Yanchun Zhang
Jeffrey Xu Yu
Jingyu Hou

Web Communities

Analysis and Construction

Springer

Web Communities

Yanchun Zhang · Jeffrey Xu Yu · Jingyu Hou

Web Communities

Analysis and Construction

With 28 Figures

Springer

Authors
Yanchun Zhang

School of Computer Science and Mathematics
Victoria University of Technology
Ballarat Road, Footscray
PO Box 14428
MC 8001, Melbourne City, Australia
yzhang@csm.vu.edu.au

Jeffrey Xu Yu

Dept. of Systems Engineering and Engineering Management
Chinese University of Hong Kong
Shatin, N.T., Hong Kong, China
yu@se.cuhk.edu.hk

Jingyu Hou

School of Information Technology
Deakin University
Burwood, Victoria 3125, Australia
jingyu@deakin.edu.au

Library of Congress Control Number: 2005936102

ACM Computing Classification (1998): H.3, H.5

ISBN-10 3-540-27737-4 Springer Berlin Heidelberg New York
ISBN-13 978-3-540-27737-8 Springer Berlin Heidelberg New York

Typeset by the authors using a Springer TEX macro package
Production: LE-TEX Jelonek, Schmidt & Vöckler GbR, Leipzig
Cover design: KünkelLopka Werbeagentur, Heidelberg

Printed on acid-free paper 45/3142/YL - 5 4 3 2 1 0

Dedication

To Jinli and Dana
From Yanchun

To Hannah, Michael and Stephen
From Jeffrey

To Huiming, Mingxi and Mingyi
From Jingyu

Contents

Preface

The rapid development of Web technology has made the World Wide Web an important and popular application platform for disseminating and searching information as well as conducting business. However, due to the lack of uniform schema for Web documents, the low precision of most search engines and the information explosion on the World Wide Web, the user is often flooded with a huge amount of information.

Unlike the conventional database management in which data models and schemas are defined, the Web community, which is a set of Web-based objects (documents and users) that has its own logical structures, is another effective and efficient approach to reorganize Web-based objects, support information retrieval and implement various applications. According to the practical requirements and concerned situations, the Web community would appear as different formats.

This book addresses the construction and analysis of various Web communities based on information available from Web, such as Web document content, hyperlinks, semantics and user access logs. Web community applications are another aspect emphasized in this book. Before presenting various algorithms, some preliminaries are provided for better understanding of the materials. Representative algorithms for constructing and analysing various Web communities are then presented and discussed. These algorithms, as well as their discussions, lead to various applications that are also presented in this book. Finally, this book summarizes the main work in Web community research and discusses future research in this area.

Acknowledgements

Our special thanks go to Mr. Guandong Xu and Mr. Yanan Hao for their assistance in preparing manuscripts of the book.

1 Introduction

1.1 Background

The rapid development of Web technology has made the World Wide Web an important and popular application platform for disseminating and searching for information as well as conducting business.

As a huge information source, World Wide Web has allowed unprecedented sharing of ideas and information on a scale never seen before. The boom in the use of the Web and its exponential growth are now well known, and they are causing a revolution in the way people use computers and perform daily tasks. On the other hand, however, the Web has also introduced new problems of its own and greatly changed the traditional ways of information retrieval and management.

Due to the lack of uniform schema for Web documents, the low precision of most search engines and the information explosion on the World Wide Web, the user is often flooded with a huge amount of information. Because of the absence of a well-defined underlying data model for the Web (Baeza-Yates and B. Ribeiro-Neto 1999), finding useful information and managing data on the Web are frequently tedious and difficult tasks, since the data on the Web is usually represented as Web pages (documents).

Usually, the effectiveness and efficiency of information retrieval and management are mainly affected by the logical view of data adopted by information systems. For the data on the Web, it has its own significantly different features compared with the data in conventional database management systems. The features of Web data are as follows.

- The amount of data on the Web is enormous. No one could have exactly estimated the data volume on the Web. Actually, the exponential growth of the Web poses scaling issues that are difficult to cope with. Even the current powerful search engine, such as *Google*, can only cover a fraction of the total documents on the Web. The enormous data on the Web makes it difficult to manage Web data using traditional database or data warehouse techniques.

- The data on the Web is distributed. Due to the intrinsic nature of the Web, the data is distributed across various computers and platforms, which are interconnected with no predefined topology.
- The data on the Web is heterogeneous. In addition to textual data, which is mostly used to convey information, there are a great number of images, audio files, video files and applications on the Web. In most cases, the heterogeneous data co-exist in a Web document, which makes it difficult to deal with them at the same time with only one technique.
- The data on the Web is unstructured. It has no rigid and uniform data models or schemas, and therefore there is virtually no control over what people can put on the Web. Different individuals may put information on the Web in their ways, as long as the information arrangement meets the basic display format requirements of Web documents, such as HTML format. The absence of well-defined structure for Web data brings a series of problems, such as data redundancy and poor data quality (Broder et al. 1997; Shivakumar N. 1998). On the other hand, documents on the Web have extreme variation internal to the documents, and also in external meta information that might be available (Brin and L. Page 1998). Although the currently used HTML format consists of some structuring primitives such as tags and anchors (Abiteboul 1997), these tags, however, deal primarily with the presentation aspects of document and have few semantics. Therefore, it is difficult to extract required data from Web documents and find their mutual relationships. This feature is quite different from that of traditional database systems.
- The data on the Web is dynamic. The implicit and explicit structure of the Web data may evolve rapidly, data elements may change types, data not conforming to the previous structure may be added, and dangling links and relocation problems will be produced when domain or file names change or disappear (Baeza-Yates and Ribeiro-Neto 1999). These characteristics result in frequent schema modifications that are another well-known headache in traditional database systems (McHugh et al. 1997).
- The data on the Web is hyperlinked. Unlike "flat" document collections, the World Wide Web is a hypertext and people are likely to surf the Web using its link graph. The hyperlinks between Web pages (data) provide considerable auxiliary information on top of the text of the Web pages and establish topological or semantic relationships among the data. This kind of relationship, however, is not in a predefined framework, which brings a lot of uncertainty, as well as much implicit semantic information, to the Web data.

The above features indicate that Web data is neither raw data nor very strictly typed as in conventional database systems.

Because of the above Web data features, Web information retrieval and Web data management are becoming a challenging problem. In the last several years, much research and development work has been done in this area. For this work, Web information search and management are always the main themes. Accordingly, the research and development work could be roughly classified into two main sub-areas: Web search engines and Web data management.

Web search engine technology has scaled dramatically with the growth of the Web since 1994 to help Web users find desired information, and has resulted in a large number of research results such as (McBryan 1994) (Brin and L. Page 1998) (Brin and Page) (Cho et al. 1998) (Sonnenreich and Macinta 1998) (Chakrabarti et al. 1999) (Chakrabarti et al. 1999) (Rennie J. and A. McCallum 1999) (Cho and. 2000; Cho and Garcia-Molina 2000; Diligenti et al. 2000; Hock 2000; Najork and Wiener 2001; Talim et al. 2001), as well as various Web search engines such as *World Wide Web Worm* (*WWWW*), *Excite*, *Lycos, Yahoo*!, *AltaVista* and *Google*. Search engines can be classified into two categories: one is general-purpose search engine and another one is special-purpose search engine. The general-purpose search engines aim at providing the capability of searching as many Web pages on the Web as possible. The search engines mentioned above are a few of the well-known ones. The special-purpose search engines, on the other hand, focus on searching those Web pages on particular topics. For example, the Medical World Search (*www.mwsearch.com*) is a search engine for medical information and Movie Review Query Engine (*www.mrqe.com*) lets the users to search for movie reviews. No matter what category the search engine is, each search engine has a text database defined by the set of documents that can be searched by the search engine. The search engine should have an effective and efficient mechanism to capture (crawl) and manage the Web data, as well as to provide the capabilities to handling queries quickly and returning the most related search results (Web pages) with respect to the user's queries. To reach these goals, effective and efficient Web data management is necessary.

Web data management refers to many aspects. It includes data modeling, languages, data filtering, storage, indexing, data classification and categorization, data visualization, user interface, system architecture, etc. (Baeza-Yates and B. Ribeiro-Neto 1999). In general, the purpose of the Web data management is to find intrinsic relationships among the data to effectively and efficiently support Web information retrieval and other Web-based applications. It can be seen that there are intersections between the research in Web search engines and Web data management. Effective and efficient Web data management is the base for a good Web search engine. On the other hand, the data management could be applied to many other Web applications, such as

Web-based information integration systems and *metasearch engines* (Meng et al. 2002).

Although much work has been done in Web-based data management in the last several years, there remain many problems to be solved in this area because of the characteristics of the Web data mentioned before. How to effectively and efficiently manage Web-based data, therefore, is an active research area.

1.2 Web Community

As Web-based data management systems are a kind of information system, there is much work trying to use traditional strategies and techniques to establish databases and manage the Web-based data.

For example, many data models and schemas have been proposed for managing Web data (Papakonstantinou et al. 1995; McHugh et al. 1997; Bourret et al. 2000; Laender et al. 2000; Sha et al. 2000; Surjanto et al. 2000; Yoon and Raghavan 2000). Some of them tried to define schemas, which are similar to the conventional database schemas, for Web data, and use the conventional DBMS methods to manage Web data. Others tried other ways of establishing flexible data structures, such as trees and graphs, to organize Web data and proposed corresponding retrieval languages. However, since the Web data is dynamic, which is significantly different from the conventional data in database systems, using relative fixed data schemas or structures to manage the Web data could not reflect the nature of the Web data (McHugh et al. 1997). On the other hand, the mapping of Web data into a predefined schema or structure would break down the contents of the Web data (text, hyperlinks, images, tags etc.) into separated information pieces, and intrinsic semantic relationship within a Web page and among the Web pages would be lost. In other words, Web databases alone could not provide the flexibility to reflect the dynamics of the Web data and effectively support various Web-based applications.

Unlike the conventional database management in which data models and schemas are defined, the Web community, which is a set of Web-based objects (documents and users) that has its own logical structures, is another effective and efficient approach to reorganize Web-based objects, support information retrieval and implement various applications. According to the practical requirements and concerned situations, Web community would appear as different formats.

In this book, we focus on Web community approach, i.e. establishing good Web page communities, to support Web-based data management and infor-

mation retrieval. A *Web page (data) community* is a set of Web pages that has its own logical and semantic structures. For example, a Web page set with clusters in it is a community; Web pages in a set that are related to a given Web page also form a community. The Web page community considers each page as a whole object, rather than breaking down the Web page into information pieces, and reveals mutual relationships among the concerned Web data. For instance, the system *CiteSeer* (Lawrence et al. 1999) uses search engines like *AltaVista*, *HotBot* and *Excite* to download scientific articles from the Web and exploits the citation relationships among the searched articles to establish a scientific literature searching system. This system reorganizes the scientific literature on the Web and improves the search efficiency and effectiveness. The Web page community is flexible in reflecting the Web data nature, such as dynamics and heterogeneity. Furthermore, Web page communities could be solely used by various applications or be embedded in Web-based databases to provide more flexibility in Web data management, information retrieval and application support. Therefore, database & community centered Web data management systems provide more capabilities than database-centered ones in Web-based data management.

1.3 Outline of the Book

This book will address the construction and analysis of various Web communities based on information available from Web, such as Web document contents, hyperlinks, semantics and user access logs. Web community applications are also another aspect emphasized in this book. Before presenting various algorithms, some preliminaries are provided for better understanding of the materials. Then representative algorithms for constructing and analysing various Web communities are presented and discussed. These algorithms, as well as their discussions, lead to various applications that are also presented in this book. Finally, this book will summarize the main work in Web community research and discuss future research in this area. In this book, we focus on Web community of Web documents or Web objects based on their logical, linkage and inter-relationships. The user community and social issues related to the usage of Web documents are not included. A separate volume is planned and will be devoted to user community and recommendation design based on user's access patterns or usage logs.

The book contains eight chapters.

Chap. 2 will introduce some preliminary mathematical notations and background knowledge. It covers graph and matrix representation of hyperlink information among Web documents/objects, matrix decomposition such as sin-

gular value decomposition, graph theory basis, Vector Space Model, and the Markov Model.

Chap. 3 presents Hyperlink Induced Topic Search (HITS) algorithm, its variations and related approaches.

Chap. 4 describes the popular page rank and related approaches.

Chap. 5 presents affinity and co-citation approaches for Web community analysis, including matrix-based hierarchical clustering algorithms, Co-Citation and extended algorithms etc.

Chap. 6 presents graph-based algorithms and approaches for constructing Web communities, and discusses Web community evolution patterns.

Chap. 7 introduces several techniques to either find Web communities or help analyze Web communities. This includes how to use user access patterns from Web log to explore Web community, how to use co-occurrence to enlarge Web communities, and also includes techniques for formal analysis and modeling of Web communities.

Chap. 8 presents a summary and some future directions.

1.4 Audience of the Book

This book should be interesting to both academic and industry communities for research into Web search, Web-based information retrieval and Web mining, and for the development of more effective and efficient Web services and applications.

This book has the following features:

- It systematically presents, describes and discusses representative algorithms for Web community construction and analysis.
- It highlights various important applications of the Web community.
- It summarizes the main work in this area, and identifies several research directions that readers can pursue in the future.

2 Preliminaries

This chapter briefly presents some preliminary background knowledge for better understanding of the succeeding chapters. The matrix model of hyperlinks is introduced in Sect. 2.1. Some matrix concepts and theories commonly used in matrix-based analysis are presented accordingly, especially Sect. 2.2 introduces concepts of matrix eigenvalue and eigenvector; Sect. 2.3 mainly introduces matrix norm, gives some commonly used matrix norms and their properties; singular value decomposition (SVD) of matrix is discussed in Sect. 2.4. Similarity measure of two vectors in vector space is introduced in Sect. 2.5. The last two sections, Sect. 2.6 and 2.7 are dedicated respectively to graph theory basics and the Markov chain.

2.1 Matrix Expression of Hyperlinks

The Matrix model has been widely used in many areas to model various actual situations, such as the relationship between a set of keywords and a set of documents, where keywords correspond to the columns of a matrix and documents correspond to the rows of the matrix. The intersection element value of the matrix indicates the occurrence of a keyword in a document, i.e. if a keyword is contained in a document, the corresponding matrix element value is 1, and otherwise 0. Of course, the matrix element values could also more precisely indicate the relationship between two concerned sets of objects. For example, for the keyword-document matrix, an element value could indicate the weight of a keyword that occurs in a document, not just 1 or 0. Similarly, for the pages in a concerned Web page set, such as the set of pages returned by a search engine with respect to a user's query or all the pages in a Web site, the relationship between pages via their hyperlinks can also be expressed as a matrix. This hyperlink matrix is usually called an *adjacency matrix*.

Without loss of generality, we can suppose the adjacency matrix is an $m \times n$ matrix $A = [a_{ij}]_{m \times n}$. Usually, the element of A is defined as follows (Kleinberg 1999): if there is a hyperlink from page i to page j ($i \neq j$), then the value of a_{ij} is 1, and otherwise 0. For the situation $i = j$, the entry a_{ij} is usually set to 0.

