria 1 and 2, but generally is far from satisfying criterion 3. The set R_q is limited to top t ranked

pages resulting in a small set of pages S_q. Since $R_q \subset P_q$ and P_q often fails to satisfy criterion 3 it follows that R_q may not satisfy criterion 3 in most cases.

Though R_q does not satisfy criterion 3, still R_q can be used to produce a candidate set satisfying the criteria for S_q. A good Authority on q may not be present in R_q but it is quite likely that it has a link from at least one page in R_q. If R_q is expanded by incorporating the pages that link to or have links from any page in R_q, then the resulting set B_q contains much more good Authorities than R_q, which implies that B_q may satisfy criterion 3. This set B_q is called the *Base Set* for q. To satisfy criterion 1, HITS maintains a limit of d backward links per web page in B_q, where d is a fixed parameter, typically set to 50. That is, if the number of web pages that link to a particular web page in the root set is greater than d, then HITS randomly selects d of them to be included in B_q. Kleinberg [Kleinberg, 1999] reported that B_q typically satisfies each of the criteria 1, 2, and 3 for the experimented value of $t = 200, d = 50$ using the search engine Altavista. Therefore B_q is a very good candidate for S_q.

Let $\mathcal{W}[B_q]$ denote the subgraph of $\mathcal{W}$ induced by the base set B_q. Kleinberg [Kleinberg, 1999] applied a heuristic to reduce the effect of links serving purely navigational function on the extraction of best Authorities and Hubs. He defined two types of links in $\mathcal{W}[B_q]$:

Transverse Link A link between two pages with different domain names.

Intrinsic Link A link between two pages with the same domain name.

Since intrinsic links are very often created for the navigational purpose within the infrastructure of a site, they convey much less information on inferring Authority by a link than transverse links. The heuristic is to delete all intrinsic links from $\mathcal{W}[B_q]$, resulting in the focused subgraph of $\mathcal{W}$ on q, denoted by $\mathcal{W}_q$.

In summary,

1. R_q is the set of top t ranked web pages for the search query q.

2. $B_q = \{u \in V(\mathcal{W}) \ : \ v \in R_q$ and $((u,v) \in E(\mathcal{W})$ or $(v,u) \in E(\mathcal{W}))\}$. It is noteworthy that a limit of d backlinks per page is maintained, that is, if a page u has more than d backlinks then d pages, randomly chosen from the set of pages that link to u, are included in the set B_q.

3. $\mathcal{W}[B_q]$ is the subgraph of $\mathcal{W}$ induced by the base set B_q, that is, $V(\mathcal{W}[B_q]) = B_q$ and $E(\mathcal{W}[B_q]) = \{(x,y) \in E(\mathcal{W}) \ : \ x \in B_q$ and $y \in B_q\}$.

4. $\mathcal{W}_q$ is the subgraph of $\mathcal{W}[B_q]$ defined as follows:
   $V(\mathcal{W}_q) = V(\mathcal{W}[B_q])$ and
   $E(\mathcal{W}_q) = \{(x,y) \in E(\mathcal{W}[B_q]) : x$ and y are pages under different domains$\}$.

2.5.3 The HITS Algorithm

Hubs and Authorities show a mutually reinforcing relationship – a good Authority is a web page that has links from many good Hubs, and a good Hub is a web page that links to many good Authorities. For a given web page v, HITS maintains two weights — Authority weight a_v and Hub weight h_v. Let $\vec{a} = (a_{v_1}, \ldots, a_{v_n})^T$ (respectively $\vec{h} = (h_{v_1}, \ldots, h_{v_n})^T$) be the vector of the Authority (respectively Hub) weights for the vertices $v_1, \ldots, v_n$, where $n = |V(\mathcal{W}_q)|$ is the number of web pages in the graph $\mathcal{W}_q$ and the notation X^T is used to denote the transpose of a vector X. HITS performs the following steps iteratively until the vectors $\vec{a}$ and $\vec{h}$ converge:

1. For each $v \in V(\mathcal{W}_q)$, compute $a_v = \sum_{u \, : \, (u,v) \in E(\mathcal{W}_q)} h_u$.

2. For each $v \in V(\mathcal{W}_q)$, compute $h_v = \sum_{u \, : \, (v,u) \in E(\mathcal{W}_q)} a_u$.

3. Normalize a_v and h_v so that $\sum_{v \in V(\mathcal{W}_q)} a_v = \sum_{v \in V(\mathcal{W}_q)} h_v = 1$. Obtaining the normalized values is done by multiplying the vectors $\vec{a}$ and $\vec{v}$ from the previous two steps by the constants $\left(\sum_{v \in V(\mathcal{W}_q)} a_v \right)^{-1}$ and $\left(\sum_{v \in V(\mathcal{W}_q)} h_v \right)^{-1}$ respectively.

2.5.4 Convergence of HITS

Let A be the adjacency matrix representation of the graph $\mathcal{W}_q$, that is, $A_{i,j} = 1$ if $(v_i, v_j) \in E(\mathcal{W}_q)$, and $A_{i,j} = 0$ otherwise. Then we can write the update rule for Authority weights as $\vec{a} \leftarrow A^T \vec{h}$ and the update rule for Hub weights can be written as $\vec{h} \leftarrow A\vec{a}$. Continuing replacement of $\vec{a}$ and $\vec{h}$ we get

$$\vec{a} \leftarrow A^T \vec{h} \leftarrow A^T A \vec{a} = (A^T A)\vec{a}$$

and

$$\vec{h} \leftarrow A\vec{a} \leftarrow AA^T \vec{h} = (AA^T)\vec{h}.$$

For $\vec{a}$ (respectively $\vec{h}$), performing multiple iterations is equivalent to multiplying the initial vector by larger powers of $A^T A$ (respectively AA^T). The vector $\vec{a}$ (respectively $\vec{h}$) converges to the principal eigenvectors of $A^T A$ (respectively AA^T) under the assumption that $|\lambda_1(A)| > |\lambda_2(A)|$, where $\lambda_1(A), \lambda_2(A), \ldots, \lambda_n(A)$ are the eigenvalues of A sorted in descending order of their absolute value [Kleinberg, 1999].

2.5.5 HITS in similar-page queries

Kleinberg [Kleinberg, 1999] described the application of the HITS algorithm in the area of similar-page queries, that is, using the link structure to infer a notion of 'similarity' among pages. In a similar-page query, the question seeking answer is as follows: which pages are related to a particular web page v? From the perspective of HITS,

the question is [Kleinberg, 1999]: "what do users of the WWW consider to be related to v, when they create pages and hyperlinks?" If v is a page which has links from many pages then the link structure surrounding v cannot give a conclusive answer due to a huge number of independent opinions about the relation of v to other pages. To eliminate this problem the following question may be asked [Kleinberg, 1999]: "In the local region of the link structure near v, what are the strongest Authorities?" The set of those Authorities may serve as a broad-topic summary of the pages related to v.

For a similar-page query on a particular web page v, HITS finds the top c Authorities on a focused subgraph $\mathcal{W}_v$ constructed as follows [Kleinberg, 1999]:

1. The root set R_v is the set of top t ranked web pages linking to the page v.

2. R_v is expanded to a base set B_v as was done previously, that is, $B_v = \{x \in V(\mathcal{W}) : y \in R_v \text{ and } ((x,y) \in E(\mathcal{W}) \text{ or } (y,x) \in E(\mathcal{W}))\}$. As before, a limit of d backlinks per page is maintained here.

3. $\mathcal{W}[B_v]$ is the subgraph of $\mathcal{W}$ induced by the base set B_v, that is, $V(\mathcal{W}[B_v]) = B_v$ and $E(\mathcal{W}[B_v]) = \{(x,y) \in E(\mathcal{W}) : x \in B_v \text{ and } y \in B_v\}$.

4. A focused subgraph $\mathcal{W}_v$ is constructed as before by deleting all intrinsic links from $\mathcal{W}[B_v]$. That means $V(\mathcal{W}_v) = V(\mathcal{W}[B_v])$ and
   $E(\mathcal{W}_v) = \{(x,y) \in E(\mathcal{W}[B_v]) : x \text{ and } y \text{ are pages under different domains}\}$.

For similar-page queries, Dean and Henzinger [Dean and Henzinger, 1999] extended the HITS algorithm by including edge weights in the context of vicinity graphs, where the vicinity graph for a specified web page v includes v and a portion of the vertices of $\mathcal{W}$ that are close to v.

2.6 Trawling the Web

Kumar et al. [Kumar et al., 1999] identified implicit Web communities from the hypothesis that dense bipartite graphs, that are signatures of Web communities, contain at least one bipartite core; where a *bipartite core* is a complete bipartite graph of some specified size. Using their approach, a community can be found by first finding its core, and then using the core to find the rest of the community.

2.7 Additional Notations and Terminology from Graph Theory

A *directed graph* $G = (V, E)$ is a pair such that V is a set of vertices and $E \subseteq V \times V$ is a set of arcs or directed edges, where an *arc* from vertex u to vertex v is denoted as (u, v) where $u, v \in V$.

A *path* is a sequence $v_0, e_1, v_1, e_2, \ldots, e_k, v_k$ of vertices and arcs such that $e_i = (v_{i-1}, v_i)$ for $1 \leq i \leq k$ and no vertex is repeated. A *u, v-path* is a path that starts at vertex u and ends at vertex v. The *length* of a path is the number of arcs in the path.

If G has a u, v-path, then the *distance* from u to v, denoted by $d(u, v)$, is the minimum length of a u, v-path. If no u, v-path exists in G, then $d(u, v) = \infty$. The *diameter* of G is the maximum of the distances between all ordered pairs of vertices of G. In a disconnected graph, the diameter is ∞.

A directed graph G is *strongly connected* if there is a u, v-path in G for each ordered pair $u, v \in V(G)$. The *strongly connected components* of a directed graph G are the maximal strongly connected subgraphs of G.

The *in-neighborhood* or *predecessor set* $Pred(v)$ of a vertex v is $\{x \in V(G) :$

$(x, v) \in E(G)\}$. The *out-neighborhood* or *successor set* $Succ(v)$ of a vertex v is $\{x \in V(G) : (v, x) \in E(G)\}$. We extend these notations to a set $S \subseteq V(G)$ so that the in-neighborhood or predecessor set of S is $Pred(S) = \bigcup_{u \in S} Pred(u)$ and the out-neighborhood or successor set of S is $Succ(S) = \bigcup_{u \in S} Succ(u)$. The *indegree* of a vertex v is the number of incoming arcs of v, that is, $indegree(v) = |\{x \in V(G) : (x, v) \in E(G)\}| = |Pred(v)|$. The *outdegree* of a vertex v is the number of outgoing arcs of v, that is, $outdegree(v) = |\{x \in V(G) : (v, x) \in E(G)\}| = |Succ(v)|$.

Chapter 3

Neighborhood Graphs

This chapter describes the concept, construction and various properties of neighborhood graphs. At first we give the definition of the neighborhood graph followed by the method of construction of the neighborhood graphs. Then we present a comparison of relative size of regions in the bow-tie structure in the neighborhood graph for some web pages, followed by a discussion of the indegree and outdegree distribution in the neighborhood graph. In each case, we also present a comparison with the corresponding property in the whole web graph. The main results in this chapter have been submitted for publication [Nargis et al., 2007].

3.1 Definition

The motivating idea of the neighborhood graph is to model the local structure of the Web around a particular web page. Given a vertex v and a positive integer D, we define the neighborhood graph $\mathcal{N}_D(v)$ as follows:

1. Let $S_D(v)$ be the set of vertices in $\mathcal{W}$ which are at distance less than or equal to D from v in $\mathcal{W}$, that is, $S_D(v) = \{u \in V(\mathcal{W}) : d(u,v) \leq D\}$ where $d(u,v)$ is

the length of a shortest undirected path between u and v in $\mathcal{W}$. This set $S_D(v)$ is often called the D-th neighbor set of v.

2. $\mathcal{N}_D(v)$ is the subgraph of $\mathcal{W}$ induced by the set $S_D(v)$, that is, $V(\mathcal{N}_D(v)) = S_D(v)$ and $E(\mathcal{N}_D(v)) = \{(x, y) \in E(\mathcal{W}) \ : \ x \in S_D(v) \text{ and } y \in S_D(v)\}$.

For each integer $k \geq 0$, let $L_k(v)$ be the set of vertices or the *layer* of vertices in $\mathcal{W}$ which are at distance k from v in $\mathcal{W}$. Then $S_D(v) = \bigcup_{k=0}^{D} L_k(v)$, where $L_0(v) = \{v\}$, and for each $k \geq 1$, $L_k(v) = \{u \in V(\mathcal{W}) - S_{k-1}(v) : x \in L_{k-1}(v) \text{ and } ((x, u) \in E(\mathcal{W}) \text{ or } (u, x) \in E(\mathcal{W}))\}$.

Let $u \in V(\mathcal{W})$. For each $k \geq 0$, let $Succ_{k,v}(u) = \{w \in L_k(v) : (u, w) \in E(\mathcal{W})\}$ and $Pred_{k,v}(u) = \{w \in L_k(v) : (w, u) \in E(\mathcal{W})\}$. Given a subset S of $V(\mathcal{W})$, define $Succ_{k,v}(S) = \bigcup_{u \in S} Succ_{k,v}(u)$ and $Pred_{k,v}(S) = \bigcup_{u \in S} Pred_{k,v}(u)$. We now observe that for each $k \geq 1$, $L_k(v) = Succ_{k,v}(L_{k-1}(v)) \cup Pred_{k,v}(L_{k-1}(v))$.

It is noteworthy that a vertex in layer ℓ can have neighbors only in the layers ℓ, $\ell - 1$, and $\ell + 1$.

Figure 3.1 shows a sample neighborhood graph $\mathcal{N}_D(v)$ for $D = 3$. In this figure, the magenta region denotes $L_0(v)$, that is, only the vertex v (drawn as a red node) is in this region. Then we collect the vertices in $Succ_{1,v}(L_0(v))$ and $Pred_{1,v}(L_0(v))$; these vertices form the layer $L_1(v)$ which is shown as the blue region. In $L_1(v)$ each vertex is shown as a cyan node. The vertices in $Succ_{2,v}(L_1(v))$ and $Pred_{2,v}(L_1(v))$ form the layer $L_2(v)$ shown as the light blue region; each vertex in this layer is drawn as a dark brown node. Similarly, the purple layer $L_3(v)$ is formed by collecting the vertices in $Succ_{3,v}(L_2(v))$ and $Pred_{3,v}(L_2(v))$ where each vertex is represented by a pink node. The white arcs in the figure represent the arcs in $E(\mathcal{W})$ which were used to construct the layers, that is, for finding the vertices in the next layer in the iterative manner. In other words, white arcs are the arcs between two successive layers. The

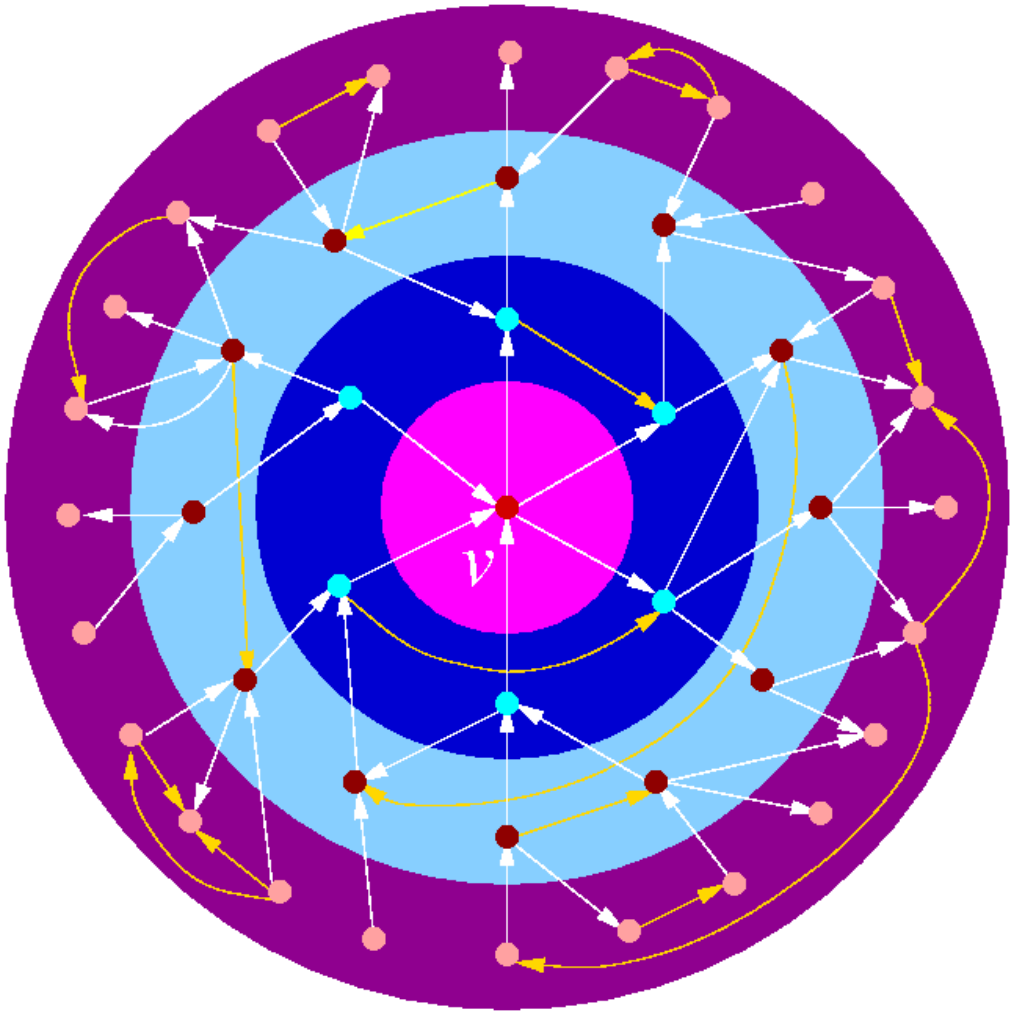

Figure 3.1: A sample neighborhood graph $\mathcal{N}_3(v)$.

yellow arcs are the arcs between vertices in the same layer as each other.

Observe that $\mathcal{N}_D(v)$ is a connected graph. Furthermore, $\mathcal{N}_D(v)$ has a diameter of at most $2D$ since each vertex is at most distance D from v.

3.2 Construction of the neighborhood graph

To implement the construction of the neighborhood graph $\mathcal{N}_D(v)$ for a web page v, we perform the following steps:[1]

1. $L_0(v) = \{v\}$.
   $S_0(v) = L_0(v)$.

2. `For` $k = 0$ to $(D-1)$ `do`

   (a) Find $Succ(u)$ for all $u \in L_k(v)$.

   To find $Succ(u)$ for $u \in L_k(v)$, we read the content of the web page corresponding to the vertex u. Then we scan the content and parse for any string that represents an URL. These linked URLs are the web pages linked to by u, that is, the vertices in $Succ(u)$.

   Define $Succ(L_k(v)) = \bigcup_{u \in L_k(v)} Succ(u)$.

   (b) Find $Pred(u)$ for all $u \in L_k(v)$.

   To find $Pred(u)$ for $u \in L_k(v)$, we use Google's Public Web API Services.[2] In Google's Public Web API Services there is a limit of one thousand queries per day for each user, where each query can return at most ten web pages. So we limit ourselves to the fifty top-ranked backlinks per page, that means we collect the top fifty ranked web pages within all the

[1] The program code for constructing $\mathcal{N}_2(v)$ is presented in Appendix A.
[2] The program code for this task is presented in Appendix B.

pages that link to the page corresponding to the vertex u. These fifty pages form $Pred(u)$.

Define $Pred(L_k(v)) = \bigcup_{u \in L_k(v)} Pred(u)$.

(c) Set $N(L_k(v)) = Succ(L_k(v)) \cup Pred(L_k(v))$. This set $N(L_k(v))$ denotes the set of all neighbors of the vertices in the set $L_k(v)$; that is, $N(L_k(v))$ is the set of all the web pages that are one click forward or backward from any page in the set $L_k(v)$.

Let $L_{k+1}(v) = N(L_k(v)) - S_k(v)$.

(d)

$$\begin{aligned} S_{k+1}(v) &= \bigcup_{m=0}^{k+1} L_m(v) \\ &= \left(\bigcup_{m=0}^{k} L_m(v) \right) \cup L_{k+1}(v) \\ &= S_k(v) \cup \big(N(L_k(v)) - S_k(v) \big) \\ &= S_k(v) \oplus N(L_k(v)), \end{aligned}$$

where $\oplus$ denotes the exclusive or function. To implement this function we used the `uniq` command in linux shell script.

`End For`

3. $V(\mathcal{N}_D(v)) = S_D(v)$.

4. To find the possible arcs between the pairs of vertices in $S_D(v)$ we perform the following steps.[3]

[3]The program code for this task is presented in Appendix C.

(a) $E(\mathcal{N}_D(v)) = \phi$.

(b) `For` each $x \in S_D(v)$ `do`

i. Find $Succ(x)$.

We find all the pages linked to by the web page corresponding to the vertex x in a similar manner as was described in step 2a.

ii. `For` each $y \in Succ(x)$ `do`

`if` $y \in S_D(v)$ `then`

$E(\mathcal{N}_D(v)) = E(\mathcal{N}_D(v)) \cup \{(x, y)\}$

`End For`

`End For`

3.3 Similarities and differences between the input graph in similar-page queries by HITS and neighborhood graphs

In Section 2.5 we described Kleinberg's approach of applying HITS to similar-page queries. The graph on which HITS operates to extract the best authorities related to a particular web page v is denoted by $\mathcal{W}_v$. The motivation for the construction of $\mathcal{N}_D(v)$ is to model the local structure surrounding the web page v whereas the motivating reason to construct $\mathcal{W}_v$ was to find the best authorities in the local region near v.

There are some similarities between $\mathcal{W}_v$ and the neighborhood graph $\mathcal{N}_D(v)$, as follows:

1. Both are focused subgraphs of $\mathcal{W}$ modeling the local structure surrounding the

web page v.

2. Both incorporate pages that link to v.

3. Both maintain a limit on the number of backward links of a particular web page.

Despite these similarities, there are several noteworthy differences between $\mathcal{W}_v$ and $\mathcal{N}_D(v)$, as follows:

1. The neighborhood graph $\mathcal{N}_D(v)$ is a more general idea than $\mathcal{W}_v$. In $\mathcal{N}_D(v)$ all web pages that are at most distance D from v are incorporated. In one extent, $\mathcal{N}_D(v)$ consists of only the page v itself when $D = 0$. On the other extreme, when $D \to \infty$, we include all web pages so $\lim_{D\to\infty} \mathcal{N}_D(v) = \mathcal{W}$. The graph $\mathcal{N}_D(v)$ expands by layers surrounding v by increasing the value of D. On the contrary, $\mathcal{W}_v$ incorporates some portion of only up to layer $L_2(v)$ of $\mathcal{N}_D(v)$. More precisely, $\mathcal{W}_v$ includes only the backward links of v so a significant number of pages of $L_1(v)$ (the forward links of v) are not included in $\mathcal{W}_v$. The graph $\mathcal{W}_v$ incorporates some pages of $L_2(v)$ by including the backward and forward links from the pages that link to v. But $\mathcal{W}_v$ does not incorporate any of the pages in $L_D(v)$, where $D > 2$. Therefore, $V(\mathcal{W}_v) \subset V(\mathcal{N}_D(v))$ for $D \geq 2$. The size of the difference set, $|V(\mathcal{N}_D(v)) - V(\mathcal{W}_v)|$ is a monotonically increasing function on D. Figure 3.2 illustrates the difference between $\mathcal{W}_v$ and $\mathcal{N}_D(v)$ by presenting the graph $\mathcal{W}_v$ corresponding to the sample neighborhood graph $\mathcal{N}_3(v)$ depicted in Figure 3.1. By comparing Figure 3.2 to Figure 3.1 we see that only a portion of pages in the layers $L_1(v)$ and $L_2(v)$ are included in $\mathcal{W}_v$. Similar to $\mathcal{W}_v$, the vicinity graph of v studied in [Dean and Henzinger, 1999] is also a subgraph of $\mathcal{N}_2(v)$.

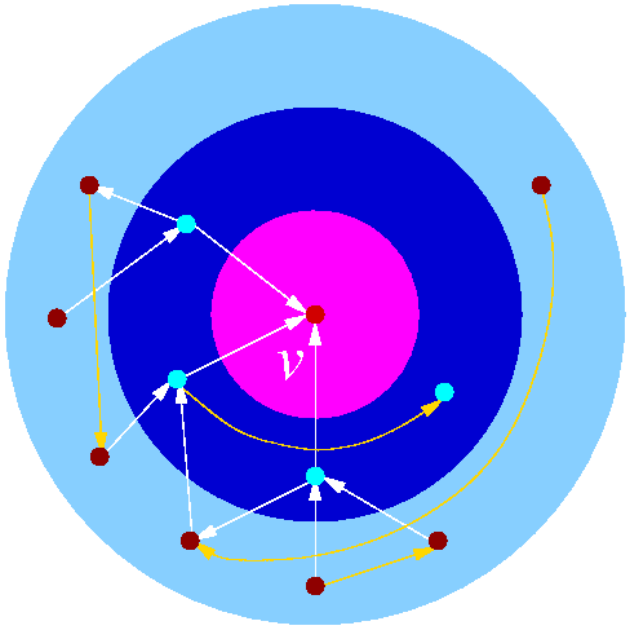

Figure 3.2: The graph $\mathcal{W}_v$ corresponding to the sample neighborhood graph $\mathcal{N}_3(v)$ presented in Figure 3.1.

2. By incorporating web pages up to a distance of D from v, $\mathcal{N}_D(v)$ should have a greater number of good authorities related to v than $\mathcal{W}_v$ for $D \geq 2$. Therefore, it satisfies criterion 3 better than $\mathcal{W}_v$. On the other hand, by including more pages it satisfies criterion 1 worse than $\mathcal{W}_v$.

3. All intrinsic links are deleted during the construction of $\mathcal{W}_v$ whereas $\mathcal{N}_D(v)$ includes all intrinsic links.

3.4 Experimental Results

3.4.1 Relative Size of Regions in the bow-tie structure

In their experiment Broder et al. [Broder et al., 2000] found SCC as the dominant region of the web graph $\mathcal{W}$. This region comprised about 28% of the entire Web. The

regions IN, OUT and TUBES and TENDRILS were approximately the same size of about 21%. The DISC region accounted for about 8% of the web pages.

Donato et al. [Donato et al., 2004] analyzed the relative sizes of these regions on a Web crawl of about 200 million web pages and 1.4 billion hyperlinks. In their experiment instead of SCC, the OUT region was the largest region with about 39% of the whole Web. The SCC had a proportion of about 33% which is larger than the corresponding 28% in the crawl analyzed by [Broder et al., 2000]. The proportion of pages in the regions IN, DISC, and TUBES and TENDRILS were observed to be about 11%, 4%, and 13%, respectively.

We have performed a similar dissection of the neighborhood graph $\mathcal{N}_2(v)$ into the bow-tie structure regions.[4] In Table 3.1 we present the relative size of regions in the bow-tie structure in $\mathcal{W}$ as observed in Web crawls conducted in 1999 [Broder et al., 2000] and in 2001 [Donato et al., 2004], and then we give a comparison with the relative size of regions in the bow-tie structure in $\mathcal{N}_2(v)$ for some common web pages.

Graph being considered	**Region Size (%)**				
	IN	**SCC**	**OUT**	**TT**	**DISC**
$\mathcal{W}$ in 1999 [Broder et al., 2000]	21.29	27.74	21.21	21.52	8.24
$\mathcal{W}$ in 2001 [Donato et al., 2004]	11.00	33.00	39.00	13.00	4.00
$\mathcal{N}_2(v)$, v = www.mail.yahoo.com	12.20	34.84	52.96	0.00	0.00
$\mathcal{N}_2(v)$, v = www.google.com	3.43	68.82	27.75	0.00	0.00
$\mathcal{N}_2(v)$, v = www.acm.org	25.97	51.86	22.17	0.00	0.00
$\mathcal{N}_2(v)$, v = www.csebuet.org	25.83	29.52	3.32	41.33	0.00

Table 3.1: Relative size of regions in the web graph $\mathcal{W}$ and the neighborhood graph $\mathcal{N}_2(v)$.

Since $\mathcal{N}_D(v)$ is a connected graph, there are no pages in DISC. The distribution

[4]The program code for finding the Bow-Tie regions in a graph G is presented in Appendix D.

of pages among the regions in $\mathcal{N}_D(v)$ varies according to the type of the starting page v. One noteworthy example is $\mathcal{N}_2(v)$ with v= www.mail.yahoo.com; in this case the distribution is very much similar to $\mathcal{W}$ (IN – 12.2% vs. 11%; SCC – 34.84% vs. 33%). In other examples, such as $\mathcal{N}_2(v)$ for v = www.acm.org and v=www.google.com, it is SCC that is the dominant region (51.86% and 68.82%, respectively). With a value of $D = 2$, we found very few examples in which the TUBES and TENDRILS region (denoted by TT in Table 3.1) was non-empty. One exceptional example was www.csebuet.org which is an alumni website of a university; the pages in close proximity to this web page exhibit a wide variety in content so the resulting graph is relatively sparse and the TT region is non-empty.

3.4.2 The Degree Distribution

We performed an analysis of the indegree and outdegree distributions in some neighborhood graphs.[5]

3.4.2.1 Example 1 – $\mathcal{N}_D(v)$ for v = www.mun.ca

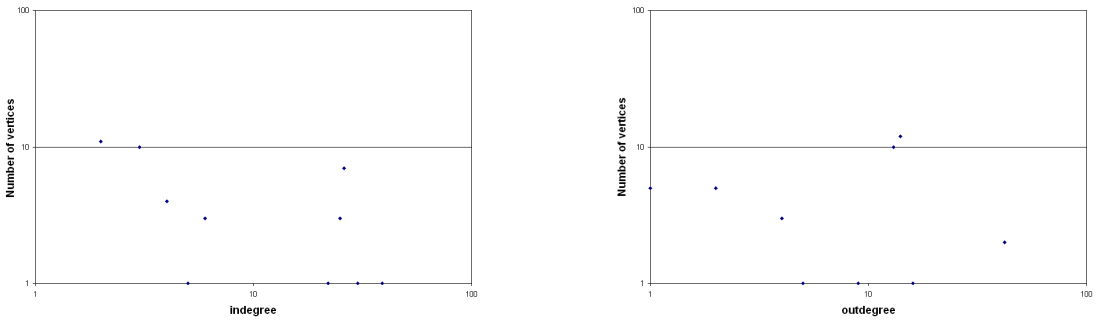

Figure 3.3: The indegree and the outdegree distribution of $\mathcal{N}_1(v)$ for v = www.mun.ca; $\mathcal{N}_1(v)$ has 43 web pages.

[5]The program code for producing the indegree and outdegree distribution in a graph G is presented in Appendix E.

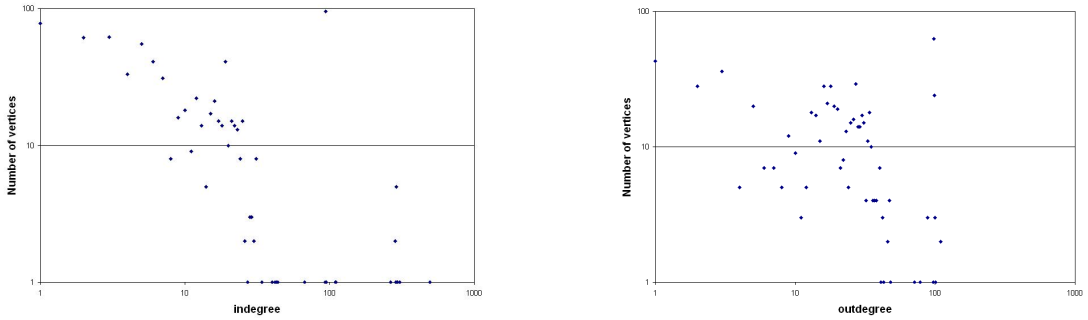

Figure 3.4: The indegree and the outdegree distribution of $\mathcal{N}_2(v)$ for v = www.mun.ca; $\mathcal{N}_2(v)$ has 774 web pages.

As a first example, Figure 3.3 and 3.4 present the indegree and the outdegree distribution of $\mathcal{N}_D(v)$ with v = www.mun.ca as of June 06, 2007 for $D = 1$, and 2, respectively. In these examples we do not observe a power law distribution, likely because these example graphs are too small to be statistically valid.

3.4.2.2 Example 2 – $\mathcal{N}_2(v)$ for v = www.imdb.com

As a second example, Figure 3.5 and 3.6 present the indegree and the outdegree distribution of $\mathcal{N}_2(v)$ with v = www.imdb.com as of June 11, 2006 ($\mathcal{N}_2(v)$ has 11,848 vertices). From Figure 3.5 we see that the indegree distribution seem to follow a power law if we discard the initial two points, meaning that the indegree distribution follows a power law after indegree of two. Figure 3.6 suggests that the outdegree distribution does not follow a power law.

3.4.2.3 Example 3 – $\mathcal{N}_2(v)$ for v = www.wikipedia.org

As another example, Figure 3.7 shows the indegree distribution of $\mathcal{N}_2(v)$ with v = www.wikipedia.org in June 2006 ($\mathcal{N}_2(v)$ has 27,795 web pages). The indegree distribution in $\mathcal{N}_2(v)$ in Figure 3.7 seems to follow a power law after a certain initial portion.

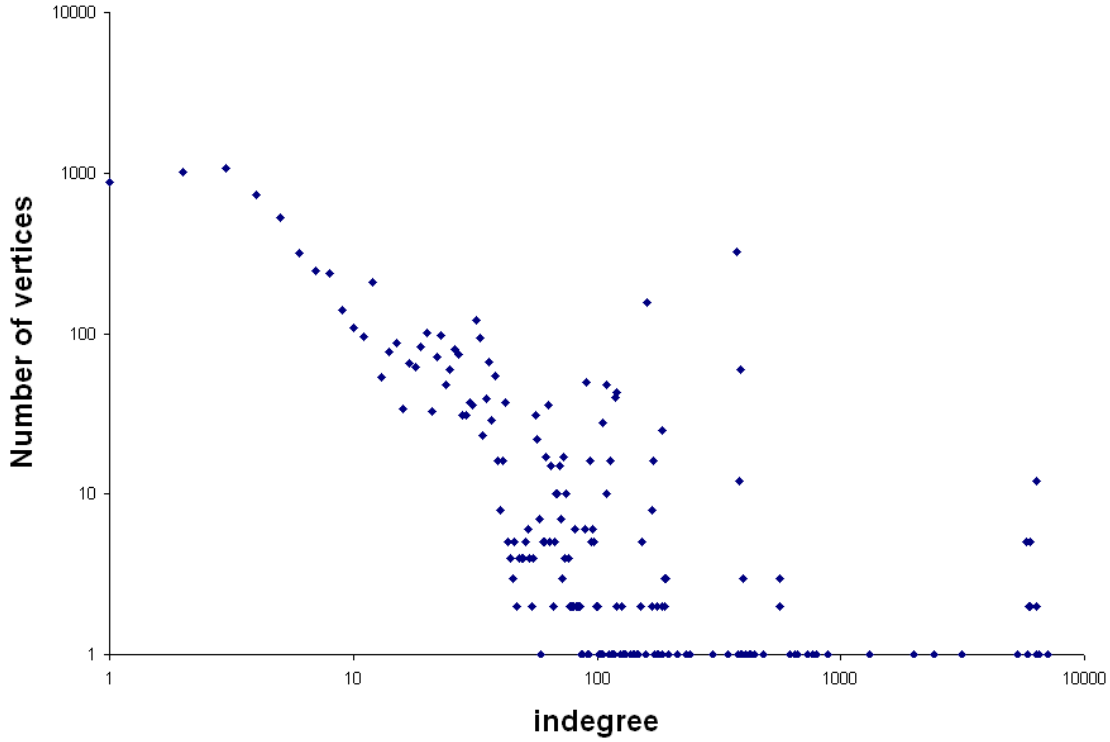

Figure 3.5: The indegree distribution of $\mathcal{N}_2(v)$ for $v =$ www.imdb.com.

To be more precise, it approximately follows a power law after indegree of six. To calculate the exponent, linear regression is used to find the best power law fit for the distribution starting at indegree seven; yielding an exponent of 1.37; the corresponding power law line is shown as the pink line in Figure 3.7. We observe that this line does not agree with the linear portion of the indegree distribution that is exemplified by the indegree values in the interval from 7 to 100; a linear regression performed on the data points in this subinterval yields an exponent of 2.10, as illustrated by the red line in Figure 3.7. This latter value we note is consistent with the web graph as a whole, for which [Barabási and Albert, 1999, Broder et al., 2000, Donato et al., 2004] obtained a power law exponent of 2.1.

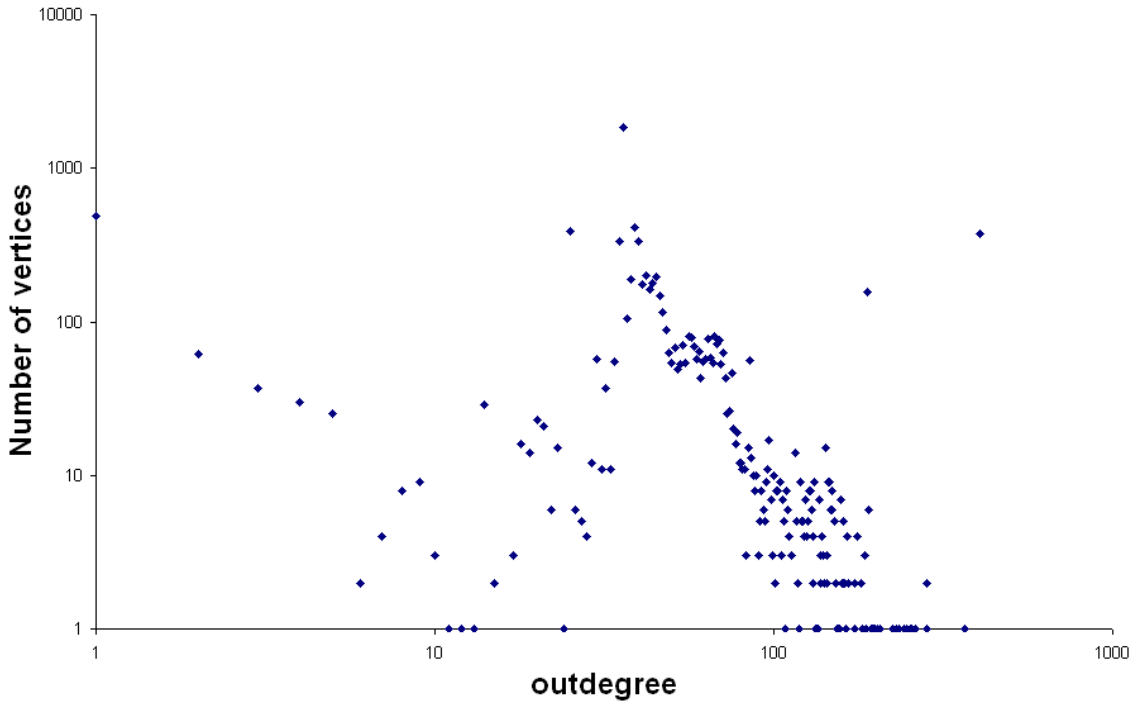

Figure 3.6: The outdegree distribution of $\mathcal{N}_2(v)$ for v = www.imdb.com.

Figure 3.8 shows the outdegree distribution of $\mathcal{N}_2(v)$ with v = www.wikipedia.org. Similar to the indegree distribution, the outdegree distribution in $\mathcal{N}_2(v)$ approximately follows a power law with an exception in the initial portion. In this case, the distribution follows a power law for outdegree at least sixteen. For the outdegree distribution starting at outdegree sixteen the exponent for the best power law fit is 1.72, shown as the pink line in Figure 3.8. In this case too, we find that this line differs from the line that the data points themselves appear to form. Performing a linear regression over the interval from outdegree 16 to 200 produces a line of best fit with exponent 2.64 (shown in red in Figure 3.8), which is very close to the value of the outdegree exponent for the whole web graph (2.72).

From Figure 3.7 and Figure 3.8 we see that both the indegree distribution and

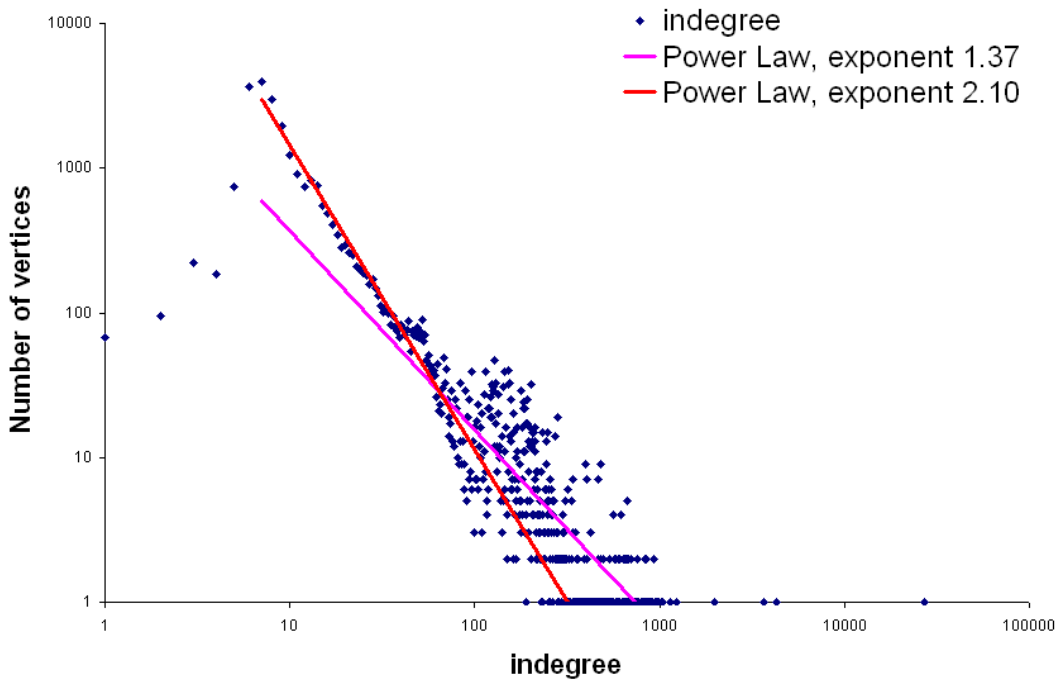

Figure 3.7: The indegree distribution of $\mathcal{N}_2(v)$ for v = www.wikipedia.org.

the outdegree distribution of $\mathcal{N}_2(v)$ approximately follow the power law, except for the initial segment. These exceptions are due to the high connectivity and the small diameter of $\mathcal{N}_2(v)$, and the correspondingly small number of web pages with small indegree or outdegree.

From the indegree and outdegree distributions presented we come to the conclusion that small instances of $\mathcal{N}_D(v)$ do not exhibit a power law degree distribution. But in case of a sufficiently large instance of $\mathcal{N}_D(v)$ like $\mathcal{N}_2(v)$ for v= www.wikipedia.org, both the indegree and outdegree distribution of $\mathcal{N}_D(v)$ exhibit the power law phenomenon with exponents that are comparable to those for $\mathcal{W}$. The neighborhood graphs therefore exhibit better power law fit as $D \to \infty$.

The similarities that we have observed between the neighborhood graphs and

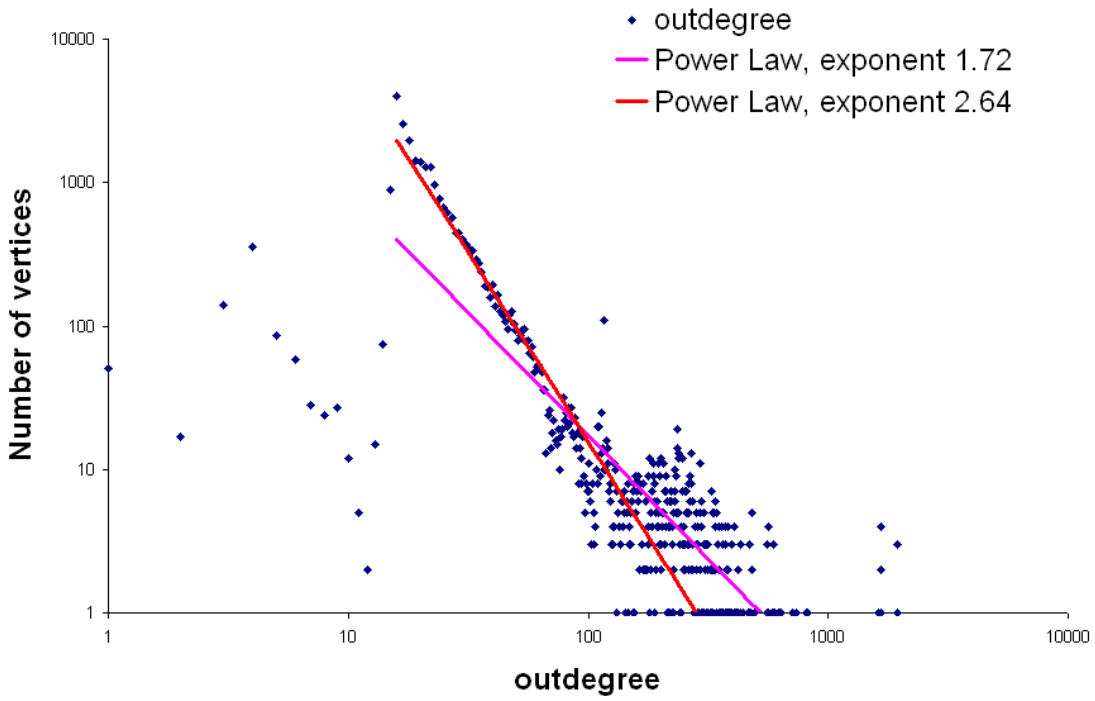

Figure 3.8: The outdegree distribution of $\mathcal{N}_2(v)$ for $v =$ www.wikipedia.org.

the whole web graph concerning indegree and outdegree distributions as well as the structural properties discussed in Section 3.4.1 are consistent with the self-similar nature of the Web observed in [Dill et al., 2002].

Chapter 4

Authorities and Hubs in a Local Scale

This chapter is devoted to the extraction of Authorities and Hubs in various aspects in the neighborhood graphs. In the World Wide Web there exist sets of authoritative and informative pages focusing on various topics; these pages are called *Authorities*. Pages that link to many related topics are called *Hubs*. A set of Authorities and Hubs on a particular topic represent a community structure on that topic in the Web. We describe the concepts of discovering Authorities and Hubs in neighborhood graphs, where the set of Authorities and Hubs gives a representative flavor on the theme of the page in question. We present the idea of the temporal evolution of Authorities and Hubs in neighborhood graphs that illustrates the transformation of the set of web pages thematically related to a particular web page. We also present the identification of core communities in the neighborhood graph, where a core community is comprised of the core Authorities and core Hubs that exist among the list of the topmost c Authorities and Hubs respectively, irrespective of the time of the experiment being conducted. The main results in this chapter have been submitted for publication

[Nargis et al., 2007].

4.1 Authorities and Hubs in a neighborhood graph

By applying the HITS algorithm to $\mathcal{N}_D(v)$ we can discover the most authoritative and Hub-like pages within a local scale.[1] From these we can get an idea of what the page v is about. This method has potential application in the extraction of thematically related web pages for a given web page.

Besides the differences listed in Section 3.3, there is a sharp difference between the approach in [Kleinberg, 1999] and our approach. In [Kleinberg, 1999] the HITS algorithm was applied to $\mathcal{W}_v$ to find the best Authorities related to v. We now apply HITS to $\mathcal{N}_D(v)$ to discover not only the best Authorities but also the best Hubs related to v. The motivation is that the Authorities alone do not show the full picture to capture the theme of the pages related to v, but together the Authorities and Hubs in $\mathcal{N}_D(v)$ form a dense community structure related to the theme of v.

4.1.1 Experimental Results

4.1.1.1 Example 1 – $\mathcal{N}_2(v)$ for v = www.cricinfo.com

Table 4.1 shows the top five Authorities and Hubs as of May 14, 2006 for the graph $\mathcal{N}_2(v)$ for v = www.cricinfo.com, which is a website on cricket. Here we see that the topmost Authority related to www.cricinfo.com is the homepage of The Wisden Group which is the owner company of the CricInfo website; the second best Authority is Statsguru – an online tool for analyzing cricket statistics; the third Authority in this local scale is CricShop – an online shop for cricket-related souvenirs; the fourth

[1] The program code implementing the HITS algorithm is presented in Appendix F.