But in some cases, it could be set to other values depending on how the relationship from page i to itself is considered. For example, if a_{ii} is set to 1, it could mean that page i definitely has a relationship to itself. If this adjacency matrix is used to model the hyperlinks among the pages in the same page set, the values of parameters m and n are the same, which indicates the number of pages in the page set (set size). In this case, the ith row of the matrix, which is a vector, represents the out-link (i.e. the hyperlink from a page to other pages) relationships from page i to other pages in the page set; the ith column of the matrix represents the in-link (i.e. the hyperlink to a page from other pages) relationships from other pages in the page set to page i.

However, if the adjacency matrix is used to model the hyperlinks between the pages that belong to two different page sets, the values of parameter m and n usually are not the same unless the numbers of pages in these two sets are the same. Suppose one page set is A with the size of m, another page set is B with the size of n. In this case, the ith row of the adjacency matrix represents the out-link relationships from the page i in set A to all the pages in set B; the jth column of the matrix represents the in-link relationships from all the pages in set A to the page j in set B.

Although the above adjacency matrix expression is intuitive and simple, the values of the matrix elements only indicate whether there exist hyperlinks between pages (i.e. value 1 of a matrix element indicates that there is a hyperlink between two pages that correspond to this element, and value 0 indicates that there is no hyperlink between two pages). In hyperlink analysis, this matrix expression can also be extended to represent semantics of hyperlinks. In this case, the values of the matrix elements are not required to be either 1 or 0. The actual element value depends on the particular situations where the matrix expression is applied. For example, the correlations between pages can be expressed in a matrix, where the value of a matrix element a_{ij}, which is between 0 and 1, indicates the correlation degrees from page i to page j, and the matrix is non-symmetric. The similarity between pages can also be expressed in a matrix in a similar way, except that the similarity matrix is usually a symmetric one. The method of determining the matrix entries depends on how the relationship between the concerned objects and the application requirements are modelled. In the following chapters, more examples of how to construct a matrix for different applications are presented. No matter which matrix will be constructed, the idea is the same, which is that the matrix model is a framework with the following requirements to be met:

1. A data (information) space is constructed. For example, in a conventional database system, the data space might be the whole documents within it. In the context of Web, a data space might be a set of Web pages. But con-

structing a data space for different Web application requirements is more complex.
2. Two sets of information entities (objects), denoted as E_1 and E_2, within the constructed data space are identified. One set should be a reference system to another. That means the relationships between entities in E_1 are determined by those in set E_2, and vice versa. For example, E_1 could be a set of documents; E_2 could be a set of keywords.
3. An original correlation expression between entities that belong to different sets E_1 and E_2 is defined and modeled into a matrix. The correlation could be conveyed by correlation information, such as keyword occurrence and hyperlinks between pages.

From adjacency matrix, each page could be considered as a row or column of the matrix. In other words, each page is represented as a vector. Therefore, it is possible to use a vector model, which is usually used in traditional information retrieval, to reveal relationships between pages, such as similarity and cluster. Furthermore, it is also possible to find deeper and global relationships among the pages through mathematical operations on the matrix, such as computing eigenvalues and eigenvectors, and singular value decomposition. The hyperlink matrix could also be directly used for other purposes, such as Web page clustering through matrix permutation and partitioning. More details of matrix construction and applications in the context of the Web will be seen in the succeeding chapters.

2.2 Eigenvalue and Eigenvector of the Matrix

Eigenvalue and eigenvector are two commonly used concepts in matrix model application. Some basic knowledge of these two concepts is presented in this section. For further details, readers could refer to linear algebra texts such as (Golub and Loan 1993; Strang 1993; Datta 1995).

Let matrix A be an $n \times n$ matrix with real numbers as entries. An *eigenvalue* of A is a number λ with the property that for some vector v, we have $Av = \lambda v$. Such a vector v is called an *eigenvector* of A associated with the eigenvalue λ. The eigenvalue with maximum absolute value is called *principal eigenvalue*, and its corresponding eigenvector is called *principal eigenvector*.

The set of all eigenvectors associated with a given eigenvalue λ forms a subspace of $\mathbf{R}^n$, and the dimension of this space will be referred to as the *multiplicity* of λ. If A is a symmetric matrix, then A has at most n distinct eigenvalues, each of them is a real number and the sum of their multiplicities is exactly n. We denote these eigenvalues of matrix A by $\lambda_1(A)$, $\lambda_2(A)$, …, $\lambda_n(A)$, listing each a number of times equal to it multiplicity.

For symmetric matrix A, if we choose an eigenvector $v_i(A)$ associated with each eigenvalue $\lambda_i(A)$, then the set of vectors $\{v_i(A)\}$ forms an orthonormal basis of $\mathbf{R}^n$, that is each vector is a unit vector and each pair of vectors is orthogonal, i.e. the inner product of each vector pair is 0.

A matrix M is orthogonal if $M^T M = I$, where M^T denotes the transpose of the matrix M, and I denotes the identical matrix, i.e. a diagonal matrix with all diagonal entries equal to 1. If A is a symmetric $n \times n$ matrix, Λ is the diagonal matrix with diagonal entries $\lambda_1(A)$, $\lambda_2(A)$, …, $\lambda_n(A)$, and M is the matrix with column equal to $v_1(A)$, $v_2(A)$, …, $v_n(A)$. Then it is easy to verify that M is an orthogonal matrix and we have $M\Lambda M^T = A$. Thus the eigenvalues and eigenvectors provide a useful "normal form" representation for symmetric square matrix in terms of orthogonal and diagonal matrices. In fact, there is a way to extend this type if normal form to matrices that are neither symmetric nor square, such as the singular value decomposition (SVD) of matrix that will be discussed later in this chapter.

2.3 Matrix Norms and the Lipschitz Continuous Function

A matrix norm, denoted by $\|\cdot\|$, is a measurement of a matrix. It is very similar to the absolute value definition of a real number. Informally and intuitively, a norm of two matrices' difference $\|A - B\|$ can be understood as the distance between these two matrices A and B.

There are many ways to define a norm of a matrix. For a give matrix A, a matrix norm $\|A\|$ is a *nonnegative* number associated with A. The norm should have the following properties:

1. $\|A\| > 0$ when $A \neq 0$ and $\|A\| = 0$ iif $A = 0$.
2. $\|kA\| = |k| \, \|A\|$ for any scalar k.
3. $\|A + B\| \leq \|A\| + \|B\|$.
4. $\|AB\| \leq \|A\| \, \|B\|$.

Let $A = (a_{ij})$ be an $m \times n$ real matrix. The commonly used matrix norms are defined as follows:

- Frobenius norm of a matrix A

$$\|A\|_F = [\sum_{j=1}^{n} \sum_{i=1}^{m} |a_{ij}|^2]^{1/2} .$$

- 1-norm of a matrix A

$$\|A\|_1 = \max_{1\le j\le n} \sum_{i=1}^{m} |a_{ij}|.$$

- ∞-norm of a matrix A

$$\|A\|_\infty = \max_{1\le i\le m} \sum_{j=1}^{n} |a_{ij}|.$$

- 2-norm of a matrix A

$$\|A\|_2 = \text{(maximum eigenvalue of } A^TA)^{1/2}.$$

$\|A\|_1$, $\|A\|_2$ and $\|A\|_\infty$ satisfy the following inequality

$$\|A\|_2^2 \le \|A\|_1 \|A\|_\infty.$$

The Lipschitz continuous function is an important function in function analysis. We give its definition here for further reference in the following chapters. Formally, a Lipschitz continuous function is a function $f(x)$ that for all x, y, we have $|f(x) - f(y)| \le L|x - y|$. L is called the Lipschitz constant. This is certainly satisfied if f has a first derivative bounded by L. Note that, even if f does not have uniformly bounded derivatives over the entire real line, the theorem will also hold so long as the derivatives are bounded within the applicable domain.

2.4 Singular Value Decomposition (SVD) of a Matrix

The singular value decomposition (SVD) of a matrix is defined as follow: let $A = [a_{ij}]_{m\times n}$ be a real $m \times n$ matrix. Without loss of generality, we suppose $m \ge n$ and the rank of A is $rank(A) = r$. Then there exist orthogonal matrices $U_{m\times m}$ and $V_{n\times n}$ such that

$$A = U\begin{pmatrix}\Sigma_1\\ 0\end{pmatrix}V^T = U\Sigma V^T \tag{2.1}$$

where $U^TU = I_m$, $V^TV = I_n$, $\Sigma_1 = diag(\sigma_1,...,\sigma_n)$, $\sigma_i \ge \sigma_{i+1} > 0$ for $1 \le i \le r-1, \sigma_j = 0$ for $j \ge r+1$, Σ is a $m \times n$ matrix, U^T and V^T are the transpositions of matrices U and V respectively, I_m and I_n represent $m \times m$ and $n \times n$ identity matrices separately. The *rank* of A indicates the maximal number of independent rows or columns of A. Equation (1) is called the singular value decomposition of matrix A. The singular values of A are di-

agonal elements of Σ (i.e. $\sigma_1, \sigma_2, ..., \sigma_n$). The columns of U are called left singular vectors and those of V are called right singular vectors (Golub and Loan 1993; Datta 1995). For example, let

$$A = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \end{pmatrix}_{3\times 2},$$

then the SVD of A is $A = U\Sigma V^T$, where

$$U = \begin{pmatrix} 0.3381 & 0.8480 & 0.4082 \\ 0.5506 & 0.1735 & -0.8165 \\ 0.7632 & -0.5009 & 0.4082 \end{pmatrix}_{3\times 3}, \quad V = \begin{pmatrix} 0.5969 & -0.8219 \\ 0.8219 & 0.5696 \end{pmatrix}_{2\times 2},$$

$$\Sigma = \begin{pmatrix} 6.5468 & 0 \\ 0 & 0.3742 \\ 0 & 0 \end{pmatrix}_{3\times 2}$$

and the singular values of A are 6.5468 and 0.3742.

The SVD could be used effectively to extract certain important properties relating to the structure of a matrix, such as the number of independent columns or rows, eigenvalues, approximation matrix and so on (Golub and Loan 1993; Datta 1995). Since the singular values of A are in non-increasing order, it is possible to choose a proper parameter k such that the last r-k singular values are much smaller than the first k singular values, and these k singular values dominate the decomposition. The next theorem reveals this fact.

Theorem [Eckart and Young]. Let the SVD of A be given by equation (2.1) and $U = [u_1, u_2, \ldots, u_m]$, $V = [v_1, v_2, \ldots, v_n]$ with $0 < r = rank(A) \le \min(m,n)$, where u_i, $1 \le i \le m$ is an m-vector, v_j, $1 \le j \le n$ is an n-vector and

$$\sigma_1 \ge \sigma_2 \ge \ldots \ge \sigma_r > \sigma_{r+1} = \ldots = \sigma_n = 0.$$

Let $k \le r$ and define

$$A_k = \sum_{i=1}^{k} u_i \cdot \sigma_i \cdot v_i^T . \tag{2.2}$$

Then

1. $rank(A_k) = k$;
2. $\min_{rank(B)=k} \| A - B \|_F^2 = \| A - A_k \|_F^2 = \sigma_{k+1}^2 + \ldots + \sigma_r^2$,
3. $\min_{rank(B)=k} \| A - B \|_2 = \| A - A_k \|_2 = \sigma_{k+1}$.

The theorem proof can be found in (Datta 1995). This theorem indicates that matrix A_k, which is constructed from partial singular values and vectors (see Fig. 2.1), is the best approximation to A (i.e. conclusions 2 and 3 of the Theorem) with rank k (conclusion 1 of the Theorem). In other words, A_k captures the main structure information of A and minor factors in A are filtered. This important property could be used to reveal the deeper relationships among the matrix elements, and implies many potential applications provided the original relationships among the considered objects (such as Web pages) can be represented in a matrix. Since $k \leq r$ and only partial matrix elements are involved in constructing A_k, the computation cost of an algorithm based on A_k could be reduced.

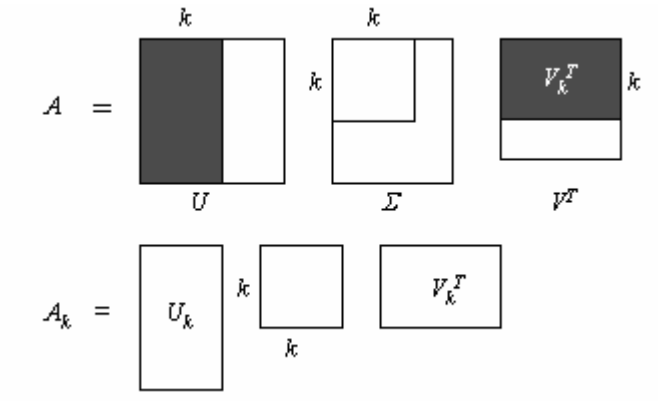


Fig. 2.1. Construction of Approximation

The SVD of matrix was successfully applied in textual information retrieval (Deerwester et al. 1990; Berry et al. 1995), and the corresponding method is called Latent Semantic Indexing (LSI). In the LSI, the relationships between documents and terms (words) are represented in a matrix, and SVD is used to reveal important associative relationships between term and documents that are not evident in individual documents. As a consequence, an intelligent indexing for textual information is implemented. Papadimitriou et al (Papadimitriou et al. 1997) studied the LSI method using probabilistic approaches and indicated that LSI in certain settings is able to uncover semantically "meaningful" associations among documents with similar patterns of term usage, even when they do not actually use the same terms. This merit of SVD, as indicated in its application to textual information retrieval, could also be applied to Web data to find deeper semantic relationships provided the Web data is correlated with each other through a certain correlation pattern, such as a hyperlink pattern. The correlation pattern between the considered objects (e.g. Web pages) is the base where the SVD is applied.

2.5 Similarity in Vector Space Models

We discuss the similarity here within a framework of vector space model, i.e., the concerned objects such as documents and Web pages are represented as vectors. These vectors could be rows/columns of a matrix, or just individual vectors.

The representation of concerned objects by vectors in Euclidean space allows one to use geometric methods in analyzing them. At the simplest level, the vector representation naturally suggests numerical similarity metrics for objects based on Euclidean distance or the vector inner product. The representative metric is the cosine measure which is defined as follow: let x and y be two vectors with the dimension m, their *cosine similarity* is

$$sim(x, y) = \frac{x \cdot y}{\| x \| \| y \|},$$

where the inner product $x \cdot y$ is the standard vector dot product defined as

$$x \cdot y = \sum_{i=1}^{m} x_i y_i ,$$

and the norm in the denominator is defined as

$$\| x \| = \sqrt{x \cdot x} .$$

This similarity metric is named cosine similarity because it is simply the cosine of the angle between two vectors x and y.

The above numerical similarity metric suggests natural approaches for similarity based indexing in information retrieval - by representing queries as vectors and searching for their nearest neighbours in a collection of concerned objects, such documents and Web pages, which are also represented as vectors. The similarity could also be used for clustering. Of course, in any application with a huge number of vector dimensions, these vector operations can be a problem not only from point of view of computational efficiency, but also because the huge dimension leads to sets of vectors with very spare patterns of non-zeroes, in which the relationships among concerned objects can be difficult to detect or explore. Therefore, an effective method for reducing the dimension of the set of vectors without seriously distorting their metric structure offers the possibility of alleviating both these problems.

2.6 Graph Theory Basics

A graph is a commonly used model for analyzing relationship between Web pages in terms of hyperlink. A *graph* $G = (V, E)$ is defined as a set of nodes V

and a set of edges E. Each element $e \in E$ represents a connection between an unordered node pair (u, v) in V. In the context of hyperlink, Web pages could be modeled as nodes of a graph, and hyperlinks between pages as edges of the graph.

A graph is *connected* if the node set cannot be partitioned into components such that there are no edges whose connected nodes occur in different components.

A *bipartite graph* $G = (V_1, V_2, E)$ consists of two disjoint sets of nodes V_1, V_2 such that each edge $e \in E$ has one node from V_1 and the other node from V_2. A bipartite graph is *complete* if each node in V_1 is connected to every node in V_2. A *matching* is a subset of edges such that for each edge in the matching, there is no other edge that shares a node with it. A *maximum matching* is a matching of largest cardinality.

A *weighted graph* is a graph with a (non-negative) weight w_e for every edge e. Given a weighted graph, the *minimum weight maximum matching* is the maximum matching with minimum weight.

A *directed graph* is a graph with an edge being an ordered pair of nodes (u, v), representing a connection from u to v. Usually the edge of an ordered pair of nodes (u, v) in a directed graph is represented as an arrow from u to v. A directed path is said to exist from u to v if there is a sequence of nodes $u = w_0, \ldots, w_k = v$ such that $(w_i; w_{i+1})$ is an edge, for all $i = 0, \ldots, k\text{-}1$. A *directed cycle* is a non-trivial directed path from a node to itself. A strongly connected component of a graph is a set of nodes such that for every pair of nodes in the component, there is a directed path from one to the other. A *directed acyclic graph* (DAG) is a directed graph with no directed cycles. In a DAG, a *sink node* is one with no directed path to any other node.

More discussions of graph model in the context of the Web can be found in (Broder et al. 2000). Many examples of graph model applications in Web community analysis will be seen in the succeeding chapters.

2.7 Introduction to the Markov Model

A (homogeneous) Markov chain for a system is specified by a set of states $S = \{1, 2, \ldots, n\}$ and an $n \times n$ non-negative, stochastic matrix M. A stochastic matrix is a matrix satisfying that the sum of each row is 1. The system begins in some start state in S and at each step moves from one state to another state. This transition is guided by M: at each step, if the system is in state i, it moves to state j with probability M_{ij}. If the current state is given as a probability distribution, the probability distribution of the next state is given by the product of the vector representing the current state distribution and M. In general, the

start state of the system is chosen according to some distribution x, which is usually a uniform distribution, on S.

After t steps, the state of the system is distributed according to xM^t. Under some conditions on the Markov chain, from an arbitrary start distribution x, the system eventually reaches a unique fixed point where the state distribution does not change. This distribution is called the stationary distribution. It can be shown that the stationary distribution is given by the principal eigenvector y of M, i.e., $My = \lambda y$ where λ is the principal eigenvalue of M. In practice, a simple power-iteration algorithm can quickly obtain a reasonable approximation to y.

In practical use, a random walk model based on a graph can also be represented as a Markov chain model under some assumptions. For example, a Web surfer surfs along hyperlinks between pages. If the surfer only takes one of two actions: going forward to another page along the hyperlink in the current visiting page, or jumping randomly to other pages in the concerned page set, the surfer's behaviour can be modelled as a Markov chain, where the states are the possibilities of pages the surfer might jump to. The final probability distribution of this Markov chain indicates the page ranks within the concerned Web page set. In a similar way, Markov chain can be used in many situations to establish simulation and analysis models.

3 HITS and Related Algorithms

Hyperlink Induced Topic Search (HITS) is a representative of algorithms that reveals Web page relationships conveyed by hyperlinks. It aroused investigation on constructing Web page community from hyperlink information. As the beginning of Web community discussion, this chapter discusses this hyperlink-based algorithm, its improvements, variations, and related issues. Some in-depth analyses of HITS are presented as well. Sect. 3.1 gives the original algorithm of HITS. The stability issues of HITS are discussed in Sect. 3.2, which is the basis of further discussions in this chapter. Randomized, subspace and weighted HITS are discussed respectively in Sect. 3.3, 3.4 and 3.5. In Sect. 3.6 and 3.7, two algorithms are discussed, which incorporate page content information to improve HITS. Before discussing other HITS related algorithms, Sect. 3.8 gives an in-depth analysis of HITS from matrix analysis point of view to reveal some features of HITS. After that, Sect. 3.9 discusses a special case of "nullification" in HITS, and gives another approach to avoid this abnormality and improve HITS accordingly. Sect. 3.10 gives another way to improve HITS by eliminating noise pages from the page base set of HITS, rather than directly tuning the HITS. In the last section of this chapter Sect. 3.11, a stochastic approach SALSA is discussed to improve HITS.

3.1 Original HITS

The Hyperlink Induced Topic Search (HITS) algorithm is essentially a link-based approach that intends to find authority and hub pages from a link induced Web graph. Authorities are those pages that provide the best source of information on a given topic, while hubs are those pages that provide collections of links to authorities (Kleinberg 1998). Hubs and authorities exhibit a mutually reinforcing relationship: A good hub is a page that points to many good authorities; a good authority is a page that is pointed to by many good hubs. The algorithm was implemented by IBM Almaden Research Centre in Clever, a prototype search engine, and was named the "Clever Algorithm" in an article of *Scientific American*. This algorithm name is also used in some research papers.

The reason for conducting hyperlink analysis is that hyperlink structure contains an enormous amount of latent human annotation that can help automatically infer notations of authority. Specifically, the creation of a hyperlink by the author of a Web page represents an implicit endorsement of the page being pointed to (Chakrabarti et al. 1999). We can get a better understanding of the relevance and quality of the Web's contents by mining the collective judgment in the set of hyperlinks.

The HITS algorithm consists of two main steps: focused sub-graph construction and iterative calculation of authority and hub weight.

Focused Sub-Graph Construction. The Web could be viewed as a directed graph, with nodes representing pages and directed edges representing hyperlinks. Since an attempt to analyse the link structure of the entire Web would prove computationally costly and time consuming for a search topic that is specified by one or more query terms, the HITS algorithm is to be run on a sub-graph. This sub-graph is constructed from a set of pages S that exhibit the following properties:

- The set S is relatively small.
- The set S is rich in relevant pages.
- The set S contains most of the strongest authorities.

For a query σ, the sub-graph construction starts with forming a root set R_σ. Initially, a collection of the top t pages, where t is typically set to 200, for the query σ is made from the results returned by the text-based search engineering such as *AltaVista* and *Google*. These top t pages form the *root set* R_σ. The root set satisfies the first two properties above, but usually does not satisfy the third. As Kleinberg observed, there are often extremely few links between pages in the root set. To increase the number of strong authorities in the sub-graph, we can expand the root set R_σ to form the set of pages S_σ, from which the required sub-graph is constructed.

The root set R_σ is expanded to form S_σ by including pages pointed to by or pointing to pages in the root set. All the pages pointed to by the pages in R_σ are include in S_σ. A restriction for this procedure is that any page in R_σ can bring in at most d pages pointing to it. Here d is a predefined number, which was set to 50 in Kleinberg's experiment (Kleinberg 1998). The resulting set S_σ is now referred to as the *base set* for the query σ. The experiments conducted by (Kleinberg 1998) found the base set S_σ with $t = 200$ and $d = 50$ typically possesses all three properties above and is generally 1000 - 5000 pages.

Once the base set has been grown, a sub-graph can be constructed. Within this sub-graph, two kinds of links should be considered. A link is considered as *transverse* if it is between pages with different domain names; a link is

considered as *intrinsic* if it is between pages with the same domain name at the first level of the URL. As many intrinsic links in a particular site exist for navigation purpose only, they convey less semantic information in determining authorities. By deleting all intrinsic links in the sub-graph, a focused sub-graph G_σ for the query σ is constructed.

Iterative Calculation of Authority and Hub Weight. HITS associates each page p in the sub-graph G_σ a non-negative *authority weight* $x^{<p>}$ and a non-negative hub weight $y^{<p>}$. The mutually reinforcing relationship between authority and hub weights can be numerically expressed as follows: if a page p points to many pages with large authority weights, then it should receive a large hub weight; and if p is pointed to by many pages with large hub weights, it should receive a large authority weight. Given weights $x^{<p>}$ and $y^{<p>}$ for all pages in G_σ, this mutual relationship is realised by the following two operations:

I operation updates the authority weights:

$$x^{<p>} = \sum_{\forall q:q\to p} y^{<q>} ,$$

where $q \to p$ means page q points to page p.

O operation updates the hub weights:

$$y^{<p>} = \sum_{\forall q:p\to q} x^{<q>} .$$

The operations are performed in an alternating way to find convergence. The set of weights $\{x^{<p>}\}$ and $\{y^{<p>}\}$ can be expressed as vectors x and y respectively. Each vector has a coordinate for each page in G_σ. To guarantee convergence of the iterative operations, after each iterative operation, the vectors x and y are normalized so that their squares sum to 1. This iterative calculation of authority and hub weight can be expressed in the following *iterate* algorithm:

Iterate (G,k) {
  G: a collection of n linked pages in the sub-graph G_σ
  k: a natural number for the times of iterative operations
  Let z denote the initial vector $(1,1, \ldots, 1) \in \boldsymbol{R}^n$
  Set $x_0 = z$
  Set $y_0 = z$
  For $i = 1,2, \ldots, k$ {
    Apply the I operation to (x_{i-1}, y_{i-1}) and obtain a new authority weight vector x'_i.
    Apply the O operation to (x'_i, y_{i-1}) and obtain a new hub weight vector y'_i.
    Normalize x'_i and obtain x_i.
    Normalize y'_i and obtain y_i.

}
Return (x_k, y_k).
}

The pages with top c authority weights in x_k are reported as authority pages, while the pages with top c hub weights in y_k are reported as hub pages for the given query σ. In (Kleinberg 1998), the parameter c is typically set to 10.

There is also a more compact way to write I and O operations. Let us number the pages $\{1,2,\ldots, n\}$ and define their adjacency matrix A to be the $n \times n$ matrix whose $(i.j)^{th}$ entry is equal to 1 if page i links to page j, or 0 otherwise. We write the set of all authority values as a vector $x=(x_1, x_2, \ldots, x_n)$ and the set of all hub values as an another vector $y=(y_1, y_2, \ldots, y_n)$. Then the I and O operations can be written respectively as

$$x = A^T y, \qquad y = Ax.$$

Unwinding these one-step operations further, we get

$$x = A^T y = (A^T A)x, \qquad y = Ax = (AA^T)y.$$

Thus, after multiple iterations, the vector x is precisely the result of applying the power iteration operations to the matrix $A^T A$. As indicated by linear algebra theory (Golub and Loan 1993), this sequence of iterations, when normalized, converges to the principal eigenvector of $A^T A$. Similarly, the sequence of values for the normalized vector y converges to the principal eigenvector of AA^T.

3.2 The Stability Issues

The initial results of the HITS algorithm were promising in some cases (Bharat and Henzinger 1998; Kleinberg 1998; Chakrabarti et al. 1999; Borodin et al. 2001). However, there are still stability issues with this algorithm to be addressed and solved. Mainly these issues are: topic drift, mutually reinforcing, topic variety, and topic generalization.

Topic Drift. Topic drift is a phenomenon where the highly ranked authorities and hubs are not relevant to the original query topic. For example, (Bharat and Henzinger 1998)ran the HITS algorithm on the query "jaguar and car", the computation drifted to the general topic "car" and returned the home pages of different car manufacturers as top authorities and lists of car manufacturers as the best hubs. It was found through experimentation (Bharat and Henzinger 1998) that the base set of HITS contains pages not relevant to the query topic. If these pages are well connected or more strongly interconnected, they will dominate the HITS operation and the topic drift occurs. A

striking example of this phenomenon is provided by (Cohn and H 2000). They applied the HITS algorithm to the query topic "jaguar". The operation returned a collection of pages about services in Cincinnati. The cause of this behaviour is the *Cincinnati Enquirer* newspaper, which has many articles about the Cincinnati Bengals and Jacksonville Jaguars football teams. Each article contains the same set of pointers to services provided by the newspaper.

Mutual Reinforcing Occurs. Mutually reinforcing occurs between hosts where a set of pages on one host point to a single page on a second the host. This inflates the authority value of the single page, and subsequently improves the hub values of those pages that reference it. The reverse can also occur if a single page on a host points to many pages on another host. As the page has a large out-degree it gains an unduly large hub value. This subsequently increases the authority score of every page it links to. The mutually reinforcing occurs typically because designers of individual Web page copy the page design from a master copy which contains the links.

Topic Variety. Chakrabarti et al (Chakrabarti et al. 1999) noticed that if a page has discussion on various topics within the same page, then the outgoing links will link to different topics depending on their position within the initial page. If this page has a large out-degree, it will receive a large hub weight that flows on a high authority weights for referenced pages regardless of their relevance to the initial query topic.

Topic Generalization. This issue was raised by (Chakrabarti et al. 1999), which was also addressed by (Kleinberg 1998) and (Gibson et al. 1998). If the query topic is too narrow, then the HITS algorithm frequently returns good resources for more general topics. For example, for the query topic "jaguar and car" the HITS yielded the results, where the general topic "car" overwhelms the more specific query. This behaviour is sometimes reversed. The reversal occurs when the returned list of pages is dominated by more specific pages. This set of specific pages exhibits a greater density of linkage within its neighborhood of the Web graph and thus the HITS algorithm converges on it.

These stability issues of the HITS algorithm shows that the purely link based approach is not without its drawbacks. Therefore, after the HITS was proposed, much work has been done to improve it or combine it with other approaches. These HITS variations are discussed in the following sections.

3.3 Randomized HITS

Considering the hyperlinks between pages as the random walk paths for Web surfers (users), (Ng et al. 2001)proposes an improved HITS algorithm named randomized HITS.

This algorithm is based on an assumption that there is a random surfer who is able to follow hyperlinks both forwards and backwards. More precisely, the surfer starts from a randomly chosen page, and visits a new Web page at every time step. Every time step, with the probability of ε, he jumps to a new Web page chosen uniformly at random; otherwise, he checks if it is an odd time step or an even time step. If it is an odd time step, he then follows a randomly chosen out-link from the current page; if it is an even time step, then he traverses a random in-link of the current page. Thus, the random surfer alternately follows links forwards and backwards, and occasionally (with the probability of ε) "resets" and jumps to a page chosen uniformly at random.

This process simulates a random walk on Web pages, and the stationary distribution on odd time steps is defined to be the authority weights. Intuitively, let t be a very large odd number, the authority weight of a page is the chance that the surfer visits that page on time step t. Similarly, the stationary distribution on even time steps is defined to be the hub weights. These quantities can also be written in the following mathematical expressions:

$$x^{(t+1)} = \varepsilon\, \vec{u} + (1-\varepsilon) A_{row}^{T} y^{(t)},$$
$$y^{(t+1)} = \varepsilon\, \vec{u} + (1-\varepsilon) A_{col}\, x^{(t+1)}$$

where $\vec{u}$ is the vector of all ones, A_{row} is the same as adjacency matrix A in Sect. 3.1 with its rows normalized to sum to 1, and A_{col} is A with its columns normalized to sum to 1. This iteration operation is similar to the original HITS. (Ng et al. 2001) indicated that iterating these equations will cause $x^{(t)}$ and $y^{(t)}$ to converge to the odd-step and the even-step stationary distributions.