Rank	Authorities	Hubs
1	The Wisden Group	Cricinfo.com - The Home of Cricket
2	Cricinfo - Statsguru	Cricinfo - Women's Cricket
3	Welcome to Cricshop!	Cricinfo - Tri-Nation Tournament in West Indies 2006
4	Cricinfo live video	Cricinfo - West Indies v Zimbabwe 2006
5	International Cricket Council	Cricinfo - West Indies v India 2006

Table 4.1: Top five Authorities and Hubs for $\mathcal{N}_2(v)$ with v = www.cricinfo.com.

Authority is the live video zone of the CricInfo website; and the fifth Authority is the homepage of the International Cricket Council. On the other hand, the topmost Hub is the CricInfo page itself, the second best Hub is the page for Women's Cricket, and the next three Hubs are web pages for ongoing cricket tournaments.

The Authorities indeed are resourceful and informative web pages on the topic of the game of cricket. It is noteworthy that one of the Authorities, the fifth Authority ('International Cricket Council') is not part of the website of the starting web page. The Hubs not only link to the authoritative pages but also reveal web pages that were popular at the time of our experiment. For example, the last three of the top five Hubs are pages on ongoing tournaments.

4.1.1.2 Example 2 – $\mathcal{N}_2(v)$ for v = www.imdb.com

As a second example, Table 4.2 shows the top five Authorities and Hubs as of June 11, 2006 for the graph $\mathcal{N}_2(v)$ for v = www.imdb.com which is the website for the Internet Movie DataBase. Here we see that the topmost Authority is Amazon.com, which is a vendor of movie DVDs, books etc.; the second best Authority is the Movie Trailers page of the IMDb website; the third Authority is the message board section of the IMDb website; the fourth Authority is the homepage of the IMDb website; and the fifth Authority is the Search page of the IMDb website. The five topmost Hubs

Rank	Authorities	Hubs
1	Amazon.com Online Shopping	Mission: Impossible III (2006)
2	Movie Trailers	The Da Vinci Code (2006)
3	IMDb Boards Main Boards	Firewall (2006)
4	The Internet Movie Database (IMDb)	X-Men: The Last Stand (2006)
5	Search the Internet Movie Database	Kiss Kiss Bang Bang (2005)

Table 4.2: Top five Authorities and Hubs for $\mathcal{N}_2(v)$ with v = www.imdb.com.

are pages on the most popular movies among the movies that were released at the time we conducted our experiment.

Here the Authorities are resourceful web pages on the topic of movies. It is interesting to observe that the topmost Authority ('Amazon.com') is not part of the website of the starting web page, although there is a corporate business relationship in that 'Amazon' owns the 'IMDb' website. The topmost Hubs reveal several web pages on movies that were popular at that time.

In both examples we see that the analysis of the topmost Authorities and Hubs in $\mathcal{N}_D(v)$ with a small value of D reveal the content and theme of the page in question, indicating a reflection of popular opinion at that time.

4.2 Temporal Evolution of Hyperlinked Communities in a neighborhood graph

Gibson et al. [Gibson et al., 1998] described temporal issues regarding Hubs and Authorities. Various short-term factors may influence the set of Authorities and Hubs. These short-term influences die out as pages and links are removed over time. Gibson et al. also indicated that to obtain the long-term core of a topic one can superimpose the results of HITS on the same topic, spaced over a several-month period. They also suggested comparing the results over time to understand the temporal evolution of

communities on the Web as a possible future research direction.

We performed a temporal analysis on the Authorities and Hubs in a neighborhood graph by applying HITS on $\mathcal{N}_D(v)$ for the same web page v on several dates separated by a significant time gap of several months.

4.2.1 Experimental Results

4.2.1.1 Example 1 – $\mathcal{N}_2(v)$ for v = www.cricinfo.com

As an example, Table 4.3 shows the top twenty five Hubs and Table 4.4 shows the top twenty five Authorities of $\mathcal{N}_2(v)$ for v = www.cricinfo.com for three different dates (May 31, 2006, January 31, 2007, and March 21, 2007).

Table 4.3: Top twenty five Hubs for $\mathcal{N}_2(v)$ with v = www.cricinfo.com for three different dates.

Rank	May 31, 2006	January 31, 2007	March 21, 2007
1	Cricinfo.com – The Home of Cricket	Cricinfo – Feedback	Cricinfo – Feedback
2	Cricinfo – Women's Cricket	Cricinfo – Who we are	Cricinfo – All Today's Yesterdays
3	Cricinfo – West Indies v India 2006	Cricinfo – Help and Feedback	Cricinfo – Who we are
4	Cricinfo – Statistics – Records	Cricinfo – The Curious Affair of Charlie Absolom	Cricinfo – Help and Feedback
5	Cricinfo – England v Sri Lanka 2006	Cricinfo – Take Tait to West Indies - Thomson	Cricinfo – Inzy's first hurrah
6	Cricinfo – England v Pakistan 2006	Cricinfo – Buchanan happy his bowlers were attacked	Cricinfo – Big picture, blurred vision
7	ICC Intercontinental Cup 2006	Cricinfo – Gough set to retire	Cricinfo – Woolmer's post-mortem inconclusive
8	Cricinfo – Fixtures	Cricinfo – 'Right now I have nothing but praise for Ganguly'	Cricinfo – Chairman tells Pakistan to play final game
9	Cricinfo – Match-series Archive	Cricinfo – Reconstructing Sehwag	Cricinfo – The wonder of Woolmer
10	Cricinfo – Live cricket scores – Ball-by-ball coverage	Cricinfo – Players and Officials – Jacob Oram	Cricinfo – Big winners, big players, big scorers
11	Cricinfo – Feedback	Cricinfo – Border claims Pietersen left tour too quickly	The Week That Was ... March 12 - March 18
12	About CricInfo	Cricinfo – 'I showed what I'm capable of' - Nixon	Cricinfo – Fuelled by team spirit
13	Cricinfo – Help and Feedback	Cricinfo – Players and Officials - Darren Gough	Cricinfo – Quote ___ unquote
14	Cricinfo – Reviews	Cricinfo – Ruling an impossible target	Cricinfo – The unforgiven

Table 4.3 is continued on the next page

This is the continuation of Table 4.3 from the previous page

Rank	May 31, 2006	January 31, 2007	March 21, 2007
15	Cricinfo – Site Map	Cricinfo – Making money and losing money	Cricinfo – LG ICC Test and ODI Championships
16	Cricinfo – Recent Results – Last 7 Days	Cricinfo – A batting line-up to die for	Wisden cricketers almanack 2006
17	Cricinfo - Forthcoming live coverage (next seven days) - Ball-by-ball coverage	Cricinfo – Players and Officials – Mohammad Asif	Wisden Almanack – Archive
18	Cricinfo - Cricinfo Magazine	Cricinfo – Shoaib - 'Woolmer thought injury was fake'	Wisden Almanack – Archive
19	Cricinfo – The Wisden Cricketer	Cricinfo – A long way from home	Cricinfo – Fixtures
20	Link to us	Cricinfo – All Today's Yesterdays	Cricinfo – Recent Results – Last 7 Days
21	Cricinfo – 5th ODI West Indies v India at Port of Spain, May 28, 2006	Cricinfo – The first All-India side	Cricinfo – Forthcoming live coverage (next seven days) – Ball-by-ball coverage
22	Cricinfo – Yorkshire v Scotland at Leeds, 29 May 2006	Cricinfo – Recovering from 0 for 5, and the shortest ODIs	Cricinfo – Grounds
23	Cricinfo – Cricket RSS feeds	Cricinfo – Slapping, slashing and sexier than Cruise	Cricinfo – Players and Officials
24	Cricinfo live video	Wisden cricketers almanack 2006	Cricinfo – Cricinfo Talk
25	Cricinfo – Mobile	Cricinfo – New Zealand complete solid victory	Cricinfo – Cricinfo Talk – The Tony Greig Show - Feature Home

We may focus on the temporal parts by concentrating on the pages which appear as Hubs (respectively Authorities) on only one of the three dates (and hence, appear in only one column in the corresponding tables). These pages contain some specific information about that time and they help to model the temporal evolution of communities.

The topmost Hubs on May 31, 2006 include some of the ongoing cricket matches

at that time (for example, the third and the fifth to seventh ranked Hubs).

The list of the topmost twenty five Hubs on January 31, 2007 is dominated by a number of remarks by cricketers and coaches and the profiles of some popular cricketers at that time. For example, the eighth ranked Hub describes the comments of Indian coach Greg Chappell about Indian cricketer Ganguly and the twelfth ranked Hub is on English player Nixon's comments after a remarkable batting. The tenth and seventeenth ranked Hubs are the profiles of two cricketers who performed well at that time.

In the list of the topmost twenty five Hubs on May 21, 2007, when the Cricket World Cup 2007 was in progress, we see articles on ongoing matches, events and archive articles related to the theme of the Cricket World Cup. For example, the sixth ranked Hub is an article on the Cricket World Cup 2007 teams and the twelfth ranked Hub tells about the unexpectedly good performance by a new Team Ireland. A prominent event at that time was the suspicious murder of Pakistani Coach Bob Woolmer, and we see related pages to this event as the seventh, eighth and ninth ranked Hubs. The sixteenth ranked Hub is an article on a match in Cricket World Cup 1992 and the seventeenth ranked Hub describes West Indies winning the first Cricket World Cup that took place in 1975.

Table 4.4: Top twenty five Authorities for $\mathcal{N}_2(v)$ with v = www.cricinfo.com for three different dates.

Rank	May 31, 2006	January 31, 2007	March 21, 2007
1	The Wisden Group	Cricinfo – Statsguru	The Wisden Group
2	Cricinfo – Statsguru	International Cricket Council	Cricshop.com
3	Cricshop.com – The Cricket Experts	Cricket Manager – Cricket Game Online – Management Sports Simulation Internet	The Wisden Group – People – Current Vacancies
4	International Cricket Council	The Ashes Top Trumps	Cricinfo.com – The Home of Cricket
5	Cricinfo World XI Fantasy League	The Wisden Group	Cricinfo – Cricinfo Magazine
6	Cricinfo live video	Cricshop.com Catalogue	Cricinfo – The Wisden Cricketer
7	Cricinfo Betting	The Wisden Group – People – Current Vacancies	Cricinfo – Cricinfo Games – Games Home
8	Cricinfo.com – The Home of Cricket	Cricshop.com	Cricinfo – Cricinfo Talk
9	Cricinfo – Feedback	Cricinfo.com – The Home of Cricket	Cricinfo – Cricket World Cup 2007 in West Indies
10	Cricinfo – Cricinfo Magazine	Cricinfo Games – the best cricket games online	Cricinfo – Feedback
11	Cricinfo – The Wisden Cricketer	Cricinfo – All Today's Yesterdays	Cricinfo – Who we are
12	Cricinfo – Live cricket scores – Ball-by-ball coverage	Cricinfo – Grounds	Cricinfo – Site Map
13	Cricinfo – Photo Index – Global	Cricinfo – Image Index – Global	About CricInfo
14	Cricinfo Media Kit	Cricinfo – Players and Officials	Cricinfo – Cricket RSS feeds
15	Link to us	Cricinfo – News Index – Global	Link to us
16	About CricInfo	Cricinfo – Fixtures	Wisden Almanack – Archive

Table 4.4 is continued on the next page

This is the continuation of Table 4.4 from the previous page

Rank	May 31, 2006	January 31, 2007	March 21, 2007
17	Cricinfo – Site Map	Cricinfo – Live cricket scores – Ball-by-ball coverage	Cricinfo – Search Cricinfo
18	Cricinfo – Who we are	Cricinfo – Recent Results – Last 7 Days	Cricinfo – Sehwag stars in thumping win
19	Wisden Almanack – Archive – From the archives: the first Test of 1957	Cricinfo – Cricinfo Magazine	Cricinfo – West Indies through without impressing
20	Cricinfo – Players and Officials – Rahul Dravid	Cricinfo – Match-series Archive	Cricinfo – Woolmer death Hard news takes a back seat
21	Cricinfo – Players and Officials – Sourav Ganguly	Cricinfo – Commonwealth Bank Series 2006-07	Cricinfo – Normal service resumed
22	Cricinfo – England amble to six-wicket victory	Cricinfo – ICC World Cricket League 2006-07, Division 1	Cricinfo – Dravid expects more from Sehwag
23	Cricinfo – Players and Officials – Kevin Pietersen	Cricinfo – South Africa v Pakistan 2006-07	Cricinfo – Players and Officials – Bob Woolmer
24	Cricinfo – Fletcher 'Panesar under a lot of pressure'	Cricinfo – India v West Indies 2006-07	Cricinfo – Players and Officials – Dwayne Leverock
25	Cricinfo – Cricket RSS feeds	ICC Intercontinental Cup 2006	Cricinfo – Players and Officials – Virender Sehwag

In the list of the top twenty five Authorities on May 31, 2006 we see the profiles of some cricketers who gained popularity at that time and some articles and remarks by players on ongoing matches. The twenty-first, twenty-second and twenty-fourth ranked pages are the profile pages of three cricketers. The twenty-third ranked page is an article on the victory of England with Sri Lanka in a test that ended on May 28, 2006 and the twenty-fourth ranked Authority describes the need of an England spinner to start all-round game.

The last five of the topmost twenty five Authorities on January 31, 2007 deal with ongoing matches at that time which is apparent from the web page titles.

The most prominent temporal effect on the Authorities is marked in the list on March 21, 2007. The ninth ranked Authority is a web page on the Cricket World Cup 2007 which was in progress at that time. The eighteenth to twenty-second ranked Authorities are articles and events related to ongoing matches whereas the twenty-fourth and twenty-fifth ranked Authorities are profile pages of popular cricketers at that time. Similar to the topmost Hubs at that time, the Woolmer murder also dominates in the list of Authorities – the twentieth ranked Authority is a related article and the twenty-third ranked Authority is the profile page of Bob Woolmer himself.

From these results we see that the topmost Authorities and Hubs illustrate the temporal evolution of the communities in the neighborhood structure of the web page in question. In each case we find pages on current events appear in the list of topmost Hubs and Authorities. It is noteworthy that the Hubs show more temporal effect than the Authorities in the sense that we find more pages focused on contemporary events in the list of topmost twenty five Hubs than in the list of topmost twenty five Authorities, regardless of the date on which we performed our experiment.

4.2.1.2 Example 2 – $\mathcal{N}_2(v)$ for v = www.imdb.com

As another example, Table 4.5 shows the top twenty five Hubs and Table 4.6 shows the top twenty five Authorities of $\mathcal{N}_2(v)$ for v = www.imdb.com for two different dates (February 01, 2007, and June 03, 2007). As before, we attempt to focus on the temporal effects by locating pages that appear in only one column.

Table 4.5: Top twenty five Hubs for $\mathcal{N}_2(v)$ with v = www.imdb.com for two different dates.

Rank	February 01, 2007		June 03, 2007	
	Web Page Title	**Release Date/ DVD Release Date**	**Web Page Title**	**Release Date/ DVD Release Date**
1	Little Miss Sunshine (2006)	DVD released on December 19, 2006	The Internet Movie Database (IMDb)	
2	Justin Timberlake		Movie Keywords Analyzer (MoKA)	
3	Flags of Our Fathers (2006)	DVD released on February 06, 2007	IMDb - Now Playing in the US	
4	Casino Royale (2006)	DVD released on March 13, 2007	Movie & TV News @ IMDb.com – Main Page	
5	Night at the Museum (2006)	22 December 2006 (USA)	Evan Almighty (2007)	22 June 2007 (USA)
6	Drop Dead Gorgeous (1999)		IMDb Charts	
7	Children of Men (2006)	5 January 2007 (USA)	Pirates of the Caribbean At World's End (2007)	25 May 2007 (USA)
8	Smokin' Aces (2006)	26 January 2007 (USA)	Knocked Up (2006)	1 June 2007 (Canada)
9	Dreamgirls (2006)	25 December 2006 (USA)	Shrek the Third (2007)	18 May 2007 (USA)
10	Salma Hayek		Spider-Man 3 (2007)	4 May 2007 (USA)
11	Ellen DeGeneres		Hannibal Rising (2007)	DVD released on May 29, 2007
12	Running with Scissors (2006)	DVD released on February 06, 2007	The Closer (2005)	DVD of the second season released on May 29, 2007

Table 4.5 is continued on the next page

This is the continuation of Table 4.5 from the previous page

Rank	February 01, 2007		June 03, 2007	
	Web Page Title	**Release Date/ DVD Release Date**	**Web Page Title**	**Release Date/ DVD Release Date**
13	Flyboys (2006)	DVD released on January 30, 2007	F Troop (1965)	DVD of the second season released on May 29, 2007
14	The Marine (2006)	DVD released on January 30, 2007	Seinfeld (1990)	DVD of the eighth season released on June 05, 2007
15	Poseidon (2006)		The Messengers (2007)	DVD released on June 05, 2007
16	Inside Man (2006)		Rescue Me (2004)	DVD of the third season released on June 05, 2007
17	Science des rves, La (2006)	DVD of the English version released on February 06, 2007	Fantastic Voyage (1966)	DVD released on June 05, 2007
18	Catch and Release (2006-II)	26 January 2007 (USA)	The Dead Zone (2002)	DVD of the fifth season released on June 05, 2007
19	United 93 (2006)		Spider-Man 3 (2007) – Trivia	4 May 2007 (USA)
20	The Prestige (2006)	DVD released on February 20, 2007	28 Weeks Later (2007) – Trivia	May 11, 2007 (USA)
21	Silent Hill (2006)		Blazing Saddles (1974)	
22	The Black Dahlia (2006)	DVD released on December 26, 2006	Edison Kinetoscopic Record of a Sneeze (1894)	
23	The Devil Wears Prada (2006)	DVD released on December 12, 2006	The Andy Griffith Show (1960)	DVD of the complete series released on May 29, 2007
24	Casanova (2005)	DVD of the French version released on January 23, 2007	Captivity (2007)	22 June 2007 (USA)

Table 4.5 is continued on the next page

This is the continuation of Table 4.5 from the previous page

Rank	February 01, 2007		June 03, 2007	
	Web Page Title	**Release Date/ DVD Release Date**	**Web Page Title**	**Release Date/ DVD Release Date**
25	Night at the Museum (2006)	22 December 2006 (USA)	Rise: Blood Hunter (2007)	1 June 2007 (USA)

The list of the topmost twenty five Hubs on February 01, 2007 is dominated by three types of pages:

1. Pages on movies that were released within a short time frame from February 01, 2007. For example, the eighth and the eighteenth ranked Hubs are pages on two movies released on January 26, 2007. In total we found six Hubs of this type.

2. Pages on movies that were released in DVD format within a short time interval around February 01, 2007. For example, the third, twelfth and seventeenth ranked Hubs are pages on movies whose DVD were released on February 06, 2007. There are eleven Hubs of this category.

3. Profile pages of some popular actors at that time. For example, the second ranked Hub is the profile page of a popular actor at that time. There are three Hub pages of this type.

In the list of the best Hubs on June 03, 2007 we found the following types of pages:

1. Pages on movies that were released within a short time frame from June 03, 2007. For example, the eighth and twenty-fifth ranked Hubs are pages on movies released on June 01, 2007. There are nine Hubs of this category.

2. Pages on movies whose DVD were released within a short time interval around June 03, 2007. For example, the fifteenth and seventeenth ranked Hubs are pages on movies whose DVD released on June 05, 2007 even though the original movie was released long ago (in 1966) in the case of the seventeenth ranked Hub. There are three Hubs of this type.

3. Pages on TV series whose DVD of a particular season were released within a short time frame from June 03, 2007. For example, the fourteenth, sixteenth and eighteenth ranked Hubs are pages on TV series whose DVD of a particular season were released on June 05, 2007. There are six Hubs of this category.

4. Pages that do not show any temporal effect, although they were not among the topmost Hubs in the February 2007 analysis. For example, the topmost Hub is the homepage of the IMDb web site which is clearly free from any temporal effects. We found a total of five Hubs of this type.

By locating the pages that appear as Authorities on only one of the two dates, as listed in Table 4.6, a closer inspection of their contents reveal nothing of a time-dependent nature. We deduce that, opposite to Hubs, Authorities seem to be very robust to temporal influence.

4.3 Core Communities in a neighborhood graph

Another analysis that we performed is the identification of core communities in the neighborhood graphs. We can consider the pages that consistently appear in the lists of the topmost c Authorities (respectively Hubs) in all neighborhood graphs $\mathcal{N}_D(v)$ constructed at different times for the same web page v as the core Authorities (respectively core Hubs) for the page in question.

4.3.1 Experimental Results

4.3.1.1 Example 1 – $\mathcal{N}_2(v)$ for v = www.cricinfo.com

Table 4.7 gives a list of the core Hubs along with their average rank in $\mathcal{N}_2(v)$ with v= www.cricinfo.com for three different dates (May 31, 2006, January 31, 2007, and March 21, 2007). The two pages that appear in all three lists of top $c = 25$ Hubs are somewhat related – they are the 'Feedback' and 'Help and Feedback' pages of the web site.

There are four pages that appear in each of the three lists of the topmost $c = 25$ Authorities – in Table 4.8 we present a list of the core Authorities along with their average rank $\mathcal{N}_2(v)$ for three different dates (May 31, 2006, January 31, 2007, and March 21, 2007).

The most authoritative core Authority is the page of the Wisden group, followed by the cricshop page, the homepage of the Cricinfo website and the page on the CricInfo magazine. The pages in this list are informative pages on cricket.

4.3.1.2 Example 2 – $\mathcal{N}_2(v)$ for v = www.imdb.com

In the example of www.imdb.com described in Table 4.5 and Table 4.6 we did not find any overlap between the two lists of top $c = 25$ Hubs on the two different dates (February 01, 2007, and June 03, 2007). That implies the neighborhood graph $\mathcal{N}_2(v)$ with v=www.imdb.com is sufficiently dynamic that the lists of the topmost Hubs have nothing in common.

On the other hand, we found the list of the topmost $c = 25$ Authorities to be very similar on the two different dates (February 01, 2007, and June 03, 2007). Table 4.9 presents the core Authorities for $\mathcal{N}_2(v)$ with v = www.imdb.com, $c = 25$.

It is quite interesting to observe that almost all (more precisely, 23) of the topmost

Hubs appear in both of the lists. The topmost core Authority is the web site of Amazon.com, which is followed by the homepage of IMDb web site, Movie Trailers and so on, all of which are informative and resourceful web pages on the topic of movies.

From the analysis of temporal evolution of communities and the extraction of core communities in the neighborhood structure of a particular web page v, we may distinguish two types of neighborhood structure:

1. Pages like www.cricinfo.com for which $\mathcal{N}_D(v)$ shows high temporal effect on the list of Authorities and Hubs and there exists a prominent core community in $\mathcal{N}_D(v)$. It is noteworthy that Hubs are more affected temporally than are Authorities.

2. Pages like www.imdb.com for which $\mathcal{N}_D(v)$ behaves differently for Authorities and Hubs. In this example, the list of topmost Hubs is very temporal in nature whereas the list of topmost Authorities tends to be static. As a result there exists a core community comprised of only the robust list of Authorities in $\mathcal{N}_D(v)$.

From the analysis of core Authorities and Hubs we can extract the core community comprised of the densely linked structure among the core Hubs and the core Authorities. This core community exists irrespective of the temporal, short-term effects. The identification of core communities in the neighborhood structure of a web page v may be useful to get an enhanced, authoritative set of web pages related to v.

Rank	February 01, 2007	June 03, 2007
1	Amazon.com Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more	Amazon.com Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more
2	Movie Trailers	The Internet Movie Database
3	The Internet Movie Database	IMDb – Charts
4	IMDb – Advertising	Game Base!
5	IMDb – Boards Main Boards	Movie & TV News @ IMDb.com – Main Page
6	IMDb – Privacy Notice	IMDb – Now Playing in the US
7	IMDb – Charts	Movie Trailers
8	Game Base!	Cinema Showtimes
9	Movie & TV News @ IMDb.com – Main Page	IMDb Index
10	IMDb – Now Playing in the US	IMDb – Advertising
11	Cinema Showtimes	IMDb – Boards Main Boards
12	IMDb – Help	IMDb – Help
13	How do I join the IMDb team?	How do I join the IMDb team?
14	Register at IMDb.com	IMDb – Privacy Notice
15	Search the Internet Movie Database	Register at IMDb.com
16	Internet Movie Database Licensing	Search the Internet Movie Database
17	IMDb – Top 250	Internet Movie Database Licensing
18	IMDb – IMDbTV	IMDb – Top 250
19	My Movies	IMDb – IMDbTV
20	IMDbPro.com Free Trial Signup	My Movies
21	DVD New Releases	IMDbPro.com Free Trial Signup
22	Search Tips	IMDb – Resume Welcome
23	Internet Movie Database – Browse IMDb	Search Tips
24	The Internet Movie Database – Photo Galleries	Internet Movie Database – Browse IMDb
25	Independent Film at IMDb	The Internet Movie Database – Photo Galleries

Table 4.6: Top twenty five Authorities for $\mathcal{N}_2(v)$ with v = www.imdb.com for two different dates.

Average Rank	**Core Hubs**
4.33	Cricinfo – Feedback
6.67	Cricinfo – Help and Feedback

Table 4.7: The Core Hubs for $\mathcal{N}_2(v)$ with v = www.cricinfo.com, $c = 25$.

Average Rank	**Core Authorities**
2.33	The Wisden Group
5	Cricshop.com
7	Cricinfo.com – The Home of Cricket
11.33	Cricinfo – Cricinfo Magazine

Table 4.8: The Core Authorities for $\mathcal{N}_2(v)$ with v = www.cricinfo.com, $c = 25$.

Average Rank	Core Authorities
1	Amazon.com Online Shopping for Electronics, Apparel, Computers, Books, DVDs & more
2.5	The Internet Movie Database
4.5	Movie Trailers
5	IMDb – Charts
6	Game Base!
7	IMDb – Advertising
7	Movie & TV News @ IMDb.com – Main Page
8	IMDb – Boards Main Boards
8	IMDb – Now Playing in the US
9.5	Cinema Showtimes
10	IMDb – Privacy Notice
12	IMDb – Help
13	How do I join the IMDb team?
14.5	Register at IMDb.com
15.5	Search the Internet Movie Database
16.5	Internet Movie Database Licensing
17.5	IMDb – Top 250
18.5	IMDb – IMDbTV
19.5	My Movies
20.5	IMDbPro.com Free Trial Signup
22.5	Search Tips
23.5	Internet Movie Database – Browse IMDb
24.5	The Internet Movie Database – Photo Galleries

Table 4.9: The Core Authorities for $\mathcal{N}_2(v)$ with $v =$ www.imdb.com, $c = 25$.

Chapter 5

Iterative Cycle Contraction

This chapter describes the Iterative Cycle Contraction algorithm to extract a hierarchy of Web communities. At first we present the idea of the algorithm by presenting two hypotheses regarding relationship between two web pages. Then we present our algorithm and experimental results of our algorithm on some focused subgraphs of $\mathcal{W}$. Some example communities in the first two iterations are presented to illustrate the hierarchical structure of communities. Then we describe the similarity measures between web pages as well as between web communities in the experimental data. Finally, we present a distribution of the size of communities in a particular iteration. The main results in this chapter have been submitted for publication [Nargis and Pike, 2007].

5.1 The Underlying Idea

5.1.1 Definitions

A *directed path* $\overrightarrow{P} = v_0 v_1 \dots v_t$ from v_0 to v_t exists if (v_i, v_{i+1}) is an arc in G for each $i = 0, 1, \dots, t-1$. We say that $\overrightarrow{P}$ is a *bidirectional path* if $\overleftarrow{P} = v_t \dots v_1 v_0$ is also a

directed path in G. The length of this bidirectional path is $2t$, where the *length* of a path is defined to be the number of arcs on the path. A bidirectional path of length 2 is also called a *directed 2-cycle*. A directed 2-cycle with vertices u, v is denoted by $u \leftrightarrow v$.

The *contraction* of a directed 2-cycle $u \leftrightarrow v$ is defined as the operation of replacing u and v by a single vertex whose incident arcs are the arcs other than (u, v) and (v, u) that were incident to u or v.

The concept of directed 2-cycle contraction was introduced in [Graham, 2003], in which the approach involved contracting directed 2-cycles individually. In our case we contract several directed 2-cycles simultaneously. We also focus on a different relationship hypothesis than that considered in [Graham, 2003].

5.1.2 Hypothesis 1

We might propose that two pages u and v have related content if there is a directed path from u to v, and also a directed path from v to u. However, the largest strongly connected component comprises about 28% of all pages in the World Wide Web [Broder et al., 2000]. That means, according to Hypothesis 1 about 28% of the pages in the World Wide Web are related to one another, which is most probably not true. Therefore, we can reject this hypothesis and we have to be more specific in the definition of related pages.

5.1.3 Hypothesis 2

We consequently refine Hypothesis 1 to get Hypothesis 2. In Hypothesis 2 we propose that two pages u and v have related content if u links to v and also v links to u, that is, $u \leftrightarrow v$ is a directed 2-cycle in $\mathcal{W}$. We further extend the notion of relatedness

between two pages by proposing that two vertices u and v are related if and only if there exists a bidirectional path between them. Based on this extension, we define an equivalence relation $\sim$ on the vertex set of a directed graph such that $u \sim v$ if and only if there exists a bidirectional path between u and v.

5.1.4 Outline of the Algorithm

Let G_0 denote an initial graph G. We find every directed 2-cycle in G_0 and contract each directed 2-cycle simultaneously, resulting in a new graph that is denoted by G_1.

Hence, the vertex set of G_1 is the set of equivalence classes in G_0. Each vertex w in G_1 represents a community comprised of the web pages corresponding to the initial vertices contracted to w. There exists an arc in G_1 from one class to another if there is an arc in G_0 from one member vertex of the first class to one member vertex of the second class. If there is more than one arc from one class to another we just keep one arc, so the resulting graph has no multi-edges; hence G_1 is still a simple graph.

In a similar way of applying the directed 2-cycle contraction process, the equivalence classes in G_1 can be identified, and the process repeated to obtain G_2, G_3, and so on until there are no more directed 2-cycles remaining. A community (that is, an equivalence class) in G_i corresponds to a single vertex in G_{i+1} and consists of a collection of vertices in G_i, $i \geq 0$.

5.1.5 Similarity Measure

A similarity measure between two vertices u and v in G_0 can be defined by noting at which iteration u and v are contracted into a single vertex. We label u and v as having an i-th degree content relationship if u and v are grouped together into a single vertex in G_i but not in G_{i-1}. In the case of vertices u and v that are never grouped

into a single community, we define the degree of their relationship to be infinite.

This similarity measure can also be applied to communities. For instance, if A and B are two communities represented by two vertices in G_i, then A and B have a k-th order relationship if they are first merged together in G_{i+k}.

5.2 The Algorithm

Here we present the Iterative Cycle Contraction Algorithm. The function `Contract` presented in Figure 5.1 takes an integer t (the iteration number) and a graph $G_t = (V(G_t), E(G_t))$ as inputs. This function outputs the graph $G_{t+1} = (V(G_{t+1}), E(G_{t+1}))$ and an integer k which denotes the number of directed 2-cycles in G_t.

In Figure 5.2 we give the code in the `Main` function that repeats contracting directed 2-cycles by calling the function `Contract` on G_t until the number of directed 2-cycles is zero.

To briefly review the actions performed by the `Contract` subroutine, first note that $label(v_i)$ denotes the cluster label of the vertex v_i. Initially each vertex is in a cluster by itself, so $label(v_i)$ is initialized to v_i. The function `Contract` checks for the possibility of the existence of a directed 2-cycle $v_i \leftrightarrow v_j$, $1 \leq i < j \leq n$. If there is a directed 2-cycle $v_i \leftrightarrow v_j$, then v_i and v_j have to be contracted to a single vertex, and so the label of vertex v_j is made the same as the label of v_i. Let $L = \{l_1, \ldots, l_k\}$ be the set of distinct labels which remain after all the relabeling is completed. A list S_i of vertices in G_t with label l_i is maintained. The vertex set for the graph G_{t+1} is $V(G_{t+1}) = \{u_1, \ldots, u_k\}$, where each u_i is an equivalence class of vertices of $V(G_t)$. Initially the arc set for G_{t+1} is empty. `Contract` checks, for each ordered pair of equivalence classes (u_i, u_j), $1 \leq i, j \leq k$, whether there exists an arc from a vertex in

```
Function Contract(Integer t) returns Integer
Inputs :  Graph G_t = (V(G_t), E(G_t)),
          Integer t as the iteration number.
Outputs : Graph G_{t+1} = (V(G_{t+1}), E(G_{t+1})),
          Integer cycle_count as the number of directed 2-cycles in G_t.
Begin
   n = |V(G_t)|;
   Let V(G_t) = {v_1, ..., v_n};
   For 1 ≤ i ≤ n Do
      label(v_i) = v_i;
   cycle_count = 0;
   For 1 ≤ i ≤ n Do
      For i + 1 ≤ j ≤ n Do
        If (v_i, v_j) ∈ E(G_t) and (v_j, v_i) ∈ E(G_t) Then
        Begin
          cycle_count = cycle_count + 1;
          label(v_j) = label(v_i);
        End If
      End For
   End For
   Let k be the number of distinct labels in the set {label(v_1), ..., label(v_n)};
   Let L be the set of distinct labels in the set {label(v_1), ..., label(v_n)},
      and denote L = {l_1, ..., l_k};

   Let S_i be the set of vertices with label l_i, 1 ≤ i ≤ k,
      and denote S_i = {S_{i_1}, ..., S_{i_|S_i|}};
   V(G_{t+1}) = {u_1, ..., u_k};
   E(G_{t+1}) = φ;
   For 1 ≤ i ≤ k Do
      For 1 ≤ l ≤ |S_i| Do
        For 1 ≤ j ≤ k Do
          For 1 ≤ m ≤ |S_j| Do
            If (S_{i_l}, S_{j_m}) ∈ E(G_t) Then
              E(G_{t+1}) = E(G_{t+1}) ⋃ {(u_i, u_j)};
          End For
        End For
      End For
   End For
   Return cycle_count;
End
```

Figure 5.1: The `Contract` Function.

```
Function Main
Input :  Graph G_0 = (V(G_0), E(G_0)), the initial graph.
Begin
   t = 0;
   Do
      Count = Contract(t);
      t = t + 1;
   While (Count > 0);
End
```

Figure 5.2: The `Main` Function.

S_i to a vertex in S_j. If so, then it adds the arc (u_i, u_j) to the arc set $E(G_{t+1})$. At last `Contract` returns the number of directed 2-cycles that were in G_t.

5.3 An Example

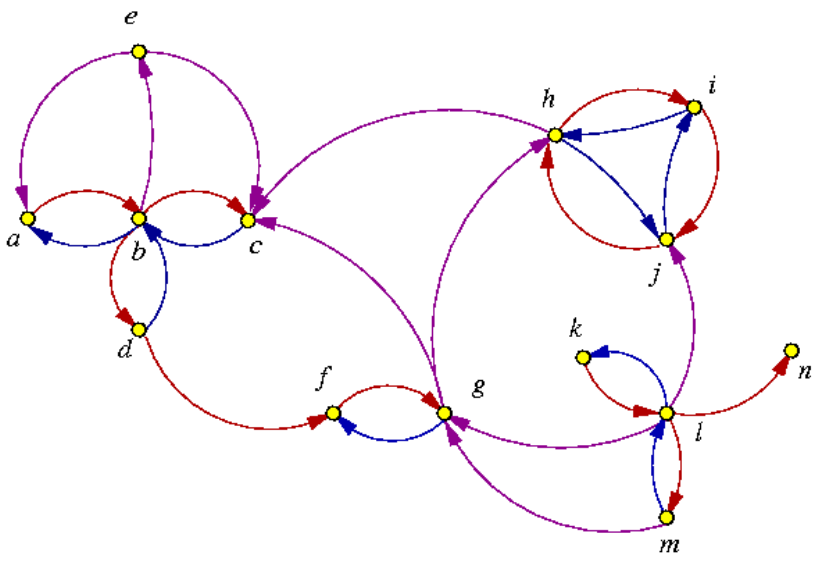

Figure 5.3: An example graph G_0.

Figure 5.3 presents an example of the initial graph G_0. At the first iteration of

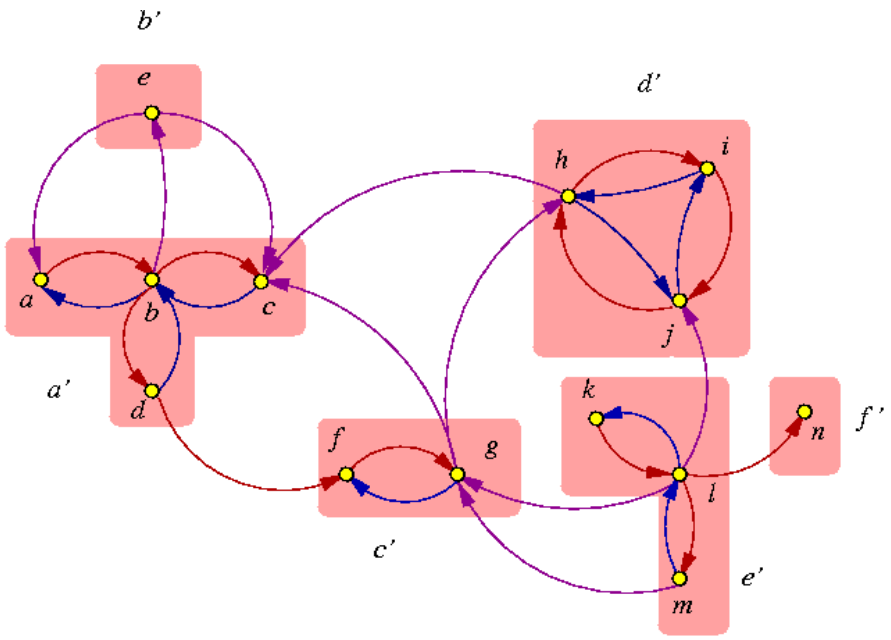

Figure 5.4: The communities in the graph G_0 are each shaded in pink.

our algorithm the communities in G_0 are identified, as depicted in Figure 5.4. For instance, when the directed 2-cycle $a \leftrightarrow b$ is encountered, the label of the vertex b becomes the label of the vertex a, that is, a. Later on the directed 2-cycle $b \leftrightarrow c$ is found, and the label of vertex c is made the same as the label of vertex b. Another directed 2-cycle $b \leftrightarrow d$ is detected, and the label of the vertex d is changed to the label of the vertex b. Now all the vertices a, b, c, and d have the same label a. The set of vertices with label a is therefore $\{a, b, c, d\}$. In a similar way a total of six communities are found in G_0, of which two communities each consist of a single vertex. Each community gets a distinct label in the new graph G_1. The community with vertices $\{a, b, c, d\}$ gets the label a' in G_1. Here the simultaneous contraction of the directed 2-cycles $a \leftrightarrow b$, $b \leftrightarrow c$ and $b \leftrightarrow d$ (shown in Figure 5.4) results in a single vertex a' in Figure 5.5. The vertex e in G_0 (which becomes a community with a single vertex in G_1 with the label of b') has arcs to the vertices a and b of G_0, so only a single arc (b', a') is included in the arc set of the graph G_1. Similarly, there

exists an arc from vertex b to vertex e in G_0, resulting in an arc (a', b') in G_1. As another example, the vertices f and g in G_0 form a community in G_0 which gets the label c' in G_1. There is an arc from vertex d to vertex f in G_0 that results in an arc from vertex a' to vertex c' in G_1. Similarly, there exists an arc from vertex g to vertex c in G_0 that causes an arc from vertex c' to vertex a' to be included in G_1. The

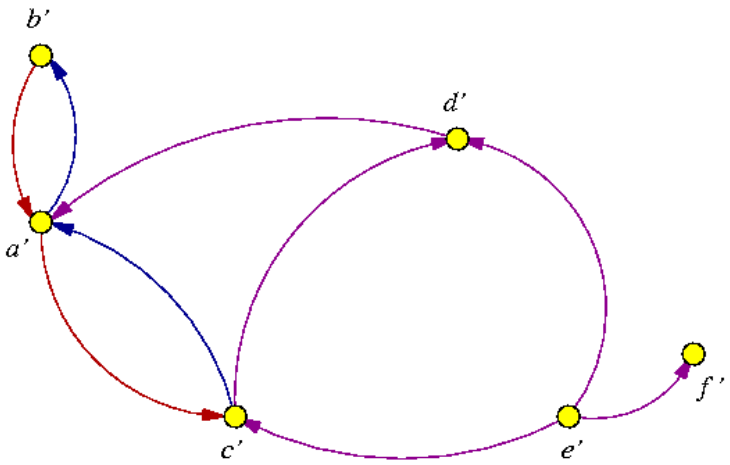

Figure 5.5: The graph G_1 obtained from G_0 in iteration 1.

resulting graph G_1 is presented in Figure 5.5. Figure 5.6 presents the communities

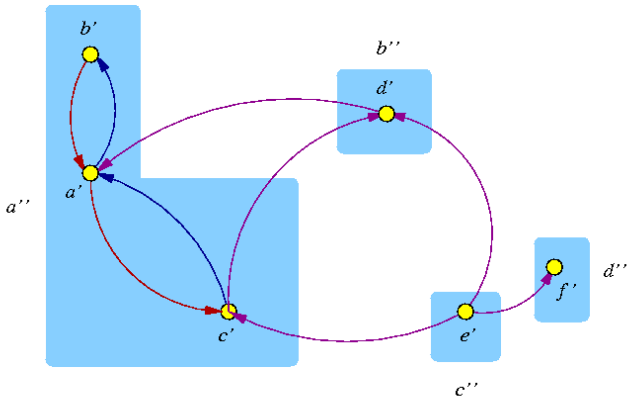

Figure 5.6: The communities in the graph G_1 are each shaded in light blue.

identified in the graph G_1 in iteration 2 of the algorithm. In a similar way of iteration

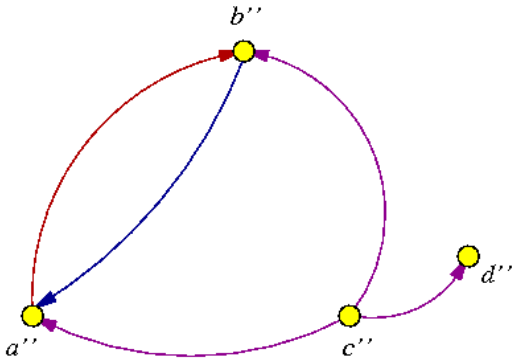

Figure 5.7: The graph G_2 obtained from G_1 in iteration 2.

1, the next graph G_2 is constructed, depicted in Figure 5.7. Figure 5.8 presents the

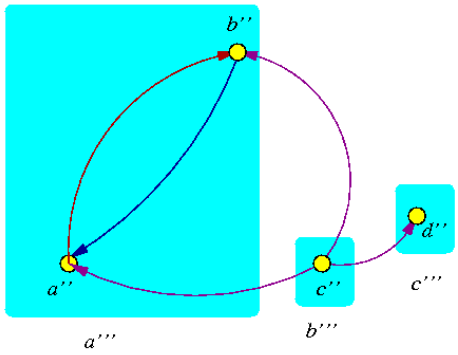

Figure 5.8: The communities in the graph G_2 are each shaded in cyan.

communities identified in the graph G_2 in iteration 3 of the algorithm. Figure 5.9 shows the resulting graph G_3 in iteration 3. At iteration 4, the algorithm finds that there is no more directed 2-cycles in G_3, so the algorithm terminates.

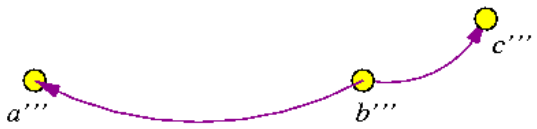

Figure 5.9: The graph G_3 obtained from G_2 in iteration 3.

5.4 Experimental Results

We implemented our algorithm and applied it to some focused subgraphs of the web graph $\mathcal{W}$.[1] We present the result of some initial iterations of the algorithm to illustrate the communities extracted and the hierarchical structure of communities. Then we present some web pages with various similarity measures. Finally we describe the distribution of community size and give a comparison with previous results on community size distribution.

5.4.1 Description of the Input Graph

The neighborhood graph, as introduced in Chapter 3 (and defined in Section 3.1), is a focused subgraph of the web graph $\mathcal{W}$ that models the region surrounding a particular web page by considering the subgraph of $\mathcal{W}$ that is induced by those vertices which are close to a particular given vertex.

5.4.2 Examples of communities detected

As a first example, here we describe the communities extracted as of October 26, 2006 for $G_0 = \mathcal{N}_2(v)$ with $v =$ www.mun.ca, which is the website of Memorial University

[1]The program code implementing the Iterative Cycle contraction algorithm is presented in Appendix G.

of Newfoundland. This graph G_0 has 847 vertices and 12,211 arcs.

5.4.2.1 Example 1 – $\mathcal{N}_2(v)$ for v = www.mun.ca

Table 5.1 shows the communities discovered in the first iteration of the Iterative Cycle Contraction algorithm. All of the communities with at least two web pages are presented in descending order of community size. For each community we state the three member web pages with the highest indegree in G_0.

In the first column we give a name reflecting the apparent content of the three web pages listed for each community. If two or more distinct communities appear to share the same theme then we name the communities with an extension of a Roman number.

The second column records the size of the community where the size of a community is defined by the number of vertices in G_0 that were combined to form a single vertex in G_1. So the size of a community is simply the number of web pages that are in the equivalence class that comprises the community.

For example, the third community is given the name 'Office of the Registrar' since the three pages listed for this community are related to the Office of the Registrar. In total, there are forty four web pages in this community.

A close inspection of the titles of the pages listed in each community reveals that the pages in a particular community share a common theme. Therefore the algorithm is successful in clustering pages according to their theme.

Most (about 94%) of the pages in $\mathcal{N}_2(v)$ for this example are pages in the web site www.mun.ca, yet our algorithm successfully distinguishes between Web communities within this confined web space. Indeed, the algorithm discovered several communities that correspond to various entities that we expect to observe in a university.

It is noteworthy that the 'Weather Office' community is under a different domain

www.weatheroffice.ec.gc.ca than the starting web page www.mun.ca.

Table 5.2 shows the communities discovered in the second iteration for the same graph $G_0 = \mathcal{N}_2(v)$ with $v =$ www.mun.ca, meaning that Table 5.2 presents the communities discovered in G_1. Similar to Table 5.1 we show all the communities with at least two vertices from G_1 and for each community we present its three member communities with the highest indegree in G_1.

The first column again gives a name to each community to represent the theme(s) for the communities and web pages contracted to them. In the second and third columns we present two different measures of the community size for each community C in G_1, starting with the number of vertices in G_1 that formed C. The third column gives the size in terms of the number of web pages in G_0 that were contracted to C.

For example, the 'Alumni' community brings together the 'Alumni I' and 'Alumni II' communities from the first iteration and the web page on an Alumni Newsletter. The communities and pages merged are thematically related in that they are focused on Alumni affairs. This community encompasses three vertices in G_1 and twenty four web pages in G_0.

In some cases, two or more thematic communities are merged to form a more general thematic community. As a good example, the first community listed in Table 5.2 combines the 'Campus Maps', 'People/Departments' and 'Welcome to Memorial' communities from the first iteration. What they share in common is that they all have information mostly sought by newcomers and visitors to campus. Therefore it is natural that they are merged together to form a more general community in the second iteration and it reflected well on our algorithm's ability to bring communities together in a hierarchical fashion.

5.4.2.2 Example 2 – $\mathcal{N}_4(v)$ for v = www.acm.org

As another example, here we describe the communities extracted as of July 07, 2007 for $G_0 = \mathcal{N}_3(v)$ with v = www.acm.org, which is the website of the Association for Computing Machinery. This graph G_0 has 32,936 vertices.

Table 5.3 presents the communities of size at least twenty five discovered in the first iteration of our algorithm. All of the communities with at least twenty five web pages are presented in descending order of community size and for each community we state the three member web pages with the highest indegree in G_0.

We have observed the following types of communities in the first iteration:

1. Communities on various components of the ACM web site. For example, 'ACM Queue I' and 'ACM Queue II' are two communities consisting of web pages inside ACM Queue. Some other instances are 'ACM JobCentre I', 'ACM JobCentre II', 'ACM DL I' to 'ACM DL VI'.

2. Communities on conferences sponsored by ACM. As an example, the 'SC07' community is comprised of web pages on an international conference on high performance computing, networking, storage, and analysis. ACM is the main sponsor of this conference.

3. Communities on companies. For example, the 'NetworkWorld I' and 'NetworkWorld II' communities encompass 193 and 10 web pages in the web site of the company named NetworkWorld.

4. Educational communities like 'Princeton University' and 'Princeton Computer Science' communities.

5. Communities on News. Examples are 'Bloomberg.com' and 'MIT News' communities.

6. Communities on Magazines like 'ComputerWorld', 'ComputerWorld UK', 'ACM TechNews' and 'Risks Digest' communities.

Table 5.4 presents the communities of size at least fifty, discovered in the second iteration for the graph $G_0 = \mathcal{N}_3(v)$ with v = www.acm.org, implying that Table 5.4 presents the communities discovered in G_1. As before, for each community in G_1 we present the three member communities with highest indegree in G_1 and we also present two measures of the size of the community. The communities are presented according to the descending order of the number of vertices in G_1 that were combined to form the community.