The experimental results in (Ng et al. 2001) empirically show that this algorithm produces more stable results than the original HITS on both academic citation and Web query data. The result stability means some minor changes to the linkage structure of base set could not dramatically change authority and hub weights of the pages. The above randomized HITS improves the stability of the original HITS algorithm, i.e. the randomized HITS is insensitive to small perturbations.

3.4 Subspace HITS

This algorithm, which was proposed by (Ng et al. 2001), is another way of improving the stability of HITS. As indicated in Sect. 3.1, the final authority vector x and hub vector y of the HITS algorithm are the principal eigenvectors of matrices A^TA and AA^T respectively. Sometimes, however, individual eigenvectors of a matrix may not be stable, but *subspaces* spanned by eigenvectors may be. Theoretically, if the eigengap between the k-th and $k+1$-st eigenvalues is large, then the subspace spanned by the first k eigenvectors will be stable (Stewart and JG 1990). That's the base of constructing a subspace that is spanned by some eigenvectors to obtain authority scores instead of examining eigenvectors individually.

The subspace HITS algorithm for calculating authority scores (or hub scores) is described as the following procedure (Ng et al. 2001), where $f(\cdot)$ is a non-negative, monotonically increasing function that will be specified later:

1. Find the first k eigenvectors $v_1, \ldots, v_k$ of A^TA (or AA^T for hub weights), and their corresponding eigenvalues $\lambda_1, \ldots \lambda_k$. In the case of repeated eigenvalues, the eigenvectors are chosen orthogonal to each other.
2. Let e_j be the j-th basis vector (whose j-th element is 1, and all other elements 0). Calculate the authority scores

$$x_j = \sum_{i=1}^{k} f(\lambda_i)(e_j^T v_i)^2.$$

This is the square of the length of the projection of e_j onto the subspace spanned by $v_1, \ldots, v_k$, where the projection in the v_i direction is "weighted" by $f(\lambda_i)$.

There are many choices for function f:

- If we define $f(\lambda) = 1$ when $\lambda \geq \lambda_{max}$ and $f(\lambda) = 0$ otherwise, we get back the original HITS algorithm;
- If we take $k = n$, here n is the number of pages in the base set, and $f(\lambda) = \lambda$, this choice corresponds to simple citation counting;
- If we define $f(\lambda) = 1$, the authority scores depend only on the subspace spanned by the k eigenvectors.

For computational reasons, it is practical to use $k < n$ eigenvectors as an approximation to using the full set, which drops the eigenvectors with the smallest weights. In the experiments of (Ng et al. 2001), $k = 20$ and function f is defined as $f(\lambda) = \lambda^2$.

This method gives a way of automatically combining multiple eigenvectors into a single measure of authoritativeness for each page. In general, subspaces

are more stable than individual eigenvectors (see (Stewart and JG 1990)), and therefore it is reasonable to expect that Subspace HITS will do better than HITS in certain cases. The experimental results presented in (Ng et al. 2001) demonstrate this expectation, in which the results returned by subspace HITS are more stable than those returned by HITS.

(Ng et al. 2001) also theoretically proved that the subspace HITS algorithm is stable, i.e. for the minor perturbation of matrix A^TA (or AA^T for hub weights), the effect of this perturbation to the authority scores (or hub scores) can be bounded within a small area. The following theorem (Ng et al. 2001) illustrates this fact:

THEOREM. *Let f be Lipschitz continuous with Lipschitz constant L, and let k = n. Let the matrix* $S = A^TA$ *be perturbed according to* $S' = S + E$, *where* $||E||_F = \varepsilon$ *(E is symmetric,* $||\cdot||_F$ *is the Frobenius norm of a matrix). Then the change in the vector of authority scores is bounded as follows:*

$$||x - x'||_2 \leq L\varepsilon.$$

The proof of the theorem is given in (Ng et al. 2001).

3.5 Weighted HITS

The HITS algorithm performs well in some cases, but it ignores the text in Web pages that describe the query topics. (Chakrabarti et al. 1998) proposed an improvement of HITS by incorporating text around the <*a href*> tags of a page. The text is used to weight the links between pages. This weighted HITS contains two modifications of the original HITS: the first one is modification of the focused sub-graph construction, and the second one is weighted adjacency matrix.

As stated in Sect. 3.1, the focused sub-graph construction contains two steps: gathering topic related pages returned by a search engine to form a root set; expanding the root set to form a base set by adding pages that are pointed to by and point to pages in the root set. (Chakrabarti et al. 1998) improved this construction by using the second step twice and called the page set obtained in this way the *augmented set* rather than the base set. Therefore, the augmented set includes all pages that are link-distance two or less from at least one page in the root set. As indicated in (Chakrabarti et al. 1998), the augmented set contained between a few hundred and 3000 distinct pages, depending on the query topic.

The HITS algorithm can be expressed as iteration operations based on the adjacency matrix which entries are either 1 or 0. In practice, it is very likely that the text around the <*a href*> tag that introduces a link to a page *p* is descriptive of the contents of *p*, while the <*a href*> tag is not in *p* but in pages

pointing to p. It is reasonable to assume that if a text descriptive of the query topic occurs in the text around an <*a href*> tag in a good hub, it will enforce our believe that a page pointed to by this link is an authority on the topic. Therefore, this kind of link should have a larger weight than other links, and the corresponding entry value in the adjacency matrix should be more than 1. The idea of incorporating text in (Chakrabarti et al. 1998) is to look on either side of the *href* tag for a window of B bytes, where B is 50 which is determined through experiments. This text range is called the *anchor window*, which includes the text between the <*a href* = "…"> and </*a*> tags. Suppose there is a link from page p to q, let $n(t)$ denote the number of matches between terms in the query topic description and those in the anchor window, then the entry value of the adjacency matrix A is modified as

$$A(i, j) = 1 + n(t),$$

where a term is specified as a contiguous string of words.

(Bharat and Henzinger 1998) proposed another weighted HITS, addressing the mutually reinforcing relationships issue indicated in Sect. 3.2. Mutually reinforcing relationships between hosts gives undue weight to the opinion of a single page. Therefore, it is ideal that all the pages on a single host have the same influence on the page they are connected to, as a single page would. To this end, (Bharat and Henzinger 1998) gave fractional weights to links in the following way.

If there are k hyperlinks from pages on the first host to a single page on the second host, each link is given an authority weight of $1/k$. This weight is used when computing the authority score of the page on the second host. Similarly, if there are l links from a single page on the first host to a set of pages on the second host, each link is given a hub weight of $1/l$, which is used when computing the hub score of the page on the first host. Additionally, the isolated pages in the base set are discarded. This leads to the following weighted HITS:

Weighted I operation, updating the authority weights

$$x^{<p>} = \sum_{\forall q:q\to p} y^{<q>} \times auth\_weight(q, p).$$

Weighted O operation, updating the hub weights

$$y^{<p>} = \sum_{\forall q:p\to q} x^{<q>} \times hub\_weight(p, q).$$

The vectors x and y are normalized after each iteration. Similar to the proof in (Kleinberg 1998), (Bharat and Henzinger 1998) also proved that this weighted HITS converges, i.e., that the algorithm terminates after finite steps of iteration operations. The experimental results presented in (Bharat and Henzinger 1998) demonstrated that this weighted algorithm was effective in eliminating the mutually reinforcing relationship problem in all the cases where they are encountered.

Although Bharat's weighted algorithm in (Bharat and Henzinger 1998) improves the original HITS, there are still some cases where these algorithms could not get satisfactory results. (Li et al. 2002) investigated a case of so called *small-in-large-out* links, which means a root page has few in-links but a large number of out-links to pages that are not relevant to query topics. (Li et al. 2002) gave a simple query *cruises*. All of the top 10 authorities are from the out-links of a root page *www.cyprus-cruises.com*, which has only 1 in-link but 271 out-links, most of which are in different domains; while the average in-degree and out-degree of the root set link are 34 and 17 respectively. The top 3 to top 10 hubs are from another root page *cyprus-car-hire.com*, which is also an out-link of the root page *www.cyprus-cruises.com*. (Li et al. 2002) noticed that these two root pages and their hyperlink neighborhood form a tightly-knit community. The hub value of the root page *www.cyprus-cruises.com* dominates the iteration operations of the Bharat's weighted HITS algorithm. As a result, the documents it points to get the largest authority values. Consequently, the in-links of root page *cyprus-car-hire.com* get the top hub values because it is assigned the largest authority value. It was found in (Li et al. 2002) that most of the hubs and authorities are of very low relevance to the query *cruises*. The major problem of the above example is that the root page *www.cyprus-cruises.com* has *small-in-large-out* links compared with the average in-degree and out-degree of root pages.

Without content analysis, both Bharat's weighted HITS and HITS can not solve the above problem as most out-links of a small-in-large-out link are in different domains. To solve this kind of problem, (Li et al. 2002) modified the HITS algorithm as another weighed HITS algorithm, in which the two-step iteration operations are as follows.

- For all pages *j* in the base set that points to page *i*,
$$a_i = \sum_j w_{a_j} \cdot h_j .$$
- For all pages j in the base set that is pointed to by page i,
$$h_i = \sum_j w_{h_j} \cdot a_j .$$

In the first equation, the value of h_j can be HITS hub weight, Bharat's HITS hub weight, or the hub weight of other HITS-based algorithms. This notation similarly applies to the value of a_j in the second equation.

In (Li et al. 2002), all the w_{h_j} values are set to 1. The setting of w_{a_j} consists of two parts:

1. Before starting a HITS-based algorithm, if there exists a root page whose in-degree is among the three smallest ones and whose out-degree is among the three largest ones, then set w_{a_j} to 4 for in-links of all the root pages.
2. Otherwise, set all w_{a_j} to 1. Run the HITS-based algorithm for one iteration without normalization. If there exists a root page whose authority value is among the three smallest ones and whose hub value is among the three largest ones, set w_{a_j} to 4 for in-links of all the root pages.

According to (Li et al. 2002), in the above two steps, usually the in-degree of a small-in-large-out link is as small as 0, 1, or 2, while the out-degree can be more than several hundred. Intuitively, in most cases, it is hard to believe that a root page with no or few in-links can point to many highly relevant documents. Even if it points to many good documents, due to the large number of documents in the base set, there may be some duplicates between the out-links of the small-in-large-out link and the neighbourhood of other pages, and these well duplicated documents still have the chance to top the hub set or the authority set. The method of setting in-link weights is very simple and can be further improved by adaptively changing the weights of both in- and out-links of a small-in-large-out link.

The experimental results obtained by (Li et al. 2002) show that the top 10 authorities and top 10 hubs from the above weighted algorithm combining with Bharat's weighted HITS are much better than those from pure Bharat's weighted HITS.

3.6 The Vector Space Model (VSM)

The vector space model has been widely used in traditional information retrieval. With this model, a space is created in which both pages and queries are represented by vectors. Particularly, for a collection of pages, an m-dimensional vector is generated for each page and each query from a set of terms with associated weights, where m is the number of unique terms in the page collection. Then, a vector similarity function, such as the inner product, can be used to compute the similarity between a page and a query.

Since vector space model refers to the contents of pages, it is natual to apply this values model to improve the HITS algorithm by adding semantic information into it. Actually, if r_i is the relevance/similarity score of a Web page i to the query that is calculated by vector space model, and h_i is the hub value of the page i, $r_i \cdot h_i$ instead of h_i is used to compute the authority of pages

it points to. Similarly, if a_i is its authority value, $r_i \cdot a_i$ instead of a_i is used to compute the hub values of pages that point to it.

In VSM, weights associated with the terms are calculated based on the following two numbers:

- term frequency, f_{ij}, the number of occurrence of term j in page i; and
- inverse document frequency, $g_j = \log(N/d_j)$, where N is the total number of pages in the collection and d_j is the number of pages containing term j.

The similarity $sim(q, i)$, between a query q and a page i, is defined as the cosine similarity of the query vector $Q=(q_1, q_2, \ldots, q_m)$ and the page vector $X_i = (x_{i1}, x_{i2}, \ldots, x_{im})$:

$$sim(q,i) = \frac{\sum_{j=1}^{m} q_j \cdot x_{ij}}{\sqrt{\sum_{j=1}^{m} q_j^2 \cdot \sum_{j=1}^{m} x_{ij}^2}}$$

where m is the number of unique terms in the page collection. Page weight x_{ij} and query weight q_j are calculated as follows:

$$x_{ij} = f_{ij} \cdot g_j = f_{ij} \cdot \log(N / d_j),$$

$$q_j = \begin{cases} \log(N / d_j) & \text{if term } j \text{ is in } q \\ 0 & \text{otherwise} \end{cases}.$$

Due to the dynamic nature of the Web and inability to access the whole Web, the VSM method cannot be applied directly because the inverse document frequency g_j is not available as N and d_j are often unknown for the pages on the Web. (Li et al. 2002) summarized three ways of dealing with this problem:

1. making simple assumptions, such as g_j is a constant for all the pages;
2. estimating parameter values by sampling the Web; and
3. using information from search engines, or/and experts' estimates.

In (Li et al. 2002), a combination of the second and third approaches was used. In that work, the Web is treated as a big database. To estimate N, the coverage of *Google* is used, which achieved 1,387,000,000 pages in August 2001. Thus, we can set N = 1,387,000,000. To estimate d_j, the number of pages containing a certain term j, (Li et al. 2002) used information extracted from several search engines as follows:

- Submit the term j to several search engines. Each search engine i returns α_i, the number of pages containing the term.
- Calculate the normalized value γ_i for each search engine i, based on α_i and the relative size β_i of the search engine to that of the whole Web: $\gamma_i = \alpha_i / \beta_i$.

As reported in 2002 (Li et al. 2002), the sizes of the five commonly used search engines, *AltaVista*, *Fast*, *Google*, *HotBot,* and *NorthernLight*, are 550, 625, 1000, 500, and 350 million pages, respectively. Since the total size of the Web is estimated to be about 1.387 billion pages, the relative sizes β_i of these search engines are 0.397, 0.451, 0.721, 0.360, and 0.252, respectively.

- Take the median of the normalized values of the search engines as the final result.

After getting the values of N and d_j, similarity/relevance score of a page to the query can be easily computed from the above formula.

Still using term vectors to determine the relevance scores between pages and query topic, the (Bharat and Henzinger 1998) used these relevance scores in different way to tackle the topic drift problem in the original HITS algorithm. In that method, the pages in the root set are used to define a broader query, instead of using the original query terms. Specifically, the first 1000 words from each page in the root set are concatenated to be a query (Bharat and Henzinger 1998). Then the pages are matched against this broader query based on their term vector similarities/relevance to the query. The similarity is defined in the same way as the above, but with a little bit difference. Instead of incorporating relevance scores into the HITS, (Bharat and Henzinger 1998) used these relevance scores to eliminate pages from the focused graph. All pages whose scores are below a threshold are pruned, then HITS or other HITS based algorithms are applied to this pruned graph. Thresholds are picked in one of three ways as follows (Bharat and Henzinger 1998):

1. Median Score: The threshold is the median of all the relevance scores.
2. Root Set Median Score: The threshold is the median of the relevance scores of the pages in the root set.
3. Fraction of Maximum Score: The threshold is a fixed fraction of the maximum score, such as max/10.

Readers who are interested in the details and experimental results of this method could refer to (Bharat and Henzinger 1998).

3.7 Cover Density Ranking (CDR)

The idea of using this method in improving HITS is the same as that of using VSM. However, instead of computing the relevance scores from term appearance, cover density ranking (CDR) is based on the appearance of phrases. CDR is more concerned with the relationship between the terms, rather than the terms individually. The idea is that a document containing most or all of the

query terms should be ranked higher than a document containing fewer terms, regardless of the frequency of term occurrence (Wilkinson et al. 1995). Before discussing CDR, we give the definition of cover first. A *cover* is an ordered pair (p_j, q_j) over a document, specifying the shortest interval of two distinct terms in the document (Clarke et al. 2000). p_j is the position of one term, q_j the position of another term, and q_j is assumed to be larger than p_j.

In CDR, the results of phrase queries are ranked in the following two steps (Clarke et al. 2000):

1. Documents containing one or more query terms are ranked by coordination level, i.e., a document with a larger number of distinct query terms ranks higher. The documents are thus sorted into groups according to the number of distinct query terms each contains, with the initial ranking given to each document based on the group in which it appears.
2. The documents at each coordination level are ranked to produce the overall ranking. The score of the cover set $\omega = \{(p_1, q_1), (p_2, q_2), \ldots, (p_n, q_n)\}$ is calculated as follows:

$$S(\omega) = \sum_{j=1}^{n} I(p_j, q_j) \text{ and}$$

$$I(p_j, q_j) = \begin{cases} \dfrac{\lambda}{q_j - p_j + 1} & \text{if } q_j - p_j + 1 > \lambda \\ 1 & Otherwise, \end{cases}$$

where λ is a constant such as 16 in (Li et al. 2002). Covers of length λ or shorter are given score 1, and longer covers are assigned scores less than 1 in proportional to the inverse of their lengths.

To adapt CDR to the Web, (Li et al. 2002) first finds out how many distinct query terms a page has and ranks the pages with more distinct terms higher. The method using CDR to compute the relevance scores of pages contains two steps:

1. Pages are scored according to the regular CDR method. Each page belongs to a coordination level group and has a score within that group.
2. The scores are normalized to range (0, 1] for pages containing only one term, to range (1, 2] for pages containing two different terms, and so on.

The benefit of this method is that it not only considers the number of distinct terms in a page, but also how these distinct terms appeared in the page, such as how close they are (Li et al. 2002).

3.8 In-depth Analysis of HITS

The original HITS and other improved HITS algorithms are all based on the topology information conveyed by hyperlinks. Then comes another problem: does an algorithm return similar results upon a small perturbation of the link structure or the page collection? This problem refers to the *stability* of a link analysis algorithm and should be paid attention to. This is because in the context of the Web, due to the dynamic feature of the Web, the Web may give us different views of the topology structure on different occasions. Ideally, a link analysis algorithm should be insensitive to hyperlink perturbations, i.e. if some pages are truly authoritative or influential, the addition or deletion of a few links should not make us change our minds about these pages having been very influential.

Actually, HITS uses matrix eigenvector calculations to assign "authority" or "hub" weights to pages. Therefore, the stability of HITS and other HITS based algorithms is turned out to be the eigenvector stability to the matrix perturbations. One of the examples regarding to this stability problem was given in (Ng 2001). This simple example showed how a small addition to a collection of Web pages could result in a large change to the eigenvectors returned. (Ng 2001) supposes there is a collection of Web pages that contains 100 Web pages linking *to http://www.algore.com*, and another 103 Web pages linking to *http://www.georgewbush.com*. The adjacency matrix A has all zeros except for the two columns corresponding to these two Web pages, therefore the principal eigenvector a^* will have non-zero values only *for algore.com* and *georgewbush.com*. Fig. 3.1 (Ng et al. 2001) (a) presents a jittered scatterplot of links to these two Web pages, along with the first two eigenvectors (only the non-zero portions of the eigenvectors are shown). Now, suppose five new Web pages trickle into the collection, which happen to link to both *algore.com* and *georgewbush.com*. Fig. 3.1(b) shows the new plot, and we see that the eigenvectors have changed dramatically, with the principal eigenvector now near the 45° line. Thus, a relatively *small* perturbation to the collection has caused a *large* change to the eigenvectors. (Ng et al. 2001) also indicated that a smaller number of pages that trickle into the collection also resulted in relatively large swings of the eigenvectors. For example, 1, 2, 3, 4 pages cause the principle eigenvectors to lie at 73°, 63°, 58° and 55° respectively. If this phenomenon is pervasive, then it needs to be addressed by any algorithm that uses eigenvectors to determine authority.

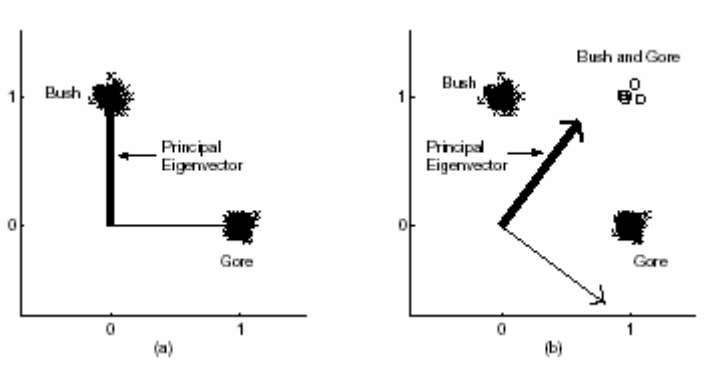


Fig. 3.1. Jittered scatterplot of hyperlink graph

HITS uses the principal eigenvector of matrix $S = A^T A$ to determine authorities. In (Ng 2001), it is shown that the stability of this eigenvector under small perturbations is determined by the *eigengap* of S, which is defined to be the difference between the largest and the second largest eigenvalues. The following example gives an intuitive view of the importance of the eigengap in determining eigenvector stability. Fig. 3.2 (Ng et al. 2001) plots the contours associated with two matrices S_1 and S_2 before (with solid lines) and after (with dashed lines) the same additive perturbation has been made to them. The eigenvalues of the matrices are indicated by the directions of the principal axes of the ellipses (Ng 2001).

Fig. 3.2. Contours of two matrices with different eigengaps

The matrix S_1 shown in Fig. 3.2(a) has eigengap $\delta_1 \approx 0$, and a small perturbation to S_1 (and hence the ellipse) results in eigenvectors away from the original eigenvectors; the matrix shown in Fig. 3.2(b) has eigengap $\delta_2 = 2$, and the perturbed eigenvectors are nearly the same as the original eigenvectors. It can be seen from this example that the size of the eigengap directly affects the stability of the eigenvectors.

In the following discussion, we use a tilde to denote perturbed quantities (e.g. $\tilde{S}$ denotes a perturbed version of S). Theorem 1 below reveals a fact that when the eigengap is large, the HITS is insensitive to small perturbations (Ng 2001).

Theorem 1. *Let $S = A^T A$ be given. Let a^* be the principal eigenvector and δ the eigengap of S. Assume the maximum out-degree of every Web page is bounded by d. For any $\varepsilon > 0$, suppose we perturb the Web/citation graph by adding or deleting at most k links from one page, where $k < (\sqrt{d+\alpha} - \sqrt{d})^2$, where $\alpha = \varepsilon\delta/(4+\sqrt{2}\varepsilon)$. Then the perturbed principal eigenvector $\tilde{a}^*$ of the perturbed matrix $\tilde{S}$ satisfies:*

$$\| a^* - \tilde{a}^* \|_2 \leq \varepsilon \tag{3.1}$$

This theorem also applies directly to hub weight calculations in which $S = AA^T$. This result is proved by showing i) the *direction* of the principal eigenvector does not change too much, and ii) the *magnitudes* of the relevant eigenvalues do not change too much, so the second eigenvector does not "overtake" the first and become the new principal eigenvector. (Ng 2001) gives the proof of this theorem as follows.

Proof. Let $\|\cdot\|_F$ denote the Frobenius norm. We apply Theorem V.2.8 from matrix perturbation theory (Stewart and JG 1990): Suppose $S \in R^{n\times n}$ is a symmetric matrix with principal eigenvalue λ^* and eigenvector a^*, and eigengap $\delta > 0$. Let E be a symmetric perturbation to S. Then the following inequalities hold for the old principal eigenpair (λ^*, a^*) and *some* new eigenpair $(\tilde{\lambda}, \tilde{a})$

$$\| a^* - \tilde{a} \|_2 \leq \frac{4 \| E \|_F}{\delta - \sqrt{2} \| E \|_F}, \text{ and} \tag{3.2}$$

$$| \lambda^* - \tilde{\lambda} | \leq \sqrt{2} \| E \|_F \tag{3.3}$$

assuming that the denominator in (2) is positive. Let the complementary eigenspace to (λ^*, a^*) be represented by (L_2, X_2), i.e. X_2 is orthonormal, and its columns contain all the eigenvectors of S except a^*; L_2 is diagonal and contains the corresponding eigenvalues, all of which are at least δ less than λ^*; and $SX_2 = X_2 L_2$. A bound similar to (3) holds for L_2:

$$\| L_2 - \tilde{L}_2 \|_F \leq \sqrt{2} \| E \|_F . \tag{3.4}$$

Let $\tilde{\lambda}_2$ be the largest eigenvalue of $\tilde{L}_2$. Using Corollary IV.3.6 from (Stewart and JG 1990), one can show that (4) implies

$$\tilde{\lambda}_2 \leq \lambda_2 + \sqrt{2} \parallel E \parallel_F . \tag{3.5}$$

If in turn $\sqrt{2} \parallel E \parallel_F < \delta / 2$, then inequalities (3) and (5) together will ensure that $\tilde{\lambda} > \tilde{\lambda}_2$, i.e. $(\tilde{\lambda}, \tilde{a})$ is the principal eigenpair of $\tilde{S}$.

Since we are adding or deleting links from only one page, let *F* denote the perturbation to one row of *A*, so that $\tilde{S} = (A+F)^T(A+F)$. It is straightforward to show $\|F^TF\| \leq k$ and $\|A^TF\|_F = \|F^TA\|_F \leq \sqrt{dk}$. We can thus bound the norm of the perturbation to *S*:

$$\parallel E \parallel_F = \parallel \tilde{S} - S \parallel_F \leq k + 2\sqrt{dk} . \tag{3.6}$$

Using inequalities (2) and (6) to determine when we may guarantee inequality (2) to hold, we arrive at the bound $k < (\sqrt{d+\alpha} - \sqrt{d})^2$, where $\alpha = \varepsilon\delta / (4 + \sqrt{2}\varepsilon)$. One can easily verify that the same bound on *k* also ensures $\sqrt{2} \parallel E \parallel_F < \delta / 2$, which also guarantees that the denominator in (2) is positive, hence $\tilde{a}^* = \tilde{a}$ as previously stated. □

(Ng et al. 2001) also gives another theorem below, which is the converse of the theorem 1, i.e. if the eigengap is small, then eigenvectors can be sensitive to perturbations.

Theorem 2. *Suppose S is a symmetric matrix with eigengap δ. Then there exists a O(δ) perturbation to S, i.e.* $\parallel S - \tilde{S} \parallel = O(\delta)$ *that causes a large* (Ω(1)) *change in the principal eigenvector.*

Proof. Since $S = S^T$, it can be diagonalized:

$$S = U \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \Sigma \end{pmatrix} U^T$$

where *U* is orthogonal, and whose columns are the *S*'s eigenvectors. Let u_i denote the *i*-th column of *U*. We pick $\tilde{S} = S + 2\delta u_2 u_2^T$. Since $\|u_2\|_2 = 1$, the norm of the perturbation is only $\|2\delta u_2 u_2^T\|_F = 2\delta$. Moreover,

$$\tilde{S} = U \begin{pmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 + 2\delta & 0 \\ 0 & 0 & \Sigma \end{pmatrix} U^T$$

As $\tilde{\lambda}_2 = \lambda_2 + 2\delta > \lambda_1$, $(\tilde{\lambda}_2, u_2)$ is the new principal eigenpair. But u_2 is orthogonal to u_1, so $\| u_2 - u_1 \|_2 = \Omega(1)$. □

To ground these results and illustrate why Theorem 1 requires a bound on out-degrees, (Ng et al. 2001) gave the following another example. This example showed a small perturbation—adding a single link—can have a large effect. In this example, a simple situation where two connected components in the focused graph was considered. Here a connected component of a graph is a subset whose elements are connected via link length ≥ 1 paths to each other, but not to the rest of the graph. The example used the fact that if a graph has multiple connected components, then the principal eigenvalue will have non-zero entries in nodes only from the "largest" connected component, i.e. the component with the largest eigenvalue.

Let us consider the Web/linkage graph shown in Fig. 3.3 (Ng et al. 2001), which is a small subset of a much larger graph. Solid arrows denote the original set of hyperlinks; the dashed arrow represents the link we will add. The original principal eigenvalue for each of the two connected components shown is $\lambda = 20$; with the addition of a single link, it is easy to verify that this jumps to $\tilde{\lambda} = 25$. Suppose that the community shown is part of a larger Web/linkage graph with multiple sub-communities, and that originally the biggest sub-community had eigenvalue $20 < \lambda_1 < 25$. It can be seen that by adding one link, the graph shown in Fig. 3.3 becomes the biggest sub-community, and the principal eigenvector now has positive values only for nodes shown in this figure, and zeros elsewhere. In this case, the algorithm is sensitive to a small perturbation to the topology structure.

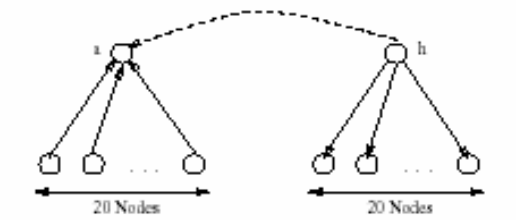


Fig. 3.3. Picture of a Web community

3.9 HITS Improvement

HITS is based on the adjacency matrix which considers only single directed hyperlinks. This, in some cases, will produce unreasonable results. (Wang 2002) gave a simple example revealing this problem.

Suppose there is a 4-node graph G as follow:

$$p_1 \leftarrow p_2 \rightarrow p_3 \rightarrow p_4,$$

i.e. there are three hyperlinks: from p_2 to p_1, from p_2 to p_3, and from p_3 to p_4. Applying the HITS algorithm to G, both the authority score for p_4 and the hub score for p_3 would converge to 0. This is an abnormal scoring, since the authority value of p_4 and the hub value of p_3 should be conferred by the hyperlink from p_3 to p_4.

Furthermore, if the node p_4 is removed from G, by applying HITS algorithm, p_1, p_2, and p_3 in the resulting graph:

$$p_1 \leftarrow p_2 \rightarrow p_3,$$

will be assigned exactly the same authority scores and hub scores as in G. In other words, HITS algorithm actually "nullified" the node p_4 in G in this case.

The basic reason why HITS algorithm presents this kind of problem is analysed by (Wang 2002) as follows: HITS is based on the authority-hub reinforcement through single directed hyperlinks only (by adjacency matrix) and has no consideration of following multiple consecutive hyperlinks (a normal behaviour when users navigate the Web), and thus no score updating is done through multiple consecutive hyperlinks.