For instance, the first community listed in Table 5.4 named 'ACM Online Books and Courses' is comprised of 7125 member communities in G_0, with a total of 7152 web pages. Three of the communities from G_0 that it merges are the 'ACM Online Books and Courses' community and two communities each consisting of a single page on SkillSoft Course in ACM Online Books and Courses. These three communities all discuss aspects of ACM Online Books and Courses, though there are subtle distinctions which were successfully made by the algorithm by putting them in different communities in iteration one.

In the second iteration we observe more than one community on the same topic to be merged to a single community. As an example, the 'eLearn' community in G_1 groups together the 'eLearn' community in G_0 and two communities each with a single web page on eLearn Tutorials and eLearn Opinions.

Table 5.5 presents the communities of size at least twenty five, discovered in the third iteration for the graph $G_0 = \mathcal{N}_3(v)$ with v = www.acm.org, meaning that Table 5.5 presents the communities in G_2. As before, for each community in G_2 we present the three member communities with highest indegree in G_2 and we also

present two measures of the size of the community. The second column gives the size of a community C in G_2 in terms of the number of vertices in G_2 contracted to C and the third column presents the size in terms of the number of vertices in G_0, that is, the number of web pages contracted to C. The communities are presented according to the descending order of the number of vertices in G_2 that were combined to form the community.

For instance, the 'ACM SIG + ACM SRC + ACM TWEB' community is comprised of twenty eight member communities in G_2, with a total of 348 web pages. It merges the 'ACM SIG', 'ACM SRC' and 'ACM SIG + ACM TWEB' communities in G_2, where the first and the third communities share a common theme of ACM SIG (Special Interest Groups).

All these extracted communities illustrate the hierarchical structure of communities.

5.4.3 Mirroring

We observed both mirrored web pages as well as mirrored communities in our experiments. By mirroring of a web page we refer to the existence of more than one web page with the same title and content but different urls.

In case of mirrored web pages that do not link to each other, they may exist in different independent communities as we iterate our algorithm. For example, copies of the page on ACM Digital Library appear in four communities in G_0 named 'ACM DL II', 'ACM DL IV', 'CACM', and 'ACM DL V', listed in Table 5.3.

Typically, mirrors that link to each other eventually collapse into a single community. For instance, the 'Works' community in G_0 listed in Table 5.1 contains two mirror web pages with title 'Works – news'.

Mirrors that do not directly link to each other may also be contracted into a single community at a later iteration than the first iteration. One such merging takes place in the 'Library' community in G_1 listed in Table 5.2. It contains the 'Library' community in G_0, the homepage of Memorial University Libraries and a page on borrowing books from libraries. Looking at the member web pages in the 'Library' community in G_0 listed in Table 5.1 or Table 5.6 we see the homepage of Memorial University Libraries again – there are two mirrored web pages that merge to a single community in second iteration.

An example of a mirrored community is 'HPDC' in G_0 listed in Table 5.3. It has three mirrors, each containing 138 web pages. Accordingly, each of these 138 web pages has three mirrors in G_0.

5.4.4 Similarity Measures

In Section 5.1 we described a similarity measure between two vertices u and v in G_0, that is, a similarity measure between two web pages. We also described a similarity measure between two communities. Here we present some pages (and also some communities) with various similarity measures as extracted by our algorithm.

5.4.4.1 Example 1 – $\mathcal{N}_2(v)$ for v = www.mun.ca

As a first example, Table 5.6 presents the titles of the web pages in a single community named 'Library' in G_0 as of October 26, 2006 for $G_0 = \mathcal{N}_2(v)$ with v = www.mun.ca, meaning that these pages are grouped into a single vertex in G_1. Therefore, all of the pages listed in Table 5.6 have a first degree content relationship and we observe that all of them are on the topic of Memorial University Libraries. This list illustrates that the pages with similarity measure one are focused on a particular theme.

Table 5.8 illustrates the 'Weather Office' community in G_1; it has four G_1 vertices and twenty eight G_0 vertices or web pages. Inside this table, Table 5.7 lists the web pages in the 'Weather Office' community in G_0 – hence all of the web pages listed in Table 5.7 have a first degree content relationship. All these web pages are on the topic of Canadian Weather. This community is merged with three other communities, each comprised of a single web page, to form the 'Weather Office' community in G_1. These three communities are on the topic of Canadian Climate, Environment Canada, and Skywatchers, the latter being an online tool to teach students weather-related topics. It is noteworthy that these topics are related to the theme of weather but they are somewhat different from pure weather-related topics. Each page in Table 5.7 has a similarity measure of two with the pages in any of the remaining three communities.

5.4.4.2 Example 2 – $\mathcal{N}_3(v)$ for v = www.acm.org

As another example, Table 5.9 presents the titles of the web pages in a single community named 'ACM Women I' in $G_0 = \mathcal{N}_3(v)$ with v = www.acm.org. The similarity measure between any two pages in Table 5.9 is one. All of the pages in this community are on the topic of ACM's Committee on Women in Computing.

Table 5.12 presents the 'Ubiquity' community in G_1; it has three G_1 vertices and eight G_0 vertices or web pages. Inside this table, Tables 5.10 and 5.11 present the web pages in two member communities – the 'Ubiquity Forum User Issues' and 'ACM Ubiquity' communities in G_0. All of the web pages listed in Table 5.10 (resp. Table 5.11) have a first degree content relationship. The corresponding two communities are combined with another community consisting of a single web page which is the homepage of Ubiquity Forums to form the 'Ubiquity' community in G_1. The three communities have the common theme of ubiquity yet there are distinguishable themes for each community. Each page in Table 5.10 has a similarity measure of

two with each page in Table 5.11 and also with the single page in 'Ubiquity Forums' community.

Table 5.21 presents the 'SIGCSE + ITiCSE' community in G_2 where $G_0 = \mathcal{N}_3(v)$ with v = www.acm.org. This community consists of two G_2 vertices, six G_1 vertices and thirty three G_0 vertices or web pages. This community merges the 'ITiCSE' and 'SIGCSE' communities in G_1, which are presented in Tables 5.15 and 5.20, respectively.

The 'ITiCSE' community in G_1, presented in Table 5.15, is comprised of two G_1 vertices – the 'ITiCSE' and 'ITiCSE 2008' communities in G_0. Table 5.13 presents the pages in the 'ITiCSE' community in G_0. This community contains the homepages of ITiCSE (Annual Conference on Innovation and Technology in Computer Science Education) 2007 and ITiCSE 2008. Table 5.14 presents the pages in the 'ITiCSE 2008' community in G_0.

The 'SIGCSE' community in G_1 is comprised of four G_1 vertices – the 'SIGCSE I', 'SIGCSE 2008', 'SIGCSE II', and 'SIGCSE III' communities in G_0. Table 5.16 presents the pages in the 'SIGCSE I' community in G_0. This community has three G_0 vertices or web pages on ACM SIGCSE (Special Interest Group on Computer Science Education). The pages in 'SIGCSE 2008' community in G_0 presented in Table 5.17 are on the Technical Symposium on Computer Science Education to be held in 2008, which is the flagship conference of the ACM SIGCSE community. The 'SIGCSE II' (resp. 'SIGCSE III') community in G_0 presented in Table 5.18 (resp. Table 5.18) contains eight pages on SIGCSE.

There exists a first degree content relationship between any pair of web pages listed in any community in G_0 (e.g. 'SIGCSE I', 'SIGCSE 2008' communities in G_0 presented in Tables 5.16 and 5.17, respectively). Each page in the 'SIGCSE 2008' community in G_0 has a similarity measure of two with each of the single pages in

the 'SIGCSE I', 'SIGCSE II' and 'SIGCSE III' communities in G_0. Similarly, each page in the 'ITiCSE' community in G_0 (listed in Table 5.13) has a second degree content relationship with each page in the 'ITiCSE 2008' community in G_0 (listed in Table 5.14). Each page in the 'ITiCSE' community in G_1 (listed in Table 5.15) has a similarity measure of three with each of the pages in the 'SIGCSE' community in G_1 (listed in Table 5.20).

The similarity measure between any two of the 'SIGCSE I', 'SIGCSE 2008', 'SIGCSE II' and 'SIGCSE III' communities in G_0 is one since they are merged into a single community 'SIGCSE' in G_1. Similarly, the 'ITiCSE' and 'SIGCSE' communities in G_1 are contracted into a single community in G_2, so there exists a first degree content relationship between them. The relation between these two communities is that the ITiCSE Conference is organized by ACM SIGCSE. The 'ITiCSE' and 'SIGCSE I' communities in G_0 are first grouped together into a single community 'SIGCSE + ITiCSE' in G_2 implying that the similarity measure between these two communities is two.

5.4.5 Distribution of the size of communities

Broder et al. [Broder et al., 2000] reported that the distribution of the size of the strongly connected components in $\mathcal{W}$ follows a power law; they measured the power law exponent to be around 2.54. Donato et al. [Donato et al., 2004] performed some experiments on a Web crawl of about 200 million web pages and 1.4 billion hyperlinks and also observed the power law phenomenon in the distribution of the size of strongly connected components, but this time with an exponent of 2.07.

We have performed a similar analysis on the statistical distribution on the size of communities extracted by our algorithm in a particular iteration. As an example,

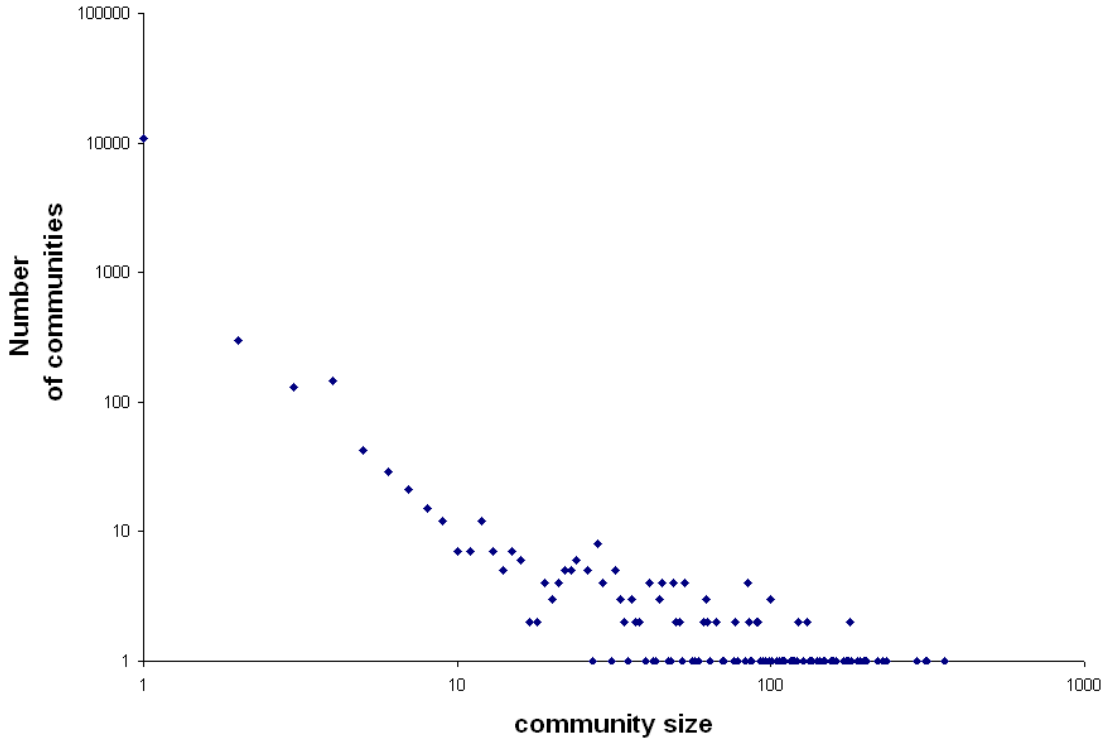

Figure 5.10: The distribution of the size of communities extracted in the first iteration on the graph $\mathcal{N}_2(v)$ for $v =$ www.wikipedia.org.

Figure 5.10 shows the distribution of the size of communities extracted in the first iteration on the graph $\mathcal{N}_2(v)$ with v = www.wikipedia.org ($\mathcal{N}_2(v)$ has 27,795 web pages). From Figure 5.10 we see that the distribution of the size of communities approximately follows a power law. We have performed linear regression with best power law fit to get an approximation of the power law exponent; for Figure 5.10 the exponent is 1.02.

Table 5.1: Communities in $G_0 = \mathcal{N}_2(v)$ with $v =$ www.mun.ca.

Community Name	**Size**	**Three web pages with highest indegree**
SWGC I	88	SWGC – Homepage
		SWGC – Site Index
		SWGC – Prospective Students
Campus Maps	49	Campus Maps
		Campus Maps – MUNnel
		Human Resources – Welcome
Office of the Registrar	44	Office of the Registrar – Apply
		Office of the Registrar – A–Z
		Office of the Registrar
Distance Learning	31	Distance Learning – Complete Programs Offered by Distance
		Distance Learning – Student Services
		Distance Learning – Guide to Taking Courses
Library	28	Memorial University Libraries
		Site Index for MUN libraries Site
		Ferriss Hodgett Library
Weather Office	25	Weather – Choose a Province, Territory, City or Marine Symbol
		Weather – Important Notices and Disclaimers
		Weather – Environment Canada's Official Text Forecasts
Human Resources	24	Human Resources – Welcome
		Human Resources – Business Process Review
		Human Resources – Mission Statement
Works	24	Welcome to The Works
		Works – news
		Sea Hawks Season Tickets
Research	22	Research – Research at Memorial
		Research – For media
		Research – Office of Research
Alumni I	21	Alumni and Friends – Giving to Memorial
		Alumni and Friends – Aeroplan Miles Charitable Pooling Program
		Alumni and Friends – Faculty & Staff Campaign
People/Departments	21	People/Departments – A–Z Listing
		Programs & Courses – Undergraduate programs
		People/Departments – Academic departments
Co-op	20	About Co-op

Table 5.1 is continued on the next page

This is the continuation of Table 5.1 from the previous page

Community Name	Size	Three web pages with highest indegree
		Site Map of The Division of Co-operative Education Site
		Computer Industry Internship Option
SAS	20	Student Affairs and Services – Welcome
		Student Affairs and Services – Finances
		Student Affairs and Services – Enhancing Student Life on Campus
Welcome to Memorial	17	MUNdays
		Welcome to Memorial – Services for visitors
		Welcome to Memorial – How we can help
Memorial University	15	Memorial University Homepage
		Memorial University – Privacy
		Memorial University – Feedback
Harlow Campus	15	Harlow Campus – Welcome!
		Life in Harlow
		Harlow Campus – Guestbook
MUNdays I	14	MUNdays – Kindness
		MUNdays – Diversity
		MUNdays – Excellence
MAC	13	Marketing & Communications – Site Map
		Marketing & Communications – Web Content Services
		Marketing & Communications – Privacy
MUNdays II	8	MUNdays
		MUNdays – Site Map
		MUNdays – Ingenuity
News	7	Day two of fall celebrations
		today.mun.ca
		Falling for Grenfell
Marine Institute	7	Marine Institute Homepage
		Marine Institute – Centres
		Marine Institute – News
VP Academic	7	Vice-President (Academic)
		Vice-President (Academic) – Appointments
		Vice-President (Academic) – Associate Vice-President (Academic)
Programs & Courses	4	Programs & Courses – Undergraduate programs
		Office of the Registrar
		Programs & Courses – Post-baccalaureate
Calendar of Events	4	Calendar of Events October 8, 2006 to October 14, 2006

Table 5.1 is continued on the next page

This is the continuation of Table 5.1 from the previous page

Community Name	Size	Three web pages with highest indegree
		Calendar of Events October 22, 2006 to October 28, 2006
		Calendar of Events October 15, 2006 to October 21, 2006
Admission	4	Admission-Readmission to the University (Undergraduate)
		English Language Proficiency Requirements
		Transfer Credit
Nursing	2	School of Nursing – Bachelor of Nursing (Post-RN)
		School of Nursing – Bachelor of Nursing Collaborative Program
FAS	2	FAS Travel Guidelines – General
		FAS Risk Management – Insurance Coverage
Weather Office NL	2	Newfoundland & Labrador – Weather Conditions & Forecast
		Terre-Neuve-et-Labrador – Prvisions et conditions par endroits
Research Chair	2	Research – Canada research chairs
		Research – Staff directory
SWGC II	2	SWGC – Schedule of Events
		SWGC – Grenfell Grapevine
Alumni II	2	Alumni and Friends – Welcome Alumni & Friends
		Online Giving

Table 5.2: Communities in G_1, where $G_0 = \mathcal{N}_2(v)$ with v = www.mun.ca.

Community Name	**Number of vertices from**		**Three member communities with highest indegree**
	G_1	G_0	
Campus + People/Departments	34	153	Campus Maps community
			People/Departments community
			Welcome to Memorial community
SWGC + Registrar	18	174	SWGC I community
			Office of the registrar community
			Academic Advising Centre – Welcome
SAS + Research	7	55	SAS community
			Research community
			Student Affairs and Services – Feedback
Library	4	31	Library community
			Memorial University Libraries Homepage
			Library - Borrowing
Weather Office	4	28	Weather Office community
			Canadian Climate
			Environment Canada's Green Lane (French)
Alumni	3	24	Alumni I community
			Alumni II community
			Luminus Express – MUN Alumni Newsletter
Co-op	3	22	Career Development & Experiential Learning – Building Career
			Co-op community
			The Division of Co-operative Education Feedback
SWGC II	3	4	SWGC II community
			SWGC – Schedule of Events
			SWGC – Pictures
Human Resources	2	25	Human Resources community
			Human Resources – HR News and Notices
Memorial University + MI	2	16	Memorial University community
			MI - Applied Research
News	2	9	News community
			Calendar of Events October 29, 2006 to November 4, 2006

Table 5.3: Communities in $G_0 = \mathcal{N}_3(v)$ with v = www.acm.org.

Community Name	**Size**	**Three member web pages with highest indegree**
ACM Guide Author Index	556	ACM Guide Author Index – A
		ACM Guide Author Index – A
		ACM Guide Author Index – A
ACM Queue I	481	ACM Queue – Hardware & Systems
		ACM Queue – Data Management
		ACM Queue – Recent Curmudgeon
ACM Queue II	344	ACM Queue – Recent Features
		ACM Queue – Data Management
		ACM Queue – Recent Curmudgeon
USACM I	294	USACM Technology Policy Weblog
		USACM Technology Policy Weblog – Homepage
		USACM Technology Policy Weblog – Homepage
ACM Guide Index	253	ACM Guide Proper Nouns Index
		ACM Guide Keywords Index
		ACM Guide Reviewer Index
ACM DL I	252	ACM DL Proceedings
		ACM DL Home – Proceedings – SIGCSE
		ACM DL Home – Proceedings – KDD
Hack	212	Undetected Counter Strike Hacks
		Undetected Hacks
		WoW Hacks – World of Warcraft Hacks – Counter strike Hacks
Apartments	211	Apartments for rent – apartments rental
		Apartments TX
		Apartments CA
NetworkWorld I	193	Welcome To NetworkWorld.com
		Network Security Special Reports Library – Network World
		Spam, Phishing, Pharming and Spyware – Network World
Web Directory	192	AllinfoDir Web Directory
		AllinfoDir Web Directory – Partners
		Elegant Directory
ComFi	153	ComFi –Communications Fidelity – Homepage
		India Calling Cards : Lowest Rates, Variety of International Phone Cards
		Canada Calling Cards : Best rates to Greece - Athens Wide selection of Calling Cards

Table 5.3 is continued on the next page

This is the continuation of Table 5.3 from the previous page

Community Name	Size	Three member web pages with highest indegree
ComputerWorld	150	ComputerWorld – Hacking Firefox The secrets of about config
		ComputerWorld – McKesson harnesses HP's SOA for quality assurance
		ComputerWorld – Is InfiniBand finally ready for prime time
Game Tickets	139	MLB Tickets – Cheap MLB Baseball Tickets
		Oakland A's Tickets
		Fenway Park Tickets – Boston Red Sox Fenway Tickets
HPDC (3 mirrors)	138	ACM Guide to Computing Literature
		HPDC HPDC (High Performance Distributed Computing) '07
		HPDC SOCP (Service-oriented computing performance) '07
Riander Blog	127	Riander Blog
		Riander Blog – User Experience
		Riander Blog – User Experience
ACM Jobcentre I	122	ACM JobCentre – Browse Jobs
		ACM JobCentre – Browse Jobs
		Job Search Results[ACM Job Centre]
ACM TechNews	94	ACM TechNews – Past Issues
		ACM TechNews – Current Issue
		ACM TechNews – Past Issues – Dec. 01, 1999
Adobe	92	Adobe – Reader Download
		Adobe – Reader Download
		Adobe – Reader Adobe Reader for Palm OS
abc7.com	91	abc7.com Hackers Hired to Crack Calif. Electronic Voting Machines
		'ABC7 Listens' Town Hall Meetings
		Subscribe to the abc7.com Newsletter
ACM DL II	84	ACM Guide to Computing Literature
		ACM Digital Library
		ACM SIGAda Ada Letters – TOC
AICPS + ACL	82	AICPS ICML '07
		Annual Meeting of the ACL
		ACM SIGARCH Volume 35, Issue 2
Risks Digest	80	RISKS-LIST RISKS-FORUM Digest
		The Risks Digest – Volume 24 Issue 75
		The Risks Digest – Index to Volume 24
Browse8.com	77	Browse8.com web directory and website promotion

Table 5.3 is continued on the next page

This is the continuation of Table 5.3 from the previous page

Community Name	Size	Three member web pages with highest indegree
		Arts Browse8 web directory and website promotion
		Teen & Kids Browse8 web directory and website promotion
NetworkWorld II	10	NetworkWorld.com – Homepage
		NetworkWorld.com – Enterprise opinions
		NetworkWorld.com – Networking Careers
SHOP.COM	72	Shop Clothing, Shoes, Handbags, Makeup, Fragrances, Furniture, Garden, Gifts – Online Shopping at SHOP.COM
		Online Shopping Departments – SHOP.COM
		Site Map – SHOP.COM
ACM SIG I	69	ACM SIG Conference Manual – Table of Contents
		ACM SIG Volunteer Information
		ACM ACM SIG Officer Manual – Table of Contents
Chronicle.com	65	Wired Campus Blog – Chronicle.com
		The Chronicle of Higher Education Information Technology
		The Chronicle of Higher Education Site map
ACM SIGGRAPH	63	ACM SIGGRAPH News siggraph.org
		ACM SIGGRAPH News siggraph.org
		ACM SIGGRAPH News siggraph.org
Apple QuickTime	61	Apple – QuickTime – Download
		Apple – Quicktime – Quicktime Player
		Apple – QuickTime – QuickTime Player – Frequently Asked Questions
ACM DL III	58	ACM DL Newsletters
		ACM SIGCHI Bulletin
		ACM SIGPLAN Notices
ACM DL IV	58	ACM Digital Library
		FAQ – Using the Digital Library
		HPDC HPDC '07
GovTrack	57	GovTrack – Tracked Events Active Legislation
		GovTrack – Tracked Events Introduced Legislation
		Legislation
CACM	56	CACM (Communications of the ACM) Archive
		ACM Digital Library
		HPDC HPDC '07
Ways & Means	56	Committee on Ways & Means – U.S. House of Representatives

Table 5.3 is continued on the next page

This is the continuation of Table 5.3 from the previous page

Community Name	Size	Three member web pages with highest indegree
		Hearing Archives Committee on Ways & Means – U.S. House of Representatives
		Hearing Archives Committee on Ways & Means – U.S. House of Representatives
Computer Weekly	54	Computer Weekly – Life Style
		E-Business Solution News from Computer Weekly
		Business Intelligence Products News from Computer Weekly
ACM DL V	54	ACM Digital Library
		ACM Guide to Computing Literature
		HPDC HPDC '07
eLearn	54	eLearn – Past Articles
		eLearn – Best Practices
		eLearn – Past Articles
Ars Technica	51	Ars Technica – Future avatars will be adept at manipulating human response
		Ars Technica – Homepage
		Ars Technica – US government prepares for cyber war games
ACM Queue III	50	ACM Queue – QueueCasts
		ACM Queue – Data Management
		ACM Queue – Hardware & Systems
CNET	46	Technology news – CNET News.com
		Technology news – CNET News.com
		Technology news – CNET News.com
ACM SIG II	44	SIGs – Programming Languages
		ACM DL SIGs
		SIGs – Computer Graphics
SC07	43	SC07 – Broader Engagement
		SC07 – Broader Engagement Participation Grants
		SC07 – Posters
CSTA	41	CSTA Computer Science Teachers Association – Classroom Careers Resources
		CSTA Computer Science Teachers Association – Teacher Workshops
		CSTA Computer Science Teachers Association – Periodicals
Credit World Australia	41	Credit World Australia – Homepage
		Sitemap – Credit World Australia
		Leading Credit Cards Australia, Compare and Apply Online