To deal with this kind of problem, (Wang 2002) used another matrix instead of the adjacency matrix in updating stages of HITS algorithm. This new matrix takes consideration of multiple consecutive hyperlinks from any page p_i to another page p_j, rather than a single direct hyperlink. In this sense, all multiple consecutive hyperlinks from p_i to p_j on the hyperlink graph should be considered. The approach is trying to figure out the chance (probability) of following each directed path/hyperlink and making contribution to the score updating.

The details of this improvement are as follows (Wang 2002). Let $G = (V, E)$ be a connected hyperlink graph, with $V = \{p_1, p_2, \ldots, p_n\}$ where $n > 1$, and let P denote the *hyperlink probability matrix* of the hyperlink graph G; the (i, j)th entry of P is equal to the (positive) probability of following the directed hyperlink from p_i to p_j on page p_i if such hyperlink exists, and is equal to 0 otherwise. Since there is a (positive) probability of not following any directed hyperlink (Levene 2001), the sum of entries in each row of P is strictly less than *1*.

From the *hyperlink probability matrix* P, a new matrix H named as the *multiple-hyperlink probability* matrix can be generated such that the (i, j)th entry of H is the probability of following all multiple-hyperlinks (directed paths) from p_i to p_j, i.e.

$$H = \sum_{m=1}^{\infty} P^m = P(I - P)^{-1}.$$

Then for every node p_i in V, let $a[i]$ be its authority score and $h[i]$ its hub score. Initialize $a[i]$ and $h[i]$ to 1 for all nodes in V. Then, while the vectors a and h have not converged:

$$\{\text{For all nodes in V, } a[i] := \sum_{j=1}^{n} H_{ji} h[j];$$

$$\text{For all nodes in V, } h[i] := \sum_{j=1}^{n} H_{ij} a[j];$$

Normalize vectors a and h.

}

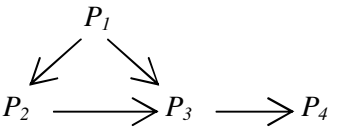
Like the HITS algorithm, this improved algorithm also converges to the principle eigenvectors of matrices H^TH and HH^T separately (Wang 2002).

Applying the above algorithm to some examples, (Wang 2002) presented some comparison results between the algorithm and the HITS, showing that the improved algorithm avoids the nullification abnormality mentioned at the beginning of this section. Two examples and the results are listed as follows:

Example 1: The previous demonstrated 4-node graph $p_1 \leftarrow p_2 \rightarrow p_3 \rightarrow p_4$.

	Authority Score	Hub Score
HITS	$(1/2, 0, 1/2, 0)^T$	$(0, 1, 0, 0)^T$
Improved Algo	$(1/4, 0, 1/4, 1/2)^T$	$(0, 1/2, 1/2, 0)^T$

Example 2: Another 4-node graph



	Authority Score	Hub Score
HITS	$(0, 0.3820, 0.6180, 0)^T$	$(0.6180, 0.3820, 0, 0)^T$
Improved Algo	$(0, 0.1706, 0.4737, 0.3558)^T$	$(0.4317, 0.3676, 0.2007, 0)^T$

The main issue of this improved algorithm is the method of constructing the hyperlink probability matrix P, i.e. how to compute the hyperlink probabilities. It is almost impossible to globally get the probability of each hyperlink on the Web. (Levene 2001) and (Borges 2000) proposed an approach to compute hyperlink probabilities of concerned Web pages from users' access log files. However, the method of computing the hyperlink probabilities within the focused graph, which is derived from a user's query rather than a user's access log file, still remains a problem. Although the overhead of this improved algorithm is the probability matrices, this algorithm presents a way

of improving the HITS algorithm as long as the hyperlink probability computation can be implemented effectively and efficiently.

3.10 Noise Page Elimination Algorithm Based on SVD

As analysed before, the topic drift problem of the HITS is mainly due to the tightly connection of many topic unrelated pages in the base set. In most cases, these topic unrelated pages, named *noise pages*, are brought into the base set by the pages in the root set (Bharat and Henzinger 1998) in the base set construction.

The algorithm in this section, which was proposed in (Hou and Zhang 2002), is to eliminate noise pages from the base set of pages B and obtain another good quality base set B', from which a better Web community with better authority and hub weights could be constructed. This algorithm purely makes use of the hyperlink information among the pages in B. Precisely, the algorithm considers the linkage relationships between the pages in root set R and pages in $B - R$. Here, $B - R$ is a page set and a page in it belongs to B but does not belong to R. These linkage relationships are expressed in a linkage (adjacency) matrix A. With the help of singular value decomposition of the matrix A (Datta 1995), the relationships among pages at a deeper level are revealed, and a numerical threshold is defined to eliminate noise pages. This approach is based on a reasonable assumption that the pages in the root set are topic related (Bharat and Henzinger 1998). Indeed, the root set R may also contain noise pages, though the possibility is small. However, by eliminating noise pages from the page set $B - R$, the influence of the remained noise pages in root set R to the HITS algorithm operation will be reduced, and better communities could be constructed. Therefore, the root set is used as a reference system in this algorithm to test if a page is a noise page.

For convenience, the root set of pages R is denoted as a directed graph $G(R)=(R,E_R)$: the nodes correspond to the pages, and a directed edge $(p,q)\in E_R$ indicates a link from p to q. Similarly, the base set of pages B is denoted as a directed graph $G(B)=(B,E_B)$. From the construction procedure of B, it can be easily inferred that $R \subset B$ and $E_R \subset E_B$.

Suppose the size of R (the number of pages in R) is n and the size of B is m. For the pages in R, a linkage (adjacency) matrix $S = (s_{ij})_{n\times n}$ is constructed as

$$s_{ij} = \begin{cases} 1 & when\ (i,j) \in E_R \ \ or \ \ (j,i) \in E_R \ \ or \ \ i = j \\ 0 & otherwise. \end{cases}$$

It represents the link relationships between the pages in R. For the pages in $B-R$, another linkage matrix $A=(a_{ij})_{(m-n)\times n}$ for page $i\in(B-R)$ and page $j\in R$ could also be constructed as

$$a_{ij}=\begin{cases}1 & when\ (i,j)\in E_B-E_R\ \ or\ \ (j,i)\in E_B-E_R\\ 0 & otherwise.\end{cases}$$

This matrix directly represents the linkage information between the pages in the root set and those not in the root set. The ith row of the matrix A, which is an n-dimensional vector, could be viewed as the coordinate vector of the page i in an n-dimensional space S_R spanned by the n pages in R.

For any two vectors $v1$ and $v2$ in an n-dimensional space S_n, as known in linear algebra, their similarity (or closeness) can be measured by their inner product (dot product) in S_n. The elements in $v1$ and $v2$ are the coordinates of $v1$ and $v2$ in the S_n respectively. In the page set $B-R$, since each page is represented as an n-dimensional vector (a row of matrix A) in the space S_R, all the similarities between any two pages in $B-R$ can be expressed as AA^T. On the other hand, for the matrix A, there exists a SVD:

$$A_{(m-n)\times n}=U_{(m-n)\times(m-n)}\Sigma_{(m-n)\times n}V^T_{n\times n}.$$

Therefore, the matrix AA^T can also be expressed as

$$AA^T=(U\Sigma)(U\Sigma)^T.$$

From this equation, it is obvious that matrix $U\Sigma$ is equivalent to the matrix A, and the ith ($i=1,\ldots,m-n$) row of matrix $U\Sigma$ could be naturally and reasonably viewed as the coordinate vector of the page i ($page\ i\in B-R$) in another n-dimensional space S'_R. Similarly, for the matrix S, there exists a SVD of S:

$$S_{n\times n}=W_{n\times n}\Omega_{n\times n}X^T_{n\times n}.$$

The ith ($i=1,\ldots,n$) row of matrix $W\Omega$ is viewed as the coordinate vector of the page i ($page\ i\in R$) in another n-dimensional space S''_R.

For the SVD of matrix A, the matrix U could be expressed as $U_{(m-n)\times(m-n)}=\left[u_1,u_2,\ldots,u_{m-n}\right]_{(m-n)\times(m-n)}$ where u_i ($i=1,\ldots,m-n$) is a $m-n$ dimensional vector $u_i=(u_{1,i},u_{2,i},\ldots,u_{m-n,i})^T$, and matrix V as $V_{n\times n}=\left[v_1,v_2,\ldots,v_n\right]_{n\times n}$ where v_i ($i=1,\ldots,n$) is an n dimensional vector $v_i=(v_{1,i},v_{2,i},\ldots,v_{n,i})^T$. Suppose $rank(A)=r$ and the singular values of matrix A are

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0.$$

For a given threshold δ ($0 < \delta \leq 1$), a parameter k is chosen such that

$$(\sigma_k - \sigma_{k+1}) / \sigma_k \geq \delta .$$

Let

$$U_k = [u_1, u_2, ..., u_k]_{(m-n)\times k},\ V_k = [v_1, v_2, ..., v_k]_{n\times k},$$

$$\Sigma_k = diag(\sigma_1, \sigma_2, ..., \sigma_k),$$

and

$$A_k = U_k \Sigma_k V_k^T .$$

As the theorem in Sect. 2.6 indicates, A_k is the best approximation to A with rank k. Accordingly, the ith row R_i of the matrix $U_k\Sigma_k$ is chosen as the coordinate vector of page i (*page* $i \in B - R$) in a k-dimensional subspace of S'_R:

$$R_i = (u_{i1}\sigma_1, u_{i2}\sigma_2, ..., u_{ik}\sigma_k), \quad i = 1, 2, \ldots, m - n. \tag{3.7}$$

Since matrix A contains linkage information between the pages in $B - R$ and R, from the properties of SVD and choice of parameter k, it can be inferred that coordinate vector (7) captures the main linkage information between the page i in $B - R$ and the pages in R. The extent to which main linkage information is captured depends on the value of parameter δ. The greater the value of δ, the more unimportant linkage information is captured. From the procedure of SVD (Datta 1995), coordinate vector transformation (7) refers to linkage information of every page in $B - R$, and whether a linkage in matrix A is dense or sparse is determined by all pages in $B - R$, rather than by a certain page. Therefore, equation (7) reflects mutual influence of all the pages in $B - R$ and reveals their relationships at a deeper level.

In a similar way, suppose *rank(S)=t* and the singular values of matrix S are

$$\omega_1 \geq \omega_2 \geq ... \geq \omega_t > \omega_{t+1} = ... = \omega_n = 0.$$

The ith row R'_i of the matrix $W_t\Omega_t$ is chosen as the coordinate vector of the page i (*page* $i \in R$) in a t-dimensional subspace of S''_R:

$$R'_i = (w_{i1}\omega_1, w_{i2}\omega_2, ..., w_{it}\omega_t), \quad i = 1, 2, \ldots, n. \tag{3.8}$$

Without loss of generality, let $k = min(k,t)$. The vector R_i can be expanded from a k-dimensional subspace to a t-dimensional subspace as

$$R_i = (u_{i1}\sigma_1, u_{i2}\sigma_2, ..., u_{ik}\sigma_k, \underbrace{0,0,...,0}_{t-k}), \quad i = 1, 2, \ldots, m-n. \tag{3.9}$$

To compare the closeness between a page in $B-R$ and the root set R, each page i in $B-R$ (i.e. vector R_i of (9)) is projected into the n-dimensional space spanned by the pages in R (i.e. vectors R'_i of (8), $i = 1,\ldots, n$). The projection of page i (*page* $i \in B-R$), PR_i, is defined as

$$PR_i = (PR_{i,1}, PR_{i,2}, ..., PR_{i,n}), \quad i = 1, 2, \ldots, m-n, \tag{3.10}$$

where

$$PR_{i,j} = (R_i, R'_j) / \| R'_j \| = \left(\sum_{k=1}^{t} R_{ik} \times R'_{jk}\right) \Big/ \left(\sum_{k=1}^{t} {R'_{jk}}^2\right)^{1/2}, \quad j = 1, 2, \ldots, n.$$

Within the same space, which is spanned by the pages in R, it is possible to compare the closeness between a page in $B-R$ and the root set R. In other words, a threshold for eliminating noise pages can be defined. In fact, for each PR_i, if

$$\| PR_i \| = \left(\sum_{j=1}^{n} PR_{i,j}^2\right)^{1/2} \geq c_{avg}, \tag{3.11}$$

where

$$c_{avg} = \sum_{j=1}^{n} \| R'_j \| \Big/ n,$$

then the page i in $B-R$ could remain in the base set of pages B. Otherwise, it should be eliminated from B. The parameter c_{avg} in the above equation represents the average link density of the root set R, and is the representative measurement of R. It is used as a threshold for eliminating noise pages. Intuitively, if a page in $B-R$ is a most likely noise page, it usually has fewer links with the pages in R. Thus its measurement $\|PR_i\|$ in (11) would be small and it is most likely to be eliminated. This noise page elimination algorithm (*NPEA*) algorithm is depicted as follows.

NPEA (*G(R)*, *G(B)*, δ)

Input:

***G(R)*:** *G(R)=(R,E_R)* is a directed graph of root set pages with nodes being pages and edges being links between pages.

***G(B)*:** *G(B)=(B,E_B)* is a directed graph of base set pages with node being pages and edges being links between pages.

δ**:** threshold for selecting matrix approximation parameter *k*.

Output:

***G'(B)*:** a new directed graph of base set pages with noise pages being eliminated by this algorithm.

Begin

Get the number of pages in *B*, *m* = size(*B*); Get the number of pages in *R*, *n* = size(*R*);

Construct linkage matrix between pages in *R*, $S = (s_{ij})_{n\times n}$;

Construct linkage matrix between $B - R$ and *R*, $A = (a_{ij})_{(m-n)\times n}$;

Compute the SVD of *S* and its singular values

$S_{n\times n} = W_{n\times n}\Omega_{n\times n}X^T_{n\times n}$; $\omega_1 \geq \omega_2 \geq ... \geq \omega_t > \omega_{t+1} = ... = \omega_n = 0$;

Compute the SVD of *A* and its singular values

$A_{(m-n)\times n} = U_{(m-n)\times(m-n)}\Sigma_{(m-n)\times n}V^T_{n\times n}$;

$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0$;

Choose parameter *k* such that $(\sigma_k - \sigma_{k+1})/\sigma_k \geq \delta$;

Compute coordinate vectors R_i (i = 1, 2, …, $m - n$) for each page in $B - R$ according to (7);

Compute coordinate vectors R'_i (i = 1, 2, …, n) for each page in *R* according to (8);

Compute the projection vectors PR_i (i = 1, 2, …, $m - n$) according to (10);

Compute the representative measurement of *R*, $c = \sum_{j=1}^{n} \| R'_j \| \Big/ n$;

if $\| PR_i \| < c$ (i = 1, 2, …, $m - n$) **then**

Begin

Eliminate page *i* from *B*, $B = B - page\ i$;

Eliminate links related with page *i* from E_B

$E_B = E_B - (page\ i \rightarrow p) - (p \rightarrow page\ i)$; $p \in B, p \neq i$.

End

return $G'(B) = (B, E_B)$;

End

The complexity of the algorithm is dominated by the SVD computation of the linkage matrices *A* and *S*. Without loss of generality, suppose $M = max(m$

– n, n). Then the complexity of the algorithm is $O(M^2 n + n^3)$ (Golub and Loan 1993). If $n << m$, the complexity is approximately $O(m^2 n)$.

3.11 SALSA (Stochastic algorithm)

As indicated in Sect. 3.1, the authorities of the base set S correspond to the principal eigenvector entries of the matrix $A^T A$, and the hubs correspond to the principal eigenvector entries of the matrix AA^T, where matrix A is the adjacency matrix of the base set S. From the definition of the matrix A, it is known that the value of the matrix entry is either 1 or 0, which means it is unified.

We define the matrix $\hat{A} = A^T A$ and matrix $\hat{H} = AA^T$, which are called association matrices. In the field of bibliometrics, the matrix $\hat{A}$ is called the *cocitation matrix* of the base set S, whose entry value $\hat{A}_{i,j}$ is the number of pages that jointly point at (cite) pages i and j. Matrix $\hat{H}$ is called the *bibliographic coupling matrix* of the base set S, whose entry value $\hat{H}_{i,j}$ is the number of pages jointly referred to (pointed at) by pages i and j. HITS algorithm provides a original approach of constructing these two association matrices with unified values. This construction approach does not consider the probability of Web surf from one page to another. Considering this situation, Lempel and Moran (Lempel 2001) proposed a *Stochastic Approach for Link-Structure Analysis (SALSA)*. The approach is based upon the theory of Markov chains, and relies on the stochastic properties of random walks performed on the collection of pages (base set). It differs from the HITS in the manner in which the association matrices are defined.

The input to SALSA consists of a base set of pages S which is built around a topic σ in the manner described in Sect. 3.1. Intuition suggests that authoritative pages on topic σ should be visible from many pages in the subgraph induced by S. Thus, a random walk on this subgraph will visit authorities with high probability. SALSA combines the theory of random walks with the notion of the two distinct types of Web pages, hubs and authorities, and actually analyses two different Markov chains: a chain of hubs and a chain of authorities (Lempel 2001). Analysing both chains allows the approach to give each Web page two distinct scores, a hub score and an authority score. The details of SALSA are presented as follows.

The SALSA begins with building a bipartite undirected graph $\hat{G} = (V_h, V_a, E)$ from base set and its link-structure:

– $V_h = \{s_h \mid s_h \in S$ and *out-degree* $(s_h) > 0\}$ (the *hub side* of $\hat{G}$)
– $V_a = \{s_a \mid s_a \in S$ and *in-degree* $(s_h) > 0\}$ (the *authority side* of $\hat{G}$).

– $E = \{(s_h, r_a) \mid s_h \in V_h, r_a \in V_a \text{ and } s_h \to r_a \text{ in } S\}$.

From the above bipartite construction, it is clear that each nonisolated page $s \in S$ is represented in $\hat{G}$ by one or both of the nodes s_h and s_a. Each link $s \to r$ is represented by an undirected edge connecting s_h and r_a. Fig. 3.4 (Lempel 2001) shows a construction of a bipartite graph from a given collection.

On this bipartite graph, two distinct random walks can be performed. These two walks will naturally start off from different sides of $\hat{G}$. Each walk will only visit nodes from one of the two sides of the graph, by traversing paths consisting of two $\hat{G}$-edges in each step. Since each edge crosses sides of $\hat{G}$, each walk is confined to just one of the graph's sides. Every path of length 2 in $\hat{G}$ represents a traversal of one link in the proper direction (when passing from the hub side of $\hat{G}$ to the authority side), and a retreat along a link (when passing from the authority side to the hub side). Since the hubs and authorities of the topic σ should be highly visible in $\hat{G}$ (reachable from many nodes by either a direct edge or by short paths), we may expect that the authorities will be among the nodes most frequently visited by the random walk on V_a, and that the hubs will be among the nodes most frequently visited by the random walk on V_h. In this way, two different Markov chains can be examined corresponding to these random walks: the chain of the visits to the authority side of $\hat{G}$, and the chain of visits to the hub side of $\hat{G}$. These chains distinguish two aspects of each page.

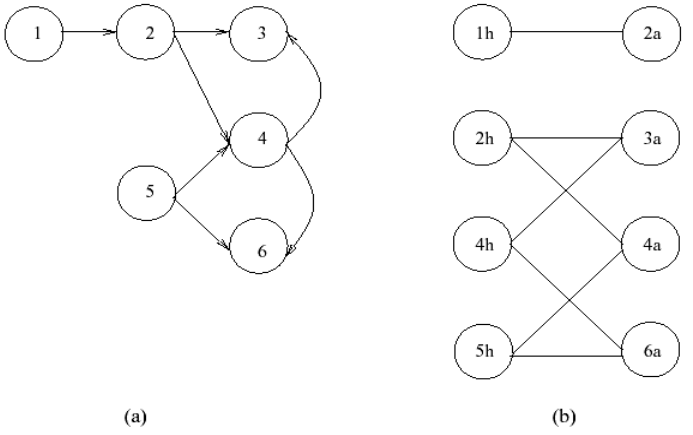


Fig. 3.4. Transforming (a) the collection S into (b) a bipartite graph $\hat{G}$

Now two stochastic matrices could be defined, which are the transition matrices of the two Markov chains at interest:

1. *The hub matrix* $\hat{H}$, defined as follows:

$$\hat{h}_{i,j} = \sum_{\{k|(i_h,k_a),(j_h,k_a)\in\hat{G}\}} \frac{1}{\deg(i_h)} \cdot \frac{1}{\deg(k_a)}$$

2. *The authority-matrix* $\hat{A}$, defined as follows:

$$\hat{a}_{i,j} = \sum_{\{k|(k_h,i_a),(k_h,j_a)\in\hat{G}\}} \frac{1}{\deg(i_a)} \cdot \frac{1}{\deg(k_h)},$$

where $deg(p)$ is the *in/out degree* of the page p, i.e. if page p is pointed to by other pages, $deg(p)$ is the *in* degree of p, otherwise is the *out* degree of p. A positive transition probability $\hat{a}_{i,j} > 0$ implies that a certain page k points to both pages i and j, and hence page j is reachable from page i by two steps: retracting along the link $k \to i$ and then following the link $k \to j$. The meaning of the probability $\hat{h}_{i,j} > 0$ is similar. From these two transition matrices, authorities and hub can be identified in the same way as HITS, i.e. the principal community of authorities (hubs) founded by SALSA will be composed of the k pages having the highest entries in the principal eigenvectors of $\hat{A}$ ($\hat{H}$) for some user defined k.

Alternatively, the matrices $\hat{H}$ and $\hat{A}$ can be defined as follows. Let A be the adjacency matrix of the directed graph defined by its link topology (see Sect. 3.1). Denoted by A_r the matrix whose each non-zero entry value is divided by the sum of the entry values in its row, and by A_c the matrix whose each non-zero entry value is divided by the sum of the entry values in its column. From the adjacency matrix construction, it is obvious that the sums of rows/columns that contain nonzero entry are greater than zero. Then $\hat{H}$ consists of the non-zero rows and columns of $A_r A_c^T$, and $\hat{A}$ consists of the non-zero rows and columns of $A_c^T A_r$. Recall that the HITS uses the association matrices AA^T, A^TA instead. We also assume that $\hat{G}$ is connected, causing both stochastic matrices $\hat{A}$ and $\hat{H}$ to be *irreducible*. This assumption does not form a limiting factor, since when $\hat{G}$ is not connected the above technique could be used on each connected component separately.

Lempel and Moran (Lempel 2001) compared SALSA and HITS by applying these two algorithms to some artificial and real situations. The comparison focused on the tightly knit community (TKC) effect, which usually caused topic drift problem in HITS. As defined in (Lempel 2001), a tightly knit community is a small but highly interconnected set of pages. Roughly, the *TKC effect* occurs when such a community scores high in link-analysing algorithms, even though the pages in the TKC are not authoritative on the topic, or pertain to just one aspect of the topic. The comparison in (Lempel 2001) indicates that HITS is vulnerable to this effect, and will sometimes rank the pages of a TKC in unjustifiably high positions, while SALSA is less vulnerable to the TKC effect and produces good results in cases where the HITS fails to do so.

It seems to be a premise for linkage analysis that considering the in-degree of pages as a sole measure of their authorities does not produce satisfactory results. Some examples in (Kleinberg 1998) endorsed this premise. However, the results produced by SALSA seem to contradict this premise: the stochastic rankings seem quite satisfactory there, since the stochastic rankings are equivalent to simple in-degree counts (normalized by the size of the connected component which each page belongs to). The reason why SALSA is successful, as indicated in(Lempel 2001), is the improvement of the assembled sub-graph quality. Actually, for SALSA, when expanding the root set to the base set, an attempt is made to filter non-informative links which exist between Web pages. This was done by studying the target URL of each link, in conjunction with the URL of the link's source(Lempel 2001).

- Following Kleinberg (Kleinberg 1998), intra-domain links are ignored, since these links tend to be navigational aids inside an intranet, and do not confer authority on the link's destination. The heuristic in SALSA did not rely solely on an exact match between the hosts of the link's source and target, and was also able to classify links between related hosts (such as "*shopping.yahoo.com*" and "*www.yahoo.com*") as being intra-domain.
- Links to *cgi scripts* are ignored. These links are usually easily identified by the path of the target URL (e.g., *http://www.altavista.com/cgi-bin/query?q=car*).
- Ad-links are tried to be identified and ignored. This was achieved by deleting links that contained certain characters in their URL (such as '=', '?' and others) which appear almost exclusively in advertisements and sponsorship links, and in links to dynamic content.

By this improvement of page set, as well as corresponding sub-graph, 38% of the links examined in (Lempel 2001) were usually ignored. The page sets/collections themselves turn out to be relatively sparse graphs, with the number of edges never exceeding three times the number of nodes.

As a final note, denoting the set of outgoing links of a page p by $L(p)$, Lempel and Moran (Lempel 2001) observed that both SALSA and HITS approaches obey the following property:

$$L(p) \subseteq L(q) \Rightarrow \text{hub-score}\ (p) \leq \text{hub-score}\ (q).$$

Thus, in both approaches, adding outgoing links to your page can only improve its hub score. In order to fight this sort of spam, link analysis must punish pages for having an excess of irrelevant links.

In general, HITS algorithm paves a way of conducting hyperlink analysis although there are still many issues to be solved about this algorithm. Many efforts have been made to improve this algorithm. These efforts can be mainly classified into two categories: the first one is to more precisely describe rela-

tionships between pages conveyed by hyperlinks, another one is to improve the quality of the focused page set as well as the corresponding sub-graphs. As stated in (Bharat and Henzinger 1998; Chakrabarti et al. 1998) although hyperlink analysis could reveal many latent relationship between Web pages, it is limited in revealing sematic information between pages. Therefore, further research can be done in combining Web page content analysis with hyperlink analysis.

4 PageRank Related Algorithms

Web page ranking is another interesting and challenging topic in Web community research with a great application potential for improving Web information searches. This chapter presents and discusses some representative algorithms in this area. Sect. 4.1 gives the original PageRank algorithm that is based on hyperlink information. The algorithm that probabilistically combines hyperlink and content information for page ranking is discussed in Sect. 4.2. As an improvement of previous two algorithms, a page ranking algorithm that considers user's query context or topics is discussed in Sect. 4.3. Sect. 4.4 and 4.5 are dedicated respectively to two algorithms that aim at accelerating PageRank computation. In Sect. 4.6, a general framework that is based on random walk model for page ranking is discussed. This random walk model considers all possible actions the Web surfer may take. Sect. 4.7 discusses another page ranking algorithm that mainly focuses on authority page ranking. The last two sections, Sect. 4.8 and 4.9, are dedicated to another kind of page ranking algorithms that rank pages with respect to user's queries or a given page.

4.1 The Original PageRank Algorithm

It is clear from Chap. 3 that Web search results can be much improved by using the information contained in link structure between pages. For example, given a query which is a set of words or other query terms, HITS invokes a traditional search engine to obtain a set of pages relevant to it, expands this set with its inlinks and outlinks, and then attempts to find two types of pages, *hubs* and *authorities*. Because this computation is carried out at query time, it is not feasible for today's search engines, which need to handle tens of millions of queries per day.

The PageRank algorithm was originally proposed by Brin and Page (Brin and L. Page 1998) and incorporated in the search engine *Google*. In contrast to HITS, PageRank computes a single measure of quality for a page at crawl time. This measure is then combined with a traditional information retrieval (IR) score at query time, and has the advantage of much greater efficiency.

Actually, PageRank also tries to capture the notion of "importance" of a page from linkage information. For instance, from our intuition, the *Yahoo!* homepage is more important than the homepage of the Stanford Database Group. This difference is reflected in the number of other pages that point to these two pages, that is, more pages point to the *Yahoo!* homepage than to the Stanford Database Group homepage. The rank of page *A* could thus be defined as the number of pages in the Web that point to *A*, and could be used to rank the results of a search query (Arasu 2000). This is actually the idea of academic literature citation that is applied to the Web for giving some approximation of a page's importance or quality. However, citation ranking does not work very well, especially against spamming, since it is quite easy to artificially create a lot of pages to point to a desired page.

PageRank extends the basic citation idea by taking into consideration the importance of the pages that point to a given page. It does this by examining page importance propagation through hyperlinks, i.e. the importance of a page both *depends on* and *influences* the importance of other pages. Thus a page receives more importance if *Yahoo* points to it than if some unknown page points to it. Citation ranking, in contrast, does not distinguish between these two cases.

Let's first discuss a simple definition of PageRank that captures the above intuition (Arasu 2000). Let the pages on the Web be denoted by 1, 2, … , m. Let $N(i)$ denote the number of forward (outgoing) links from page i. Let $B(i)$ denote the set of pages that point to page i. For now, assume that the Web pages form a strongly connected graph, i.e. every page can be reached from any other page. The simple PageRank of page i, denoted by $r(i)$, is given by

$$r(i) = \sum_{j \in B(i)} r(j) / N(j) .$$

The division by $N(j)$ captures the intuition that pages which point to page i evenly distribute their ranks to all of the pages they point to. This simple rank algorithm can be expressed in the linear algebra language. Actually it can be written as $r = A^T r$, where r is the $m \times 1$ vector $[r(1), r(2), \ldots, r(m)]^T$, and the elements $a_{i,j}$ of the matrix A are $a_{i,j} = 1/N(i)$, if page i points to page j, and $a_{i,j} = 0$ otherwise. Thus the PageRank vector r is the eigenvector of matrix A^T corresponding to the eigenvalue 1. Since the graph is strongly connected, it can be shown that 1 is an eigenvalue of A^T, and the eigenvector r is uniquely defined when a suitable normalization is performed and the vector is non-negative (Arasu 2000).

Since computing simple PageRank is equivalent to computing the principal eigenvector of the matrix A^T defined above, many methods of computing a matrix principal eigenvector can be used for page rank computation. One of the simplest methods is called *power iteration (Golub and Loan 1993)*. In the

power iteration, an arbitrary initial vector is repeatedly multiplied with the given matrix until it converges to the principal eigenvector. To guarantee the convergence of the power iteration, it is required that the graph is aperiodic. In practice, the Web is always aperiodic (Arasu 2000). The power iteration for PageRank computation is given below.