Table 5.3 is continued on the next page

This is the continuation of Table 5.3 from the previous page

Community Name	Size	Three member web pages with highest indegree
SIGIR'07	38	SIGIR'07 - Amsterdam – Homepage
		SIGIR'07 - Amsterdam – Printer Friendly
		SIGIR'07 – Industry Event
Princeton University	37	Princeton University – Award honors Peh's research and outreach
		Princeton University – Campus Announcements
		Princeton University – Campus Announcements
SIGGRAPH 2007 I	37	SIGGRAPH 2007 – The 34th International Conference and Exhibition on Computer Graphics and Interactive Techniques
		SIGGRAPH 2007 – For Presenters – Computer Animation Festival
		SIGGRAPH 2007 – For Presenters – Courses
ConceptLaboratory	35	ConceptLaboratory – Quick Updater osCommerce Module Edit osCommerce Databases Fast
		ConceptLaboratory – Free osCommerce Contributions & Free osCommerce Downloads
		ConceptLaboratory – Affiliate Marketing & osCommerce Affiliates
USACM II	32	U.S. Public Policy Committee of the Association for Computing Machinery
		ACM MemberNet Current Issue
		USACM – U.S. Public Policy Committee of the Association for Computing Machinery
About ACM	32	About ACM Association for Computing Machinery – ACM Policies & Procedures
		About ACM Association for Computing Machinery – Awards
		About ACM Association for Computing Machinery – PressRoom
ACM CRC (9 mirrors)	32	ACM – Career Resource Center – Articles & Columns – Job Search
		ACM – Career Resource Center – Articles & Columns – Professional Development
		ACM – Career Resource Center – Articles & Columns – CS/IS/MIS Careers
Internet Caucus Advisory Committee	32	ICAC Digital Rights Management Panel 2003
		Members of the Internet Caucus Advisory Committee
		Congressional Internet Caucus Advisory Committee Event Listings

Table 5.3 is continued on the next page

This is the continuation of Table 5.3 from the previous page

Community Name	Size	Three member web pages with highest indegree
ComputerWorld UK	30	Computerworld UK – The Voice of IT Management
		Technology – Computerworld UK
		News – Computerworld UK
Richard Tapia 2005	30	Richard Tapia 2005 - Celebration of Diversity in Computing – Supporters
		Richard Tapia 2005 - Celebration of Diversity in Computing – Conferences
		Richard Tapia 2005 - Celebration of Diversity in Computing – Conferences
THOMAS LOC	29	THOMAS (Library of Congress) – Homepage
		THOMAS (Library of Congress) – THOMAS
		THOMAS (Library of Congress) – About THOMAS
ACM Online Books and Courses	28	ACM Online Books and Courses – Listing of courses available to ACM Members through SkillSoft
		ACM Online Books and Courses – Safari Books Listing
		ACM – Sitemap
Bloomberg.com	28	Bloomberg.com Bloomberg Markets Magazine
		Bloomberg.com : News
		Bloomberg.com Audio-Video Reports
Princeton Computer Science	28	Industrial Affiliates – Princeton Computer Science
		Faculty – Princeton Computer Science
		Princeton Computer Science
Get21.com	27	Get21.com – Homepage
		Get21.com – SiteMap
		Get21.com – Press Centre
ACM Headquarters Staff Listing	26	ACM Headquarters Staff – S
		ACM Headquarters Staff – A
		ACM Headquarters Staff – B
ACM Volunteers Alphabetical Listing	26	ACM Volunteers – Alphabetical Listing – A
		ACM Volunteers – Alphabetical Listing – H
		ACM Volunteers – Alphabetical Listing – D
SIGGRAPH 2007 II	26	SIGGRAPH 2007 – For Attendees – Registration
		SIGGRAPH 2007 – For Attendees – Co-Located Workshops & Events
		SIGGRAPH 2007 – For Attendees – Special Events
ACM Fellows	25	ACM Fellows
		ACM Fellows – A
		ACM Fellows – B
MIT News	25	MIT News Office News by topic
		MIT News Office 2007 archive

Table 5.3 is continued on the next page

This is the continuation of Table 5.3 from the previous page

Community Name	**Size**	**Three member web pages with highest indegree**
		MIT News Office 2007 archive

Table 5.4: Communities in G_1, where $G_0 = \mathcal{N}_3(v)$ with v = www.acm.org.

Community Name	**Number of vertices from**		**Three member communities with highest indegree in G_1**
	G_1	G_0	
ACM Online Books and Courses	7125	7152	ACM Online Books and Courses community
			ACM Online Books and Courses – SkillSoft Course Detail - Introduction to PL-SQL
			ACM Online Books and Courses – SkillSoft Course Detail - Using PL-SQL with an Oracle Server
ACM Guide Index	1287	2094	ACM Guide Index community
			ACM Guide Author Index community
			Volume 14, Issue 4 , In this issue
AICPS + ACL + ACM SIG	1246	1370	AICPS+ACL community
			ACM SIG II community
			Archive, Franz Alt interview January
CACM	606	609	CACM Archive community
			CACM Volume 50, Issue 5 community
			CACM Volume 50, Issue 5 – COLUMN Inside risks
eLearn	164	218	eLearn community
			eLearn – In Depth Tutorials
			eLearn – Opinions
ACM SIG	83	110	ACM SIG III community
			ACM TWEB community
			ACM SIGSAC – ACM Special Interest Group on Security, Audit and Control (SIGSAC)
ACM DL	81	164	ACM DL II community
			Results (page 1) author : Ewa Deelman
			Results (page 1) author : Carlo Mastroianni
ACM Women + ACM Chapter	78	118	ACM Women community
			ACM Student Chapter community
			ACM Professional Chapter community
ACM SIG + ACM Publications	64	160	ACM SIG IV community
			ACM Publications community

Table 5.4 is continued on the next page

This is the continuation of Table 5.4 from the previous page

Community Name	**Number of vertices from**		**Three member communities with highest indegree in G_1**
	G_1	G_0	
			ACM SIG Volunteer Information community
Riander Blog	62	193	Riander Blog community
			Riander Blog
			Riander Blog
ACM Ubiquity + ACM Key People	52	161	ACM Ubiquity community
			About ACM : Key People community
			ACM Publications Board

Table 5.5: Communities in G_2, where $G_0 = \mathcal{N}_3(v)$ with v = www.acm.org.

Community Name	**Number of vertices from**		**Three member communities with highest indegree in G_2**
	G_2	G_0	
ACM Fellows + ACM Awards + ACM CRC + ACM Ubiquity	665	755	ACM + ACM Member community
			ACM CRC – ACM Recommends community
			ACM Ubiquity community
ACM Publications + ACM Membership + ACM Libraries	29	92	ACM Publications + ACM Membership community
			ACM Libraries – La Carte Subscriptions for Libraries Journals and Magazines
			ACM Digital Library SIG Package
ACM SIG + ACM SRC + ACM TWEB	28	348	ACM SIG community
			ACM SRC community
			ACM SIG + ACM TWEB community
ACM Key People + ACM Membership + ACM Ubiquity community	26	204	ACM Key People + ACM Membership + ACM Ubiquity community
			ACM Membership community
			ACM Association for Computing Machinery, the world's first educational and scientific computing society – Homepage

Web Page Title
Memorial University Libraries – Homepage
Memorial University Libraries – Site Index
Ferriss Hodgett Library
Memorial University Libraries – Services for Memorial Alumni
Ferriss Hodgett Library Hours
Ferriss Hodgett Library Chat Services
News for the Memorial University Libraries
Memorial University Libraries – For Distance Students
Memorial University Libarries – We Can Help
Welcome to Memorial University Libraries
Make a Request – Memorial University Libraries Users
Memorial University Libraries – Guides and Instruction
Memorial University Libraries – Contact Us
Memorial University Libraries – Hours
Memorial University Libraries – Libraries and Archives
Memorial University Libraries Chat Service
RSS feeds at Memorial University Libraries
Memorial University Libraries – Search
Memorial University Libraries – Ask a Librarian
Memorial University Libraries – Internet Resources
Memorial University Libraries – Frequently asked questions
Memorial University Libraries – Welcome to SingleSearch
Memorial University Libraries – RefWorks

Table 5.6: The web pages in 'Library' community in G_0 for $G_0 = \mathcal{N}_2(v)$ with $v =$ www.mun.ca.

Community Name in G_0	**No. of web pages in G_0**	**Web Page Title**
Weather Office	25	Weather – Choose a Province, Territory, City or Marine Symbol Weather – Important Notices and Disclaimers Weather – Value-added Services for Business and Special Applications Weather – Weather Charts Weather – Environment Canada's Official Text Forecasts Weather – Hurricane Current Conditions Weather – Lightning Activity Weather – About Us Weather – Comments & Suggestions Weather – Frequently Asked Questions, Environment Canada's Weather site Help - Environment Canada's Weather site Weather – Information and Publications Weather – Links What's New, Environment Canada's Weather site Weather – Your Environment Weather – Canadian Marine Weather Canada - Radar Imagery - Environment Canada Environment Canada Seasonal Forecast Weather – Satellite Images (data courtesy of NOAA) Public Weather Warnings for Canada Meteorological Service of Canada – Air Quality Services Meteorological Service of Canada – Jobs Environment Canada - Search Form Contents - Meteorological Service of Canada (MSC) Environment Canada - Environment Canada's Home Page
Canadian Climate	1	Canadian Climate
Environment Canada's Green Lane	1	Environment Canada's Green Lane
Skywatchers	1	Skywatchers

Table 5.7: The web pages in 'Weather Office' community in G_0.

Table 5.8: The web pages in 'Weather Office' community in G_1 for $G_0 = \mathcal{N}_2(v)$ with v = www.mun.ca.

Web Page Title
ACM-Women (ACM's Committee on Women in Computing) – Homepage
ACM-Women – Online Resources
ACM-Women – News List Page
ACM-Women – Small and Large Regional Conferences Project
ACM-Women – Profiles
ACM-Women – International
ACM-Women – Pipelining
ACM-Women – Partners
ACM-Women – Chapter Requirements
ACM-Women – News and Announcements
ACM-Women – Scholarships for Attendance at Research Conferences
ACM-Women – Policy
ACM-Women – Article: The Incredible Shrinking Pipeline

Table 5.9: The web pages in 'ACM Women I' community in G_0 for $G_0 = \mathcal{N}_3(v)$ with v = www.acm.org.

Community Name in G_0	**Number of web pages in G_0**	**Web Page Title**
Ubiquity Forums	1	Ubiquity Forums
Ubiquity Forums User Issues	6	Ubiquity Forums – Your Customization Settings Ubiquity Forums – Help On Using Ubiquity Forums Ubiquity Forums – Login To The Forums Ubiquity Forums – Your Profile Ubiquity Forums – Search The Forums
ACM Ubiquity	2	ACM Ubiquity Website ACM Ubiquity

Table 5.10: The web pages in 'Ubiquity Forums User Issues' community in G_0.

Table 5.11: The web pages in 'ACM Ubiquity' community in G_0.

Table 5.12: The web pages in 'Ubiquity' community in G_1 for $G_0 = \mathcal{N}_3(v)$ with $v =$ www.acm.org.

Community Name in G_1	No. of vertices in G_1	Web Page Title
ITiCSE	2	

Community Name in G_0	No. of vertices in G_0	Web Page Title
ITiCSE	2	Home page ITiCSE 2007 ITiCSE 2008

Table 5.13: The web pages in 'ITiCSE' community in G_0.

Community Name in G_0	No. of vertices in G_0	Web Page Title
ITiCSE 2008	4	Call for Participation – ITiCSE 2008 Important Dates – ITiCSE 2008 ITiCSE 2008 Submission – ITiCSE 2008

Table 5.14: The web pages in 'ITiCSE 2008' community in G_0.

Table 5.15: The web pages in 'ITiCSE' community in G_1.

SIGCSE	4	

Community Name in G_0	No. of vertices in G_0	Web Page Title
SIGCSE I	3	SIGCSE Home (2 mirrors) SIGCSE 2008

Table 5.16: The web pages in 'SIGCSE I' community in G_0.

Community Name in G_0	No. of vertices in G_0	Web Page Title
SIGCSE 2008	8	SIGCSE 2008 – Attendees SIGCSE 2008 – Authors SIGCSE 2008 – Contact SIGCSE 2008 – Exhibitors Page SIGCSE 2008 – Homepage SIGCSE 2008 – Reviewer Information SIGCSE 2008 – Student Information SIGCSE 2008 – Updates

Table 5.17: The web pages in 'SIGCSE 2008' community in G_0.

Community Name in G_0	No. of vertices in G_0	Web Page Title
SIGCSE II	8	About SIGCSE (2 mirrors) SIGCSE Awards SIGCSE Conferences (2 mirrors) SIGCSE – Contact Us (2 mirrors) SIGCSE Home

Table 5.18: The web pages in 'SIGCSE II' community in G_0.

Community Name in G_0	No. of vertices in G_0	Web Page Title
SIGCSE III	8	SIGCSE Membership (2 mirrors) SIGCSE Publications (2 mirrors) SIGCSE Author Guidelines SIGCSE Site Index SIGCSE Education Links SIGCSE Committee

Table 5.19: The web pages in 'SIGCSE III' community in G_0.

Table 5.20: The web pages in 'SIGCSE' community in G_1.

Table 5.21: The web pages in 'SIGCSE + ITiCSE' community in G_2 for $G_0 = \mathcal{N}_3(v)$ with v = www.acm.org.

Chapter 6

Conclusions

6.1 Summary of the Thesis

The web graph $\mathcal{W}$ contains only link information, but nevertheless it is a massive data structure that is a challenge to analyze and search. The neighborhood graph $\mathcal{N}_D(v)$ is much smaller in size, providing an easier and faster way for analyzing certain local aspects of the web graph.

We have investigated some structural and statistical properties of $\mathcal{N}_D(v)$, giving insight into the localized behavior of $\mathcal{W}$. In particular, we have observed that in many ways $\mathcal{N}_D(v)$ behaves similar to $\mathcal{W}$. Both the indegree and outdegree distribution of a sufficiently large instance of $\mathcal{N}_D(v)$ follow the power law phenomenon.

The analysis of the topmost Hubs and Authorities in $\mathcal{N}_D(v)$ reveal the theme of v which is particularly applicable in the area of similar-page queries. We also performed a temporal analysis on Authorities and Hubs in $\mathcal{N}_D(v)$ reaching to the conclusion that the topmost Authorities and Hubs illustrate the temporal evolution of the communities in the neighborhood structure of the web page in question. In each case, Hubs are more affected temporally than Authorities. We performed the

extraction of core communities in $\mathcal{N}_D(v)$ whose identification may be useful to get an enhanced, authoritative set of web pages related to a particular page v.

We have devised a new algorithm to extract a hierarchy of Web communities based on the idea of iterative cycle contraction. The algorithm is successful at identifying communities and distinguishing communities with varying thematic content in neighborhood graphs. In later iterations, some communities are focused on a particular theme and in some cases, two or more communities on related themes are merged to form a more general thematic community. The distribution of the size of communities in a particular iteration of the algorithm approximately follows a power law.

6.2 Future Research Directions

Below we describe some possible future research directions:

- Analyze the structural and statistical properties of the neighborhood graphs $\mathcal{N}_D(v)$ for $D > 2$, and analyze the effect of D on the properties of $\mathcal{N}_D(v)$.
- Analyze the neighborhood graph $\mathcal{N}_D(V)$ for a collection of web pages V instead of a single web page v. This set V may be related in some theme, for example, they may be the set of web pages under the same domain, or they may be the result of a search query.
- Apply the Iterative Cycle Contraction algorithm to a large Web crawl.
- Improve the Iterative Cycle Contraction algorithm by incorporating a mirror elimination phase.
- Apply the concepts and algorithms to other networked information structures.

Appendix A

Code for Construction of $\mathcal{N}_2(v)$

In Section 3.1 we defined neighborhood graph. In Section 3.2 we presented method of constructing a neighborhood graph. Here we present a sample program to construct $\mathcal{N}_2(v)$, the neighborhood graph of distance two starting with a particular web page v. This program is implemented as a Bash shell script. This program uses the programs listed in Appendix B and C.

```
#######################################
# Author : Isheeta Nargis             #
#
# This program is used to construct the Neighborhood Graph N_2(v)
# of distance 2 starting with vertex v or web page v


#!/bin/bash


# Inputs:
#   1. An URL (with the http://)
#   2. An integer k implying that a maximum of 10*k backward links
#      will be included in the Neighborhood Graph for each vertexh

# To run, type: bash neighbor url k


# The awk, tr, sort, uniq, grep, etc commands are for parsing the
# output of lynx and the MyGoogleAPI.

# collect 10*k backward links of the web page v
# we use search query
# to collect backward links we use "link:url" query in Google Web API
```

```
# collected pages are written into file ‘both’
java com.google.soap.search.MyGoogleAPI search link:$1 $2 |
grep "URL  = \"http" | grep -v "#" |
awk ’// { print $3 }’ | tr -d ’"’ | sort | uniq > both

# collect vertices with distance two from v

  # collect all forward links from v
  # these pages are appended to ‘both’
  lynx -dump $1 | grep "http" | awk ’// {print $2 }’  | grep "http" |
  grep -v "#" | sort | uniq >> both

  # so far we have collected web pages with distance one from v,
  # that is the web pages in layer 1
  # compute the number of web pages collected in ‘both’
  wc -l both | awk ’// {print $1}’ > size_both
  number=$(awk "(NR==1){print};{next}" size_both)
  rm size_both
  echo $number


 # collect vertices with distance two from v

  echo $1 > bothboth

  # while 11
  counter=1
  # for each page u in layer 1
  while [ $counter -lt $((number + 1)) ]; do
      # collect 10*k backward links of u
      java com.google.soap.search.MyGoogleAPI search
      link:$(awk "(NR==$counter){print};{next}" both) $2 |
      grep "URL  = \"http" | grep -v "#" |awk ’// { print $3 }’ |
      tr -d ’"’ >> bothboth
      echo $counter
      counter=$((counter + 1))
      # go to next web page
  done
  # end while 11

  # while 12
  counter=1
```

```
  # for each page u in layer 1
  while [ $counter -lt $((number + 1)) ]; do
     # collect all forward links from u
     lynx -dump $(awk "(NR==$counter){print};{next}" both) |
     grep "http" | awk '// {print $2 }' |
     grep "http" | grep -v "#" | sort | uniq >> bothboth
     echo $counter
     counter=$((counter + 1))
     # go to next web page
  done
  # end while 12

  echo $1 >> both
  sort bothboth >> both
  sort both > bothboth
  uniq bothboth > both # discard duplicate urls



# compute the number of web pages collected
wc -l both | awk '// {print $1}' > size_both
number=$(awk "(NR==1){print};{next}" size_both)
echo number is $number

# adjacent is the file that will store the arcs
# in the subgraph induced by the pages in 'both'
echo "" > adjacent

# while 2
counter=0
# for each page in 'both' do
while [ $counter -lt $((number)) ]; do
  # store the forward links from u in the file 'temp'
  lynx -dump $(awk "(NR==$((counter+1)))){print};{next}" both) |
  grep "http" | awk '// {print $2 }' | grep "http" |
  grep -v "#" | sort | uniq > temp

  #collect the arcs from u in the induced subgraph
  #append them to 'adjacent'
  adjacent.out $counter both temp >> adjacent

  counter=$((counter + 1))
```

```
  # go to next web page
done

# end while 2

#########################################################################
```

Appendix B

Code for Search of Web Pages

In Section 3.2 we described the use of Google's Public Web API services to find the pages that link to a particular web page v. Here we present a program to find the web pages for a search query using Google Web APIs. This program is adapted from GoogleWebAPIDemo provided with Google Web API documentation. This program is implemented in JAVA. This program is used by the program listed in Appendix A.

```
/*
           Author: Isheeta Nargis

       This program is used to get the web pages
       for a search query using Google Web APIs.
       This program is adapted from GoogleWebAPIDemo
       provided with Google Web API documentation.

       Inputs:
        1. Query String q.
        2. k - the number of queries to process in each call.

       Output:
        The top ranked 10*k web pages returned by search with
        the query string q in Google Search Engine.

*/

/************************************************************/

import com.google.soap.search.*;
import java.io.*;
import java.lang.String.*;
```

```
/**
 * The MyGoogleAPI is a program using the Google Web APIs for
 * search and fetching pages from the cache.
 *
 * @see com.google.soap.search.GoogleSearch
 * @see com.google.soap.search.GoogleSearchResult
 */

/***************************************************************/

public class MyGoogleAPI {

  /** Class to call the Google Web APIs service.
   ** Usage:<br>
   **   <tt>java com.google.soap.search.MyGoogleAPI [key] search Foo</tt><br>
   **/

  /*************************************/
  /**  Main function **/
  public static void main(String[] args) {

    // Parse the command line
    if (args.length != 3) {
      printUsageAndExit();
    }


    String clientKey = "RNhCh4RQFHJM4nJAf4X/NSc6qbWAenEw";
    // This is the clientKey provided from Google Web API
    // to the author Isheeta Nargis

    String directive = args[0]; // whether search, cached or spelling query
    String directiveArg = args[1]; // query key
    int k = Integer.parseInt(args[2]); // the number of queries to call

    // Report the arguments received
    System.out.println("Parameters:");
    System.out.println("Client key = " + clientKey);
    System.out.println("Directive  = " + directive);
    System.out.println("Args       = " + directiveArg);

    // Create a Google Search object, set our authorization key
```

```
    GoogleSearch s = new GoogleSearch();
    s.setKey(clientKey);
    s.setMaxResults(10); // 10 results per query,
                         // 10 is the maximum limit for this value


    try {
      // if search query
      if (directive.equalsIgnoreCase("search")) {
        s.setQueryString(directiveArg); // the query string

        // call k queries
        for(int i=0;i<k;i++) {
          // set the starting point for result
          // each query returns 10 pages
          s.setStartResult(i*10);
          // the search result
          GoogleSearchResult r = s.doSearch();
          // print search results
          System.out.println("Google Search Results:");
          System.out.println("======================");
          System.out.println(r.toString());
        } // for i
      } // if
    } catch (GoogleSearchFault f) {
      System.out.println("The call to the Google Web APIs failed:");
      System.out.println(f.toString());
    } // catch
  } // end main

  /************************************/

  private static void printUsageAndExit() {
    System.err.println("Usage: java " +
                       MyGoogleAPI.class.getName() +
                       " <client-key>" +
                       " (search <query> | cached <url> | spell <phrase>)");
    System.exit(1);
  } // end printUsageAndExit

  /************************************/
```

```
} // end MyGoogleAPI

/*****************************************************************************/
```

Appendix C

Code to list Arcs in an Induced Subgraph of $\mathcal{W}$

In Section 3.2 we described the way we construct the Arc set of a neighborhood graph $\mathcal{N}_D(v)$. Here we present a program to list the arcs from a particular vertex u (or the links from a particular web page u) in the subgraph of the web graph induced by a set S of web pages. This program is implemented in C. This program is used by the program listed in Appendix A to find the arcs from each vertex $u \in S_D(v)$.

```
/*
          Author: Isheeta Nargis

       This program is used to list the arcs from
       a particular vertex v (or the links from
       a particular web page v) in the subgraph of
       the web graph induced by a set S of web pages.

       Inputs:

       Command line inputs: adjacent.out m file1 file2

          a) adjacent.out is the executable code for this program
          b) m is the index of the url corresponding to vertex v
             in file1
          c) The file 'file1' contains the list of all urls
             in the set S, in ascending order of urls
          d) The file 'file2' contains the list of all urls
             linked to by the m-th web page in 'file1'
             (or all urls linked to by the web page
             corresponding to vertex v).
```

```
       Outputs:

            The list of arcs from vertex v to the
            vertices in S, where an arc from vertex a
            to vertex b represents a hyprelink from
            web page a to web page b.

            We use this function to construct
            the induced subgraph of the web graph.

*/

/************************************************************/


#include <math.h>
#include <fstream>
#include <istream>
#include <ostream>
#include <string>
#include <iostream>

using namespace std;

/*********************************************/
/* The main function*/

int main(int argc, char* argv[]){

 ifstream all_file; // file with all URLs
 ifstream temp_file; // file with downlinks

 char c_temp[4000];
 char c_all[4000];

 int i = 0;
 int m = atoi(argv[1]); // the total number of urls in file1
 int j;

 j = 0; // j is the index of current url in file1
```

```
string temp_str;
string all_str;

// open files
all_file.open(argv[2]);
temp_file.open(argv[3]);

// read one url s2 from file2
temp_file >> temp_str;
strcpy(c_temp,temp_str.c_str());

// read one url s1 from file1
all_file >> all_str;
strcpy(c_all,all_str.c_str());

while (temp_file.good()&&all_file.good()){
  // while none of file1 and file2 has ended

 // compare the current urls s1 and s2 from two files
 i = (strcmp(c_all,c_temp));

 if (i==0) {
   // the urls s1 and s2 are same
   // that means the index of the url s2 in 'file1' is j
   cout << m << "\t" << j << endl;
   // There is an arc from vertex m to vertex j,
   // that is, there is a link from m-th url in
   // 'file1' to j-th url in 'file1'

   // read next url from file2
   temp_file >> temp_str;
   strcpy(c_temp,temp_str.c_str());

   // read next url from file1
   all_file >> all_str;
   strcpy(c_all,all_str.c_str());

   // increment j, the index of current url in file1
   j++;
 }
 else if (i > 0) {
   // s1 follows s2 in sorted order;
```

```
    // so move forward in file2
    // read another url from file2
    temp_file >> temp_str;
    strcpy(c_temp,temp_str.c_str());
  }
  else if (i<0) {
    // s1 precedes s2 in sorted order;
    // so move forward in file1
    // read another url from file1

    all_file >> all_str;
    strcpy(c_all,all_str.c_str());

    // increment j, the index of current url in file1
    j++;

  } // else
 } // while

 // close files
 all_file.close();
 temp_file.close();

 return (0);

} // end main
/************************************************/
```

Appendix D

Code to find the Bow-Tie Regions in a Graph

In Section 2.2 we described the Bow-Tie structure of the Web. In Section 3.4.1 we presented the relative size of regions in the Bow-Tie structure in some neighborhood graphs. Here we present a program that finds the Bow-Tie regions in a given Graph G. This program is implemented in C. At first it applies the Floyd-Warshall algorithm to find the length of a shortest path between all pairs of vertices in G. Then it finds the strongly connected components in G from the computed shortest path lengths. The SCC region in Bow-Tie structure is the largest strongly connected component in G. Then it finds the Bow-Tie regions in G by checking for the existence of a path from a vertex u to v in G in various Bow-Tie regions.

```
/*
          Author: Isheeta Nargis

       This program is used to find the regions
       in Bow-Tie Structure in a graph G.

       Inputs:

       Graph G is given in two files ‘size_both’ and ’adjacent’:

          a) ‘size_both’ stores the number of vertices in V(G).
          b) ‘adjacent’ lists the arcs (u,v) in E(G).

       Outputs:

        A file named ‘scc’ storing the following information:

        1. For each vertex v in V(G) the strongly
```

```
           connected component number v is part of.
        2. The size of all the strongly connected
           components in G.
        3. For each vertex v in V(G) , the label of v
           to denote whether it is part of the Bow-Tie regions
           SCC,IN,OUT,TT or DISC.
        4. The sizes of each of the Bow-Tie regions in G.
        5. If DISC is nonzero then the list
           of vertices in DISC.


*/

/************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>


#define INF 32767
#define NONE 0
#define SCC 1
#define IN 2
#define OUT 3
#define TENDRIL 4
#define TUBE 5
#define DISC 6


bool *g; // input graph
int *d;  // all pair shortest path distance
int *c,*freq;
int *label; // label of a vertex to denote the strongly
            // connected component it is part of
int scccount; // number of strongly connected components


FILE *fin, *fout;


/*********************************************/
```

```
/* This function is an implementation of the
   Floyd-Warshall's algorithm to compute
   All Pair Shortest Paths in a Graph G.

   Inputs:
   1. n - the size of the vertex set of G
   2. g is a linear array acting as a
      2-dimensional array storing the graph G

     g[n*a+b] is a boolean value
     indicating whether there is an
     arc from a to b

   Output:
   1. d - a linear array acting as a
       2-dimensional array storing
       the all pair shortest path of G.

       d[n*a+b] denotes length of
       a shortest path from a to b
*/

void FloydWarshall(int n) {

   int i,j,k;
   int total,a,b;
   int max;
   int ij, ik, kj;


   total = n*n;

   // Allocate memory
   d = new int [total];

   /* d is a linear array acting as a
      2-dimensional array storing
      the all pair shortest path of G. */


   for(i=0;i<total;i++) {
      if (g[i] == 1) { // there is an arc
```

```
          d[i] = g[i];
          // if there is an arc then
          // distance is one
      } else {
          d[i] = INF;
          // otherwise distance is infinite
      } // if
   }// for

   /* Floyd-Warshall Algorithm */

   for(k=0;k<n;k++) {
      for(i=0;i<n;i++) {
        for(j=0;j<n;j++) {
         //Calculate corresponding indices
         ij = n*i + j;
         ik = n*i + k;
         kj = n*k + j;
         if ( d[ij] > d[ik]+ d[kj] ) {

            // the path from i to j through k
            // is a shorter path than the
            // current path from i to j

            d[ij] = d[ik] + d[kj];

            // store the length of so far minimum path
            // from i to j
         } // if
        } // for j
      } // for i
   } // for k

} // end void FloydWarshall

/*******************************************************/
/** This function is used to find the
 Strongly Connected Components in a graph G.

 Inputs:
 1. n - the size of V(G)
 2. d - a linear array acting as a
```

```
       2-dimensional array storing
       the all pair shortest path of G.

       d[n*a+b] denotes length of
       a shortest path from a to b.
    d is computed by function FloydWarshall.

 Output:
 It lists the following information to file 'scc':
 1. For each vertex v in V(G) the strongly
    connected component number v is part of.
 2. The size of all the strongly connected
    components in G.

*/

void FindSCC(int n) {

   int i,j,k;
   int comp, count;
   int ij, ji;
   int total,max;

   // Allocate Memory
   c = new int [n];
   // c is a linear array
   // containing the component number
   // of vertices in V(G)

   // initialize
   for(i=0;i<n;i++)
      c[i]= 0;

   comp = 0;
   count = 0;
   while (count < n) {
      for(i=0;i<n;i++) {
        if ( c[i] == 0) {
           // this vertex i is
           // unlabeled, so i is not yet
           // part of another SCC
           count++;
```

```
          comp++;
          c[i] = comp;
          // label i with the next
          // SCC label
          break;
        } // if
      } // for i
      for(j=0;j<n;j++) {
        // look for other vertices
        // whether they are in the same
        // SCC as i
        if (c[j] == 0) {
          // j is unlabelled, so j
          // is not part of another
          // SCC
          ij = i*n + j;
          ji = j*n + i;
          if (d[ij] < INF && d[ji] < INF) {
            // there is a path from i to j
            // and also from j to i
            // so, i and j are in the same SCC
             count++;
             c[j] = comp;
             // label j with the current SCC label
             // that is, the label of i
          } // if
        } // if
      } // for j
    } // while


    // write output
    printf("\nn = %d\n",n);
    printf("\n The Strongly Connected Component Numbers are as follows:
            (in order of vertices)\n");
    fprintf(fout,"\nn = %d\n",n);
    fprintf(fout,"\n The Strongly Connected Component Numbers are as follows:
            (in order of vertices)\n");

    for(i=0;i<n;i++)
      fprintf(fout,"%d\t",c[i]);
```

```
   // Compute the total number of SCC
   max = 0;
   for(i=0;i<n;i++) {
     if (c[i] > max)
       max = c[i];
   }

   // freq array stores the sizes of the SCCs
   freq = new int [n];

   // initialize
   for(k=1;k<=max;k++)
     freq[k]=0;
   for(k=1;k<=max;k++) {
      for(i=0;i<n;i++) {
        if (c[i] == k)
          // i is part of k-th SCC
          freq[k] += 1;
      } // for i
   } // for k

   // write outputs
   printf("\n Strong Connected Components\n\n");
   printf("\n ComponentNo\tSize\n" );
   fprintf(fout,"\n Strong Connected Components\n\n");
   fprintf(fout,"\n ComponentNo\tSize\n" );
   for(k=1;k<=max;k++) {
     printf("%d\t%d\n",k,freq[k]);
   fprintf(fout,"%d\t%d\n",k,freq[k]);
   }

   // total number of SCC in G
   scccount = max;
}

/*************************************************************/
/*** This function finds the Bow-Tie regions based on
the Strongly Connected Components in G

 Inputs:
 1. c -- a linear array containing the component
```

```
   number of vertices in V(G)
 2. freq -- an array storing the sizes of the SCCs

 Outputs:
 It writes the following information into file:
 1. For each vertex v in V(G) , the label of v
   to denote whether it is part of the Bow-Tie regions
   SCC,IN,OUT,TT or DISC.
 2. The sizes of each of the Bow-Tie regions in G.
 3. If DISC is nonzero then the list
   of vertices in DISC.

***/

void FindSize(int n) {

   int sccsize, sccind;
   int insize,outsize,tendsize,tubesize,discsize;
   int index,i_in,i_out;
   int i,j,k;
   int ij,ji,ik,ki;


   // label is an array to denote
   // the Bow-Tie region of a vertex
   label = new int [n];
   for(i=0;i<n;i++)
      label[i] = NONE;
      // initilaize to NONE

   //sccsize is the size of SCC in Bow-Tie,
   // that means, it is the size of
   // the largest Strongly Connected Component
   // in G
   sccsize = 0;
   for(i=1;i<=scccount;i++) {
      if (freq[i] > sccsize) { // find maximum
         sccsize = freq[i];
         sccind = i; // store index
                     // of the largest Strongly
                     // Connected Component
      } // if
```

```
} // for i

// find the vertices in SCC
for(i=0;i<n;i++) {
   if (c[i] == sccind) {
      // i is in SCC region, so
      // label it as SCC
       label[i] = SCC;
       index = i; // store i as a
                  // representative vertex
                  // of SCC , to use in
                  // finding IN, OUT
   } // if
}

// insize is the size of IN in Bow-Tie
insize = 0;
// find vertices in IN
for(i=0;i<n;i++) {
   j = index; // j is the representative
              // vertex in SCC
   ij = n*i + j;
   ji = n*j + i;
   if (d[ij] < INF && d[ji] == INF)  {
      // there is a path from i to SCC
      // but no path from SCC to i;
      // so i is in IN region, so label
      // i as IN
      label[i] = IN;
      insize++;
   }
}

//outsize is the size of OUT in Bow-Tie
outsize = 0;
// find vertices in OUT
for(i=0;i<n;i++) {
   j = index; // j is the representative
              // vertex in SCC
   ij = n*i + j;
   ji = n*j + i;
   if (d[ij] == INF && d[ji] < INF) {
```

```
      // there is no path from i to SCC
      // but no path from SCC to i;
      // so i is in OUT region, so
      // label i as OUT
       label[i] = OUT;
       outsize++;
   } // if
} // for


// tendsize is the size of TENDRILS in Bow-Tie
tendsize = 0;
// find vertices in TENDRILS
for(i=0;i<n;i++) {
   if ( label[i] == NONE) {
     // i is not yet part of SCC, IN or OUT
     for(j=0;j<n;j++) {
       if (label[j] == IN) {
         // j is a representative vertex in IN
         ij = n*i + j;
         ji = n*j + i;
         if ( d[ij] == INF && d[ji] < INF) {
           // there is no path from i to IN
           // but there is a path from IN to i
           // so i is in TENDRILS region
           label[i] = TENDRIL;
           tendsize++;
           break;
         } // if
       } // if (label[j] == IN)
     } // for j

     if (label[i] == TENDRIL) // skip
        continue;

     for(j=0;j<n;j++) {
       if (label[j] == OUT) {
         // j is a representative vertex in IN
         ij = n*i + j;
         ji = n*j + i;
         if ( d[ij] < INF && d[ji] == INF) {
           // there is a path from i to OUT
```

```
            // but there is no path from OUT to i
            // so i is in TENDRILS region

            label[i] = TENDRIL;
            tendsize++;
            break;
          } // if
        } // if (label[j] == OUT)
      } // for j

    } //if (label[i] == NONE)
  } // for i

  // tubesize is the size of TUBES in Bow-Tie
  tubesize = 0;
  // find vertices in Bow-Tie
  for(i=0;i<n;i++) {
    if ( label[i] == TENDRIL) {
      // TUBES region is a part
      // of TENDRILS region
      // so at first it has to be labelled
      // as TENDRIL
      for(j=0;j<n;j++) {
        if (label[j] == IN) {
          for(k=0;k<n;k++) {
            if (label[k] == OUT) {
              if ( d[ij] == INF && d[ji] < INF &&
                   d[ik] < INF && d[ki] == INF) {
                // there is a path from IN to i
                // and also a path from i to OUT
                // but no path from i to IN
                // and no path from OUT to i;
                // so i is in TUBES region, so
                // label i as TUBE
                label[i] = TUBE;
                tendsize--;
                tubesize++;
                break;
               } // if
            } // if (label[k] == OUT)
          } // for k
        } // if (label[j] == IN)
```

```
        if (label[i] == TUBE)
            break;
      } // for j
  } // if ( label[i] == TENDRIL)
} // for i

// discsize is the size of DISC in Bow-Tie
discsize = 0;
for(i=0;i<n;i++) {
   if (label[i] == NONE) {
     // i is not part of SCC, IN, OUT,
     // TENDRILs or TUBES region, so
     // i is in DISC, so
     // label i as DISC
      label[i] = DISC;
      discsize++;
   } // if
} //for

// write output
printf("\n The labels of vertices in the vertice order:\n");
fprintf(fout,"\n The labels of vertices in the vertice order:\n");
for(i=0;i<n;i++) {
   switch (label[i]) {
         case SCC: printf("S\t");
                   fprintf(fout,"S\t");
                   break;
         case IN: printf("I\t");
                  fprintf(fout,"I\t");
                  break;
         case OUT: printf("O\t");
                   fprintf(fout,"O\t");
                   break;
         case TENDRIL: printf("T\t");
                       fprintf(fout,"T\t");
                       break;
         case TUBE: printf("U\t");
                    fprintf(fout,"U\t");
                    break;
         case DISC: printf("D\t");
                    fprintf(fout,"D\t");
                    break;
```

```
      } // switch
   }// for

   printf("\n|n The sizes of the components are\n\n");
   printf("SCC = %d\n",sccsize);
   printf("IN = %d\n",insize);
   printf("OUT = %d\n",outsize);
   printf("TENDRIL = %d\n",tendsize);
   printf("TUBE = %d\n",tubesize);
   printf("DISC = %d\n",discsize);
   fprintf(fout,"\n|n The sizes of the components are\n\n");
   fprintf(fout,"SCC = %d\n",sccsize);
   fprintf(fout,"IN = %d\n",insize);
   fprintf(fout,"OUT = %d\n",outsize);
   fprintf(fout,"TENDRIL = %d\n",tendsize);
   fprintf(fout,"TUBE = %d\n",tubesize);
   fprintf(fout,"DISC = %d\n",discsize);

   if (discsize > 0) {
      printf("The DISC nodes are:\n\n");
      fprintf(fout,"The DISC nodes are:\n\n");
      for(i=0;i<n;i++) {
         if (label[i] == DISC) {
            printf("%d\t%lf\t%lf\n",i);
            fprintf(fout,"%d\t%lf\t%lf\n",i);
         }
      }
   }


} // end FindSize

/************************************************/

/* The Main Function */
/************************************************/
int main (void) {

 int a,b;
 int i,n;
```

```
// Read the size of the vertex set

fnum = fopen("size_both","rt");
fscanf(fnum,"%d",&size);
fclose(fnum);



/* Allocate memory - we use one-dimensional
   array to use as a 2-dimensional array.
   The reason is that we can use dynamic
   memory allocation in this way. */

g = new bool [n*n];

// Initialize
for (i=0;i<n*n;i++) {
 g[i]=0;
}

fin = fopen("adjacent","rt");
fout = fopen("scc","wt");

// Read the input graph
while (!feof(fin)) {
  fscanf(fin,"%d",&a);
  fscanf(fin,"%d",&b);
  g[n*a + b] = 1;
  // There is an arc from a to b
}
fclose(fin);

// Compute all pair shortest path

FloydWarshall(n);

// Find the strongly connected components
// from the computed shortest paths

FindSCC(n);

// Find the Bow-Tie regions based on
```

```
 // the strongly connected components

 FindSize(n);


 // Free allocated memory
 delete[] g;
 delete[] d;
 delete[] c;
 delete[] freq;
 delete[] label;

 fclose(fout);
 return (0);
}
```

Appendix E

Code for Degree Distributions

In Section 3.4.2 we presented the indegree and outdegree distributions in some neighborhood graphs. Here we present a program that produces the indegree and outdegree distributions in a given Graph G. This program is implemented in C.

```
/*
          Author: Isheeta Nargis

      This program is used to find the indegree
      and outdegree distributions in a graph G.

      Inputs:

      Graph G is given in two files 'size_both' and 'adjacent':

         a) 'size_both' stores the number of vertices in V(G).
         b) 'adjacent' lists the arcs (u,v) in E(G).

      Outputs:

       A file named 'degree' stores the following information:

       1. The indegree distribution of G is written as
          pairs (k, incount(k)) where incount(k) denotes
          the number of vertices in G with indegree k
       2. The outdegree distribution of G is written as
          pairs (k, outcount(k)) where outcount(k) denotes
          the number of vertices in G with outdegree k

*/
```

```
/*************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>


bool *g;
int *ind,*outd;
int *count;

FILE *fin,*fout,*fnum;

/*************************************************************/
/*
 This function produces the indegree and outdegree
 distributions of a graph G.

 Input:

   1. n - the size of the vertex set of G
   2. Input Graph G stored in the array g

 Outputs:

   The file 'degree' stores the following information:

   1. The indegree distribution of G is written as
      pairs (k, incount(k)) where incount(k) denotes
      the number of vertices in G with indegree k
   2. The outdegree distribution of G is written as
      pairs (k, outcount(k)) where outcount(k) denotes
      the number of vertices in G with outdegree k

*/

void degree(int n) {

   int i,j,k;
   int total,a,b;
   int max;
```

```
total = n*n;


// ind is the array of indegree of vertices
// outd is the array of outdegree of vertices

// Allocate memory
ind = new int [n];
outd = new int [n];


// initialize indegree and outdegree
// of each vertex to zero
for(i=0;i<n;i++)
  ind[i] = outd[i] = 0;

// compute indegree and outdegree of each vertex
for(i=0;i<total;i++) {
   if (g[i] == 1) { // there is an arc
      a = i / n;
      b = i % n;
      // there is an arc from
      // vertex a to vertex b;
      // outdegree of a is incremented by one
      outd[a] += 1;
      // indegree of b is incremented by one
      ind[b] += 1;
   } // if g[i]
} // for i

// indegree distribution

// count is the array containing number of
// vertices in G with a particular indegree

// allocate memory
count = new int [n];

// max is the maximum indegree
max = 0;
for(i=0;i<n;i++) {
  if (ind[i] > max)
```

```
    max = ind[i];
}

// initialize count array to zero
for(k=0;k<=max;k++)
  count[k]=0;

// compute count array
for(k=0;k<=max;k++) {
   // check for the existence of
   // vertices with a particular indegree k
   for(i=0;i<n;i++) {
     if (ind[i] == k)
       // vertex i has indegree k
         count[k] += 1;
   } // for i
} // for k


printf("\nn = %d\n",n);
printf("\n In-degree distribution\n\n");
fprintf(fout,"\nn = %d\n",n);
fprintf(fout,"\n In-degree distribution\n\n");

// write indegree distribution into file
for(k=0;k<=max;k++) {
  printf("%d\t%d\n",k,count[k]);
  if ( (count[k] != 0) && (k !=0) )
    // There exists some vertices with indegree k
    fprintf(fout,"%d\t%d\n",k,count[k]);
}

// Similar process is performed for the
// outdegree distribution in G


// max is the maximum outdegree
max = 0;
for(i=0;i<n;i++) {
  if (outd[i] > max)
    max = outd[i];
}
```

```
   // count is the array containing number of
   // vertices in G with a particular outdegree


   // initialize count array to zero
   for(k=0;k<=max;k++)
     count[k]=0;

   // compute count array
   for(k=0;k<=max;k++) {
      // check for the existence of
      // vertices with a particular outdegree k
      for(i=0;i<n;i++) {
        if (outd[i] == k)
          // vertex i has outdegree k
           count[k] += 1;
      } // for i
   } // for k

   printf("\nn = %d\n",n);
   printf("\n Out-degree distribution\n\n");
   fprintf(fout,"\nn = %d\n",n);
   fprintf(fout,"\n Out-degree distribution\n\n");

   // write outdegree distribution into file
   for(k=0;k<=max;k++) {
     printf("%d\t%d\n",k,count[k]);
     if ( (count[k] != 0) && (k !=0) )
       // There exists some vertices with outdegree k
       fprintf(fout,"%d\t%d\n",k,count[k]);
   }

   // close file
   fclose(fout);


} // end void degree

/*****************************************************/
```

```
/** The Main Function **/
/*****************************************************/
int main(void) {

 int a,b;
 int i;
 int n;


 // Read the size of the vertex set

 fnum = fopen("size_both","rt");
 fscanf(fnum,"%d",&size);
 fclose(fnum);

 /* Allocate memory - we use one-dimensional
     array to use as a 2-dimensional array.
     The reason is that we can use dynamic
     memory allocation in this way. */

 g = new bool [n*n];

 // initialize
 for (i=0;i<n*n;i++) {
     g[i]=0;
 }

 fin = fopen("adjacent","rt");


 while (!feof(fin)) {
   fscanf(fin,"%d",&a);
   fscanf(fin,"%d",&b);
   g[n*a + b] = 1;
   // There is an arc from a to b
 }
 fclose(fin);

 fout = fopen("degree","wt");

 // call the function to produce
 // the indegree and outdegree
```

```
 // distribution of G

 degree(n);

 // Free allocated memory
 delete[] g;
 delete[] ind;
 delete[] outd;
 delete[] count;


 return (0);
} // end main

/**************************************/
```

Appendix F

Code for HITS Algorithm

In Chapter 4 we presented the extraction of authorities and hubs in some neighborhood graphs. Here we present a program that implements the HITS algorithm. This program is implemented in C.

```
/*
      Author: Isheeta Nargis

  This program is an implementation of the HITS
  algorithm introduced by Kleinberg.

  Inputs:

  1. Graph G is given in three files 'size_both',
     'both' and 'adjacent':
     a) 'size_both' stores the number of vertices in V(G).
     b) 'both' lists the url of the vertices in ascending
         order of the url, each url in a line.
     c) 'adjacent' lists the arcs (u,v) in E(G).
  2. k - the number of iterations of HITS algorithm.

  Outputs:

  1. The file 'HITSoutput' lists the vertex number, its
     authority weight and hub weight for each vertex in V(G).
  2. The file 'auth' lists the vertex number and its
     authority weight for each vertex in V(G).
  3. The file 'hub' lists the vertex number and its
     hub weight for each vertex in V(G).

  The resaon for using three files is that later on we will
```

```
  sort the 'auth' and 'hub' files based on authority weight
  and hub weight, respectively.


*/

#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#include <math.h>


bool *g; // array to store the graph G
float *x, *y;
     // x stores authority weights of vertices in G
     // y stores hub weights of vertices in G


/* function to compute square of a real number */

float sqr(float x) {

  return (x*x);
}


/*********************************************/
/*
 This function implements HITS algorithm.

 Inputs:

 1. Input Graph G stored in the array g
 2. k - the total number of iterations of HITS
    to perform on G

 Outputs:
 1. The array x stores the authority weights
    of vertices in V(G) after performing
    k iterations of HITS on G
 2. The array y stores the hub weights
    of vertices in V(G) after performing
```

```
    k iterations of HITS on G

*/

void hits(int n, int k) {

   int i,j;
   int total,a,b;
   float *xnew, *ynew;
      // used for temporary calculation
   float sx,sy;

   total = n*n;

   // Allocate memory
   x = new float [n];
   y = new float [n];
   xnew = new float [n];
   ynew = new float [n];


   // Initialize authority weight
   // and hub weight of each vertex
   // to one
   for(i=0;i<n;i++)
     x[i] = y[i] = 1;

   for(j=0;j<k;j++) { // Perform k iterations of HITS
     // start of iteration j

     // xnew stores the authority weight
     // array for next iteration
     // ynew stores the hub weight
     // array for next iteration
     // xnew, ynew are initialized to zero
     for(i=0;i<n;i++)
         xnew[i] = ynew[i] = 0;

     // compute authority weights
     for(i=0;i<total;i++) {
       if (g[i] == 1) { // there is an arc
         a = i / n;
```

```
      b = i % n;
      // that means,
      // there is an arc from
      // vertex a to vertex b

      // authority weight of a vertex b
      // is the sum of the hub weights
      // of the vertices from which
      // there is an arc to b

      xnew[b] += y[a];

    } // if g[i]
  } // for i

  // compute hub weights
  for(i=0;i<total;i++) {
    if (g[i] == 1) { // there is an arc
      a = i / n;
      b = i % n;
      // that means,
      // there is an arc from
      // vertex a to vertex b

      // hub weight of a vertex a
      // is the sum of the authority weights
      // of the vertices to which
      // there is an arc from a

      ynew[a] += xnew[b];
    } // if g[i]
  } // for i


  // normalize computed authority weights
  // and hub weights
  sx = 0;
  for(i=0;i<n;i++)
    sx += sqr(xnew[i]);
  sx = sqrt(sx);
  for(i=0;i<n;i++)
    x[i]= xnew[i]/sx;
```

```
     sy = 0;
     for(i=0;i<n;i++)
       sy += sqr(ynew[i]);
     sy = sqrt(sy);
     for(i=0;i<n;i++)
       y[i]= ynew[i]/sy;

     // end of iteration j
   }// for j

} // end void hits


/*************************************/
/* The Main Function */

int main (void) {

    FILE *fin, *fout,*f1,*fa,*fh;
    int a,b;
    int i,k;
    int n;

    // read the size of the vertex set

    f1 = fopen("size_both","rt");
    fscanf(f1,"%d",&n);
    fclose(f1);


    /* Allocate memory - we use one-dimensional
     array to use as a 2-dimensional array.
     The reason is that we can use dynamic
     memory allocation in this way. */

    g = new bool [n*n];
    if (g== NULL){
       printf("error");
       return 2;
    }

    // Initialize
```

```
for (i=0;i<n*n;i++) {
   g[i]=0;
}


fin = fopen("adjacent","rt");
fout = fopen("HITSOutput","wt");
fa = fopen("auth","wt");
fh = fopen("hub","wt");

while (!feof(fin)) {
  fscanf(fin,"%d",&a);
  fscanf(fin,"%d",&b);
   // That means there is an arc
   // from vertex a to vertex b
  g[n*a + b] = 1;

}
fclose(fin);

// Asks user the number of iterations of HITS
printf("\n How many iterations on HITS algorithm? \n");
scanf("%d",&k);

// Call fuinction hits to iterate
// HITS algorithm k times on G
hits(n,k);

// write outputs
for(i=0;i<n;i++){
 fprintf(fout,"%d\t%.6f\t%.6f\n",i,x[i],y[i]);
 fprintf(fa,"%.6f\t%d\n",x[i],i);
 fprintf(fh,"%.6f\t%d\n",y[i],i);
}


// Close Files
fclose(fout);
fclose(fa);
fclose(fh);

return (0);
```

```
}
```

Appendix G

Code for Iterative Cycle Contraction Algorithm

In Chapter 5 we presented the Iterative Cycle Contraction algorithm. In Section 5.4 we presented the results of the algorithm. Here we present a program that implements the Iterative Cycle Contraction algorithm. This program is implemented in C.

```
/*
            Author: Isheeta Nargis

     This program is the implementation of
     Iterative Cycle Contraction Algorithm.

     Inputs:

       Graph G_0 is given in three files ‘size_both,
        ‘both’ and ’adjacent’:

          a) ‘size_both’ stores the number of vertices in V(G_0).
          b) ‘both’ lists the url of the vertices in ascending
             order of the url, each url in a line.
          c) ‘adjacent’ lists the arcs (u,v) in E(G_0).

     Outputs:

      I. ‘amalgamate_count’ file stores the
           number of directed 2-cycles in successive
           iterations of the contract function.

      II. The following files for each iteration k:
```

```
      1. 'contractk' stores all the directed 2-cycles in G_{k-1}.
      2. 'labelk' stores the vertex, label pairs (v, label(v))
          for each vertex  v in V(G_{k-1}).
      3. 'clusterk' stores the following information
         for each cluster C in G_{k-1}:
         a) The cluster label of C
         b) The size of C in terms of the number
           of vertices in G_{k-1} contracted to C.
         c) The member vertex numbers in C.
      4. 'cluster_nodek' stores the following information
         for each cluster C in G_{k-1}:
         a) The cluster label of C.
         b) The size of C in terms of the number
           of vertices in G_{k-1} contracted to C.
         c) If iteration = 1
              the member vertex urls in C
           Otherwise
              the urls in the member communities in C.
      5. 'cluster_node_degreek' stores the following
         information for each cluster C in G_{k-1}:
          a) The cluster label of C.
          b) The size of C in terms of the number
            of vertices in G_{k-1} contracted to C.
          c) If iteration = 1
              the member vertex urls in C
            Otherwise
              the urls in the member communities C.
          d) The indegree and outdegree (in G_{k-1})
            of the member vertices C.
      6. 'clustersizek' stores the following information
         for each cluster C in G_{k-1}:
          a) The size of C in terms of the
            number of vertices in G_{k-1} contracted to C.
          b) The size of C in terms of the
            number of vertices in G_0 contracted to C,
            that is, the number of web pages contracted to C.


*/

#include <math.h>
#include <limits.h>
```

```
#include <fstream>
#include <istream>
#include <ostream>
#include <string>
#include <iostream>
using namespace std;

bool *D; // input graph
bool *newD; // output graph
int *W;
int *label,*newlabel,*clusterlabel;
int *cluster;
int *clustersize;
int *clusterstart;
int *clusternodes;
bool *flag;
ifstream adjacent_file;
ofstream out;
FILE *fin,*fout,*fnum,*f2,*f3,*f4,*f5,*f6,*f62,*f0,*f7,*f8,*f9,*f10;
char s[1000],snum[10],name[30];
int x,y,z;



/*
  This function is the implementation of the contract function
  Inputs:
         1. Iteration Number
         2. The size of the vertex set of the input graph
         3. Input Graph is stored in the array D
  Outputs:

     Several output files are used to store the outputs,
     as discussed in the initial comments.
     Moreover, this function stores the
     following temporary files in each iteration:

     1. 'newgraph' stores the arcs in the new graph G_k.
         To extract the communities represented by a vertex
         in G_k we have to read the file 'cluster_nodek'.
     2. 'sizenewgraph' stores the size of the new graph G_k.
```

```
*/

int contract(int iter,int n) {

 int i,j,k,l,m,index;
 int count;
 int max,maxsize,totalcluster;
 int bigsize,newtotal,newn,start;
 int indegree,outdegree;
 int a,b;
 int size2;

 itoa(iter,snum,10);

 strcpy(name,"contract");
 strcat(name,snum);
 f0 = fopen(name,"wt");

 // cycle_count is the number of
 // directed 2-cycles in the current graph

 cycle_count = 0;
 for (i=0; i<n; i++){
   for(j=i+1;j<n;j++){
     if ( D[n*i+j] && D[n*j+i]){
           // There is a directed 2-cycle between
           // vertex i and vertex j
       fprintf(f0,"%d\t%d\n",i,j);
          // we list each directed 2-cycle in this file
       for(k=j+1;k<n;k++){
         // check whether j is the label of
         // some other vertex k, k > j
         if (label[k] == label[j]) {
           label[k] = label[i];
           // if so, then make the label
           // of that vertex k to label of i
         } // if
       } // for k
       // make the label of j to the label of i
       label[j]=label[i];
       //cycle_count is incremented
       cycle_count++;
```

```
    }// if ( D[n*i+j] && D[n*j+i])
  }// for j
} // for i

//if no directed 2-cycles in the current graph
if (cycle_count == 0)
  return cycle_count;

// open files

strcpy(name,"label");
strcat(name,snum);
f4 = fopen(name,"wt");

strcpy(name,"cluster");
strcat(name,snum);
f5 = fopen(name,"wt");

strcpy(name,"cluster_node");
strcat(name,snum);
f6 = fopen(name,"wt");

strcpy(name,"cluster_node_degree");
strcat(name,snum);
f62 = fopen(name,"wt");

strcpy(name,"clustersize");
strcat(name,snum);
f9 = fopen(name,"wt");

strcpy(name,"newgraph");
strcat(name,snum);
f7 = fopen(name,"wt");

cluster = new int [n];

max = 0;
for(i=0;i<n;i++){
   if (label[i] > max)
     max = label[i]; // max is the maximum label
} // for i
```

```
for(k=0;k<=max;k++){
    cluster[k]=0;
}

// cluster[k] is the number of vertices with label k
for(k=0;k<=max;k++) {
     for(i=0;i<n;i++) {
       if (label[i] == k)
       cluster[k] += 1;
     } // for i
} // for k

newlabel = new int [n];
clustersize = new int [n];
clusterlabel = new int [n];

// totalcluster is the number of
// clusters in the current graph

totalcluster = 0;
for(k=0;k<=max;k++) {
  if (cluster[k] > 0) {
      // there is some vertex in this cluster
    clustersize[totalcluster] = cluster[k];
      // clustersize[k] denotes the size of k-th cluster
    clusterlabel[k] = totalcluster;
      // clusterlabel[k] denotes the label of k-th cluster
    totalcluster++;
  } // if
} //for k

for(i=0;i<n;i++){
  k = label[i];
  newlabel[i] = clusterlabel[k];
    // newlabel of a vertex is the
    // cluster number of that vertex
    fprintf(f4,"%d\t%d\n",i,newlabel[i]);
} // for i

clusterstart = new int [n];

/*
```

```
We maintain a linear list containing the vertices of clusters
clusterstart[k] denotes the start of k-th cluster in this list
*/

index = 0;
for(k=0;k<totalcluster;k++) {
  clusterstart[k] = index;
  index += clustersize[k];
} //for k

// clusternodes is the linear list
// containing the vertices of clusters

clusternodes = new int [n];

newn = totalcluster; // newn is the size of the new graph

strcpy(name,"sizenewgraph");
strcat(name,snum);
f8 = fopen(name,"wt");
fprintf(f8,"%d",newn);
fclose(f8);

for(k=0;k<totalcluster;k++) {
  // For each cluster do
  start = clusterstart[k];
  l=0;
  for(i=0;i<n;i++){
    // search for vertices in k-th cluster
    if (newlabel[i] == k) {
        // the vertex i belongs to k-th cluster
        clusternodes[start+l]=i;
        l++;
    } // if (newlabel[i] == k)
    if (l == clustersize[k])
      // processing of k-th cluster is done
      break;
  } // for i
}// for k

// write outputs
for(k=0;k<totalcluster;k++) {
```

```
   // For each cluster do
fprintf(f9,"%d\t",clustersize[k]);
fprintf(f5,"\n%d\t%d\n",k,clustersize[k]);
fprintf(f6,"\n%d\t %d\n",k,clustersize[k]);
fprintf(f62,"\n%d\t %d\n",k,clustersize[k]);
fprintf(f62,"\nURL\t Indegree\t Outdegree\n");

if (iter == 1) {
    fprintf(f9,"%d\n",clustersize[k]);
}

start = clusterstart[k];
   // starting position for k-th cluster
size2 = 0;
  // size2 denotes the number of web pages
  // in a cluster
for(l=start;l<start+clustersize[k];l++){
  i = clusternodes[l]; // i is a vertex in k-th cluster
  // compute indegree and outdegree
  indegree = 0;
  outdegree = 0;
  for(j=0;j<n;j++){
     indegree += D[i*n+j];
     outdegree += D[j*n+i];
  } // end for j
  fprintf(f5,"%d\t",i);
  // extract the url(s) for vertex i
  if (iter == 1) {
     // the vertex i is a single vertex,
     // look up in the file 'both'
     f3 = fopen("both","rt");
     for(j=0;j<=i;j++)
       fscanf(f3,"%s",s);
     fprintf(f6,"%s\n",s);
     fprintf(f62,"%s\t",s);
     fprintf(f62,"%d\t%d\n",indegree,outdegree);

     fclose(f3);
  } else { // if (iter > 1)
    // the vertex i is a cluster of vertices,
    // look up through several files
```

```
      fprintf(f62,"\t%d\t%d\n",indegree,outdegree);
      strcpy(name,"clustersize");
      itoa(iter-1,snum,10);
      strcat(name,snum);
      f2 = fopen(name,"rt");

      strcpy(name,"cluster_node");
      itoa(iter-1,snum,10);
      strcat(name,snum);
      f3 = fopen(name,"rt");

      for(j=0;j<i;j++) {
        fscanf(f2,"%d %d",&x,&z);
        fscanf(f3,"%d %d",&x,&y);
        for(m=1;m<=y;m++) {
          fscanf(f3,"%s\n",s);
        } // for m
        fscanf(f3,"\n");
      }// for j
      fscanf(f2,"%d %d",&x,&z);
      size2 += z;
      fscanf(f3,"%d %d",&x,&y);
      for(m=1;m<=y;m++) {
         fscanf(f3,"%s\n",s);
         fprintf(f6,"%s\n",s);
         fprintf(f62,"%s\n",s);
      } // for m
      fprintf(f6,"\n");
      fprintf(f62,"\n");
      fclose(f2);
      fclose(f3);
    } // else // if (iter > 1)
  } // for l
  fprintf(f5,"\n");
  if (iter > 1)  {
      // write down the
      // cumulative size of the cluster
      fprintf(f9,"%d\n",size2);
  } // if
}// for k

  // write the new graph into a file
```

```
newtotal = newn*newn;
newD = new bool [newtotal];

for(k=0;k<newtotal;k++)
  newD[k] = 0;

for(i=0;i<n;i++){
  for(j=0;j<n;j++){
    if (D[i*n+j]) {
      // there is an arc from vertex i to vertex j
      if (newlabel[i] != newlabel[j]) {
         // vertex i and vertex j
         // are in different clusters
         // newlabel[i] is the cluster vertex of i
         // newlabel[j] is the cluster vertex of j
        fprintf(f7,"%d\t%d\n",newlabel[i],newlabel[j]);
         // add an arc from i's cluster vertex
         // to j's cluster vertex
      } // if (newlabel[i] != newlabel[j])
    } // if (D[i*n+j])
  } // for j
} // for i

// close files

fclose(f0);
fclose(f4);
fclose(f5);
fclose(f6);
fclose(f62);
fclose(f7);
fclose(f9);

// Free allocated memory

delete[] cluster;
delete[] clusternodes;
delete[] clusterstart;
delete[] newlabel;
delete[] newD;
delete[] clustersize;
delete[] clusterlabel;
```

```
   return cycle_count;
}
/* end Contract function*/


/* The main function */

int main(void) {

 int a,b;
 int i,total;
 int count,iter;
 int size,newsize;

 // read the size of the vertex set
 fnum = fopen("size_both","rt");
 fscanf(fnum,"%d",&size);
 fclose(fnum);

 f10 = fopen("amalgamate_count","wt");

 /* Allocate memory - we use one-dimensional
    array to use as a 2-dimensional array.
    The reason is that we can use dynamic
    memory allocation in this way. */

 W = new int [size];
 flag = new bool [size];
 label = new int [size];

 total = size*size;

 D = new bool [total];  // This is the input graph

 // initialize
 for (i=0;i<size;i++) {
  W[i]=i;
  flag[i]=0;
  label[i]=i; // initially the label of a vertex
              // is the vertex itself
 }
```

```
for (i=0;i<total;i++) {
 D[i]=0;
}

adjacent_file.open("adjacent");
while (adjacent_file.good()) {
  adjacent_file >> a;
  adjacent_file >> b;
  D[size*a + b] = 1; // That means there is an arc
                     // from vertex a to vertex b
}
adjacent_file.close();

fprintf(f10,"Iteration \tAmalgamate Count\n",iter,count);

// start iterative call to contract function

iter = 1;
count = contract(iter,size); // first call
printf("\nIteration %d amalgamate count = %d\n",iter,count);
fprintf(f10,"%d\t%d\n",iter,count);

if ( count == 0) // if no 2-cycle in G_0 then return
  return 0;

do {
  iter++; // go to next iteration, iteration k

  // read the size of the vertex set of the graph G_k
  itoa(iter-1,snum,10);
  strcpy(name,"sizenewgraph");
  strcat(name,snum);
  fnum = fopen(name,"rt");

  fscanf(fnum,"%d",&size);
  fclose(fnum);

  total = size*size;

  \\initialize
  for (i=0;i<size;i++) {
```

```
    W[i]=i;
    flag[i]=0;
    label[i]=i;
  }

  for (i=0;i<total;i++) {
     D[i]=0;
  }

  // read the graph G_k

  strcpy(name,"newgraph");
  strcat(name,snum);
  fin = fopen(name,"rt");

  while (!feof(fin)) {
    fscanf(fin,"%d",&a);
    fscanf(fin,"%d",&b);
    D[size*a + b] = 1;
  }
  fclose(fin);

  // call contract for the graph G_k
  count = contract(iter,size);
  printf("\nIteration %d amalgamate count = %d\n",iter,count);
  fprintf(f10,"%d\t%d\n",iter,count);

} while (count > 0);
  // continue iteration until the no. of 2-cycles is zero

fclose(f10);


// Free allocated memory

 delete[] W;
 delete[] D;
 delete[] flag;
 delete[] label;

 return (0);
} // end main
```

Bibliography

[Albert et al., 1999] Albert, R., Barabási, A., and Jeong, H. (1999). Diameter of the World-Wide Web. *Nature*, 401:130–131.

[Barabási and Albert, 1999] Barabási, A. and Albert, R. (1999). Emergence of scaling in random networks. *Science*, 286:509–512.

[Bonato, 2004] Bonato, A. (2004). A survey of models of the web graph. Combinatorial and Algorithmic Aspects of Networking.

[Botafogo and Shneiderman, 1991] Botafogo, R. A. and Shneiderman, B. (1991). Identifying Aggregates in Hypertext Structures. In *UK Conference on Hypertext*, pages 63–74. 3rd ACM Conference on Hypertext.

[Brin and Page, 1998] Brin, S. and Page, L. (1998). The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117.

[Broder et al., 2000] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R., Tomkins, A., and Wiener, J. (2000). Graph structure in the web. In *Proceedings of the 9th international World Wide Web conference on Computer networks*, pages 309–320.

[Carrire and Kazman, 1997] Carrire, S. and Kazman, R. (1997). WebQuery: Searching and visualizing the Web through connectivity. *Computer Networks*, 29(8–13):1257–1267.

[Chakrabarti and Faloutsos, 2006] Chakrabarti, D. and Faloutsos, C. (2006). Graph mining: Laws, generators, and algorithms. *ACM Computing Surveys*, 38(1):Article 2.

[Cho and Garcia-Molina, 2000] Cho, J. and Garcia-Molina, H. (2000). Synchronizing a database to improve freshness. pages 117–128. ACM International Conference on Management of Data (SIGMOD).

[Chung and Lu, 2004] Chung, F. and Lu, L. (2004). The small world phenomenon in hybrid power law graphs. In Ben-Naim, E. et al., editors, *Complex Networks*, pages 91–106. Springer–Verlag.

[Cooper and Frieze, 2003] Cooper, C. and Frieze, A. (2003). A General Model of Web Graphs. *Random Structures Algorithms*, 22:311–335.

[Cooper et al., 2004] Cooper, C., Frieze, A., and Vera, J. (2004). Random deletions in a scale free random graph process. *Internet Mathematics*, 1(4):463–483.

[Dean and Henzinger, 1999] Dean, J. and Henzinger, M. R. (1999). Finding related pages in the world wide web. In *WWW '99: Proceeding of the eighth international conference on World Wide Web*, pages 1467–1479, New York, NY, USA. Elsevier North-Holland, Inc.

[Dill et al., 2002] Dill, S., Kumar, R., Mccurley, K. S., Rajagopalan, S., Sivakumar, D., and Tomkins, A. (2002). Self-similarity in the web. *ACM Trans. Inter. Tech.*, 2(3):205–223.

[Donato et al., 2004] Donato, D., Laura, L., Leonardi, S., and Millozzi, S. (2004). Large scale properties of the Webgraph. *The European Physical Journal B*, 38(2):239–243.

[Fabrikant et al., 2002] Fabrikant, A., Koutsoupias, E., and Papadimitriou, C. (2002). Heuristically optimized tradeoffs: A new paradigm for power laws in the internet. 34th Symposium on Theory of Computing.

[Faloutsos et al., 1999] Faloutsos, M., Faloutsos, P., and Faloutsos, C. (1999). On power-law relationships of the Internet topology. In *SIGCOMM '99: Proceedings of the conference on Applications, technologies, architectures, and protocols for computer communication*, pages 251–262, New York, NY, USA. ACM Press.

[Gibson et al., 1998] Gibson, D., Kleinberg, J. M., and Raghavan, P. (1998). Inferring Web Communities from Link Topology. In *UK Conference on Hypertext*, pages 225–234.

[Gibson et al., 2005] Gibson, D., Kumar, R., and Tomkins, A. (2005). Discovering large dense subgraphs in massive graphs. In *VLDB '05: Proceedings of the 31st international conference on Very large data bases*, pages 721–732. VLDB Endowment.

[Graham, 2003] Graham, A. J. (2003). Investigations into the Iterative 2-cycle Contraction of the Directed Binomial Random Graph $\mathbb{D}(n, p)$. Honours Thesis. Memorial University of Newfoundland.

[Ke et al., 2006] Ke, Y., Deng, L., Ng, W., and Lee, D. (2006). Web dynamics and their ramifications for the development of web search engines. *Comput. Networks*, 50(10):1430–1447.

[Kleinberg, 1999] Kleinberg, J. M. (1999). Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM*, 46(5):604–632.

[Kleinberg et al., 1999] Kleinberg, J. M., Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. S. (1999). The Web as a Graph: Measurements, Models and Methods. *Lecture Notes in Computer Science*, 1627:1–17.

[Kumar et al., 2000] Kumar, R., Raghavan, P., Rajagopalan, S., Sivakumar, D., Tomkins, A., and Upfal, E. (2000). Stochastic models for the web graph. In *FOCS '00: Proceedings of the 41st Annual Symposium on Foundations of Computer Science*, pages 57–65, Washington, DC, USA. IEEE Computer Society.

[Kumar et al., 1999] Kumar, R., Raghavan, P., Rajagopalan, S., and Tomkins, A. (1999). Trawling the Web for emerging cyber-communities. *Computer Networks*, 286(11–16):1481–1493.

[Leskovec et al., 2005] Leskovec, J., Kleinberg, J., and Faloutsos, C. (2005). Graphs over time: Densification laws, shrinking diameters and possible explanations. In *KDD '05: Proceeding of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 177–187, New York, NY, USA. ACM Press.

[Liu et al., 2004] Liu, H., Milios, E., and Janssen, J. (2004). Probabilistic models for focused web crawling. In *WIDM '04: Proceedings of the 6th annual ACM international workshop on Web information and data management*, pages 16–22, New York, NY, USA. ACM Press.

[Mendelzon and Wood, 1995] Mendelzon, A. O. and Wood, P. T. (1995). Finding Regular Simple Paths in Graph Databases. *SIAM Journal on Computing*, 24(6):1235–1258.

[Nargis and Pike, 2007] Nargis, I. and Pike, D. A. (2007). Extracting Hierarchies of Communities from the Web. Preprint.

[Nargis et al., 2007] Nargis, I., Pike, D. A., and McKay, N. A. (2007). Modeling and Thematic Analysis of Neighborhood Structures in the Web. Preprint.

[Osthus and Buckley, 2004] Osthus, D. and Buckley, G. (2004). Popularity based random graph models leading to a scale-free degree sequence. *Discrete Mathematics*, 282(1–3):53–68.

[Pant et al., 2004] Pant, G., Srinivasan, P., and Menczer, F. (2004). Crawling the Web. In Levene, M. and Poulovassilis, A., editors, *Web Dynamics: Adapting to Change in Content, Size, Topology and Use.*, pages 153–178. Springer-Verlag.

[Redner, 1998] Redner, S. (1998). How Popular is Your Paper? An Empirical Study of the Citation Distribution. *The European Physical Journal B*, 4:131.

[Sydow, 2004] Sydow, M. (2004). Random surfer with back step. pages 352–353. 13th international World Wide Web conference on Alternate track papers & posters, ACM Press.

[Toyoda and Kitsuregawa, 2006] Toyoda, M. and Kitsuregawa, M. (2006). Whats really new on the web? identifying new pages from a series of unstable web snapshots. In *WWW '06: Proceedings of the 15th international conference on World Wide Web*, pages 233–241, New York, NY, USA. ACM Press.