1. $s \leftarrow$ any random vector
2. $r \leftarrow A^T \times s$
3. if $\| r - s \| < \varepsilon$ then end and r is the PageRank vector, otherwise go to step 4
4. $s \leftarrow r$, go to 2.

To illustrate, Fig. 4.1(a) shows the PageRank for a simple graph (Arasu 2000). It is easy to verify that this assignment of ranks satisfies the definition of PageRank. For instance, node 2 has a rank of 0.286 and two outgoing links. Half of its rank (0.143) flows to node 1 and half to node 3. Since node 3 has no other incoming links, its rank is what is received from node 2, i.e., 0.143. Node 1 receives 0.143 from 2, plus 0.143/2 from node 3, plus 0.143/2 from node 5, for a total of 0.286. Note that node 1 has a high rank because it has three incoming links. Node 2 has the same high rank because anyone who visits 1 will also visit 2. Also note that the ranks over all nodes add up to 1.0.

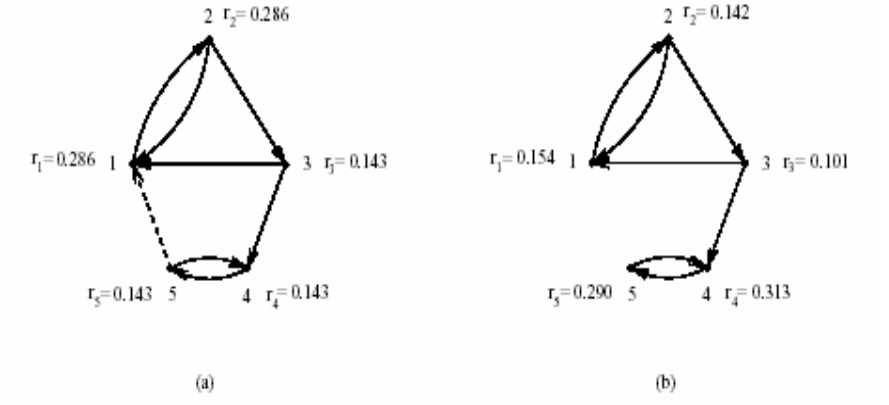


Fig. 4.1. (a) Simple PageRank (b) Modified PageRank with $d = 0.8$

Simple PageRank is well defined only if the link graph is strongly connected. However, the Web is far from strongly connected. In *Google*, Brin and Page (Brin and L. Page 1998) used a modified PageRank which is defined as follows:

$$r(i) = (1-d)/m + d * \sum_{j \in B(i)} r(j)/N(j),$$

where m is the total number of nodes in the graph, d ($0 < d < 1$) is a damping factor which is set to 0.85 in (Brin and L. Page 1998). It is clear that the simple PageRank is a special case that occurs when $d = 1$. Note that the PageRanks form a probability distribution over Web pages, so the sum of all Web pages' PageRank will be 1. Fig. 4.1 (b) shows the modified PageRank (for $d = 0.8$) for the graph of Fig. 4.1(a) with the link $5 \rightarrow 1$ removed. Nodes 4 and 5 now have higher ranks than the other nodes, indicating that surfers will tend to gravitate to 4 and 5. However, the other nodes have non-zero ranks.

PageRank can be thought of as a model of user behaviour (Brin and L. Page 1998). Suppose there is a "random surfer" who is given a Web page at random and keeps clicking on links, never hitting "back" but eventually gets bored and starts on another random page. The probability that the random surfer visits a page is its PageRank, and the damping factor d is the probability at each page the "random surfer" will get bored and request another random page.

Another intuitive justification (Brin and L. Page 1998) is that a page can have a high PageRank if there are many pages that point to it, or if there are some pages that point to it and have a high PageRank. Intuitively, pages that are well cited from many places around the Web are worth looking at. Also, pages that have perhaps only one citation from something like the *Yahoo!* homepage are also generally worth looking at. If a page was not high quality, or was a broken link, it is quite likely that *Yahoo*'s homepage would not link to it. PageRank handles both these cases by recursively propagating weights through the link structure of the Web.

The above power iteration for simple PageRank computation can also be adapted to compute this modified PageRank. In this case, the matrix A in the iteration should be defined as

$$A = (1-d)\left[\frac{1}{m}\right]_{m \times m} + dM, \text{ with } M_{i,j} = \begin{cases} \frac{1}{N(i)} & \text{if page } i \text{ points to page } j \\ 0 & \text{otherwise,} \end{cases}$$

Since the rows of A are normalized, the PageRank corresponds to the principal eigenvectors of A with the eigenvalue of 1.

Theoretically, the convergence of the power iteration for a matrix depends on the *eigenvalue gap*, which is the difference between the modulus of the two largest eigenvalues of the given matrix. Page and Brin et al (Page 1998) claim that the power iteration converges reasonably fast, in around 50 iterations. In practice, we are more interested in the relative ordering of the pages induced by the PageRank, than the actual PageRank values themselves. Thus

we can terminate the power iteration once the ordering of the pages becomes reasonably stable. This will save the overhead of iteration operations.

In *Google*, PageRank is used to prioritize the results of Web keyword searches. Particularly, *Google* uses both IR techniques and PageRank to answer keyword queries. Given a query, *Google* computes an IR score for all the pages that contain the query terms. The IR score is combined with the PageRank of these pages to determine the final rank for this query.

4.2 Probabilistic Combination of Link and Content Information

PageRank can greatly improve the traditional keyword based Web search. However, the disadvantage of it is that the PageRank score of a page ignores whether or not the page is relevant to the query at hand. In other words, the PageRank score of a page can be viewed as the rate at which a surfer would visit that page, if it surfed the Web indefinitely, blindly jumping from page to page (Richardson 2002). The probabilistic combination of link and content information in PageRank, which was proposed by Richardson and Domingos, does something closer to what a human surfer would, jumping preferentially to pages containing the query terms.

As indicated in (Richardson 2002), a problem common to both PageRank and HITS is topic drift. Because they give the same weight to all edges, the pages with the most inlinks in the network being considered (either at crawl or query time) tend to dominate, whether or not they are the most relevant to the query. The algorithm in this section, in contrast, is query-dependent, content sensitive version of PageRank.

Original PageRank simulates a Web surfer's behaviour, jumping from page to page with uniform probability of choosing link to follow at each step. Suppose now we have a more intelligent surfer, who probabilistically hops from page to page, depending on the content of the pages and the query terms the surfer is looking for. Then, the resulting probability distribution over pages is (Richardson 2002):

$$P_q(j) = (1-d)P_q'(j) + d\sum_{i \in B(j)} P_q(i)P_q(i \rightarrow j) \tag{4.1}$$

where $P_q(i \rightarrow j)$ is the probability that the surfer transitions to page j given that he is on page i and is searching for the query q. $P_q'(j)$ specifies where the surfer chooses to jump when not following links. $P_q(j)$ is the resulting probability distribution over pages and corresponds to the *query-dependent Pag-*

eRank score, i.e. QD-PageRank$_q$(j) ≡ $P_q(j)$. Compared with original Page-Rank, probability distribution (1) is different in replacing 1/*m* with $P_q'(j)$ and replacing 1/*N*(*i*) with $P_q(i \to j)$. This means, in this algorithm, the hyperlinks are not evenly considered anymore. Choosing the hyperlink to go along depends on if the link is related to the query *q*.

As with PageRank, QD-PageRank is determined by iterative evaluation of equation 1 from some initial distribution, and is equivalent to the primary eigenvector of the transition matrix $\mathbf{Z}_q$, where

$$\mathbf{Z}_q(j,i) = (1-d)P_q'(j) + d\sum_{i \in B(j)} P_q(i \to j).$$

$P_q(i \to j)$ and $P_q'(j)$ are determined as follows:

$$P_q'(j) = \frac{R_q(j)}{\sum_{k \in W} R_q(k)}, \qquad P_q(i \to j) = \frac{R_q(j)}{\sum_{k \in F_i} R_q(k)}, \tag{4.2}$$

where $R_q(j)$ is a measure of *relevance* of page *j* to query *q*, *W* is the set of pages and F_i is the set of pages page *i* links to. This equation means when choosing among multiple out-links from a page, the directed surfer tends to follow those which lead to pages whose content has been deemed relevant to the query. Similar to PageRank, where a page's outlinks all have zero relevance, or has no outlinks, links are added from that page to all other pages in the network. On such a page, the surfer thus chooses a new page to jump to according to the distribution $P_q'(j)$ (Richardson 2002).

There are many ways to determine the relevance function $R_q(j)$. In the simplest case, $R_q(j)$ is chosen as a constant, i.e. $R_q(j) = R$. This relevance function is independent of the query term and the document, and QD-PageRank then reduces to Page-Rank. One simple content-dependent function could be $R_q(j)$ = 1 if the term *q* appears on page *j*, and 0 otherwise. Much more complex functions could be used, such as the well-known TFIDF information retrieval metric (Salton and MJ 1983), a score obtained by latent semantic indexing (Deerwester et al. 1990), or any heuristic measure using text size, positioning, etc. It is important to note that most current text ranking functions could be easily incorporated into the above algorithm.

In practice, the user's query usually contains several query terms. For a given multiple-term query, $Q=\{q_1,q_2,\ldots\}$, the surfer surfs the Web like this: the surfer selects a query term *q* according to some probability distribution *P(q),* and uses that term to guide his behaviour according to equation 1 for a large number of steps. He then selects another term according to the distribution to determine his behaviour, and so on. The resulting distribution over visited Web pages is QD-PageRank$_Q$ and is given by (Richardson 2002):

$$QD - PageRank_Q(j) \equiv P_Q(j) = \sum_{q \in Q} P(q) P_q(j) \tag{4.3}$$

As indicated before, for standard PageRank, the PageRank vector is equivalent to the primary eigenvector of the matrix A^T, The vector of single-term QD-PageRank$_q$ is again equivalent to the primary eigenvector of the matrix $\mathbf{Z}_q$. From these facts, can we say QD-PageRank$_Q$ vector is equivalent to the primary eigenvector of a matrix $\mathbf{Z}_Q = \sum_{q \in Q} P(q)\mathbf{Z}_q$? In fact, this is not the case. As studied by (Richardson 2002), the primary eigenvector of $\mathbf{Z}_Q$ corresponds to the QD-PageRank obtained by a random surfer who, *at each step*, selects a new query according to the distribution *P(q)*. However, QD-PageRank$_Q$ is approximately equal to the PageRank that results from this single-step surfer. (Richardson 2002)proved this conclusion as follows.

Let $\mathbf{x}_q$ be the L_2-normalized primary eigenvector for matrix $\mathbf{Z}_q$ (note element *j* of $\mathbf{x}_q$ is QD-PageRank$_q$(*j*)). Since $\mathbf{x}_q$ is the primary eigenvector for $\mathbf{Z}_q$, we have (Golub and Loan 1993) $\forall\ q, r \in Q$, $\| \mathbf{Z}_q\mathbf{x}_q \| \geq \| \mathbf{Z}_q\mathbf{x}_r \|$. Thus, to a first degree of approximation, $\mathbf{Z}_q \sum_{r \in Q} \mathbf{x}_r \approx k\mathbf{Z}_q\mathbf{x}_q$. Suppose $P(q) = 1/|Q|$. Consider $\mathbf{x}_Q = \sum_{q \in Q} P(q)\mathbf{x}_q$, then we have

$$\mathbf{Z}_Q\mathbf{x}_Q = \left(\sum_{q \in Q} \frac{1}{|Q|} \mathbf{Z}_q \right) \left(\sum_{q \in Q} \frac{1}{|Q|} \mathbf{x}_q \right) = \frac{1}{|Q|} \sum_{q \in Q} \left(\mathbf{Z}_q \sum_{r \in Q} \frac{1}{|Q|} \mathbf{x}_r \right)$$

$$\approx \frac{1}{|Q|} \sum_{q \in Q} \frac{k}{|Q|} \mathbf{Z}_q\mathbf{x}_q = \frac{k}{|Q|} \mathbf{x}_q ,$$

and thus $\mathbf{x}_Q$ is approximately an eigenvector for $\mathbf{Z}_Q$. Since $\mathbf{x}_Q$ is equivalent to QD-PageRank$_Q$, and $\mathbf{Z}_Q$ describes the behaviour of the single-step surfer, QD-PageRank$_Q$ is approximately the same PageRank that would be obtained by using the single-step surfer. The approximation has the least error when the individual random surfers defined by $\mathbf{Z}_q$ are very similar, or are very dissimilar.

In practical use, the difficulty with calculating a query dependent PageRank is that a search engine cannot perform the computation, which can take hours, at query time, when it is expected to return results in seconds or less. This problem could be surmounted by pre-computing the individual term rankings QD-PageRank$_q$, and combining them at query time according to equation 3 (Richardson 2002). (Richardson 2002)claimed that the computation and storage requirements for QD-PageRank$_q$ for hundreds of thousands

of words are only approximately 100 times those of a single query independent PageRank.

4.3 Topic-Sensitve PageRank

Although Richardson and Domingos' PageRank (Richardson 2002) enhances search ranking by generating a PageRank vector for each possible query term, the approach requires considerable processing time and storage, and is not easy to be extended to make use of user and query context. To enable the PageRank be able to consider user's query context or topic, Havrliwala (Haveliwala 2002) proposed a topic-sensitive PageRank. This algorithm is based on 16 topics that are extracted from the top-level categories of the Open Directory Project (ODP) . In this approach, we pre-compute a set of importance scores of a page with respect to various topics. At query time, these importance scores are combined based on the topics of the query to form a composite PageRank score for those pages matching the query.

Actually, the topic-sensitive PageRank comes from the original PageRank algorithm. As indicated the Sect. 4.1, the original PageRank scores of pages corresponds to the principal eigenvector entries of the matrix A^T. We can rewrite this algorithm in the following form:

$$\vec{r} = A^T \times \vec{r} = (1 - \alpha) M^T \times \vec{r} + \alpha \vec{p},$$

where $\vec{r}$ is the page score vector, i.e. $\vec{r} = [r(i)]_{m\times 1}$, $\alpha = (1 - d)$ and $\vec{p} = [1/m]_{m\times 1}$. In (Haveliwala 2002), the value of α is set to 0.25. The key of creating topic-sensitive PageRank is to bias the computation to increase the effect of certain categories of pages by using a non-uniform $m\times 1$ personalization vector for $\vec{p}$ (Haveliwala 2002).

This topic-sensitive PageRank consists of two steps: ODP-biasing and query-time importance score computing. The details of these two steps are as presented below (Haveliwala 2002).

ODP-biasing This step is to generate a set of biased PageRank vectors from a set of basis topics. This step is performed once, offline during the pre-processing of the Web crawl. In this step, 16 different biased PageRank vectors are created by using the URLs below each of the 16 top-level categories of the ODP as the personalization vector. In particular, let T_j be the set of URLs under the ODP category c_j. Then when computing the PageRank vector for topic c_j, we use the a non-uniform vector $\mathbf{V}_j$ in place of the uniform damping vector $\vec{p}$, i.e. $\vec{p} = \vec{v}_j$ where

$$v_{ji} = \begin{cases} \frac{1}{|T_j|} & i \in T_j \\ 0 & i \notin T_j \end{cases}$$

The PageRank vector for topic c_j is denoted as $\overrightarrow{PR}(\alpha, \vec{v}_j)$. In this step, we also compute the 16 class term-vector $\vec{D}_j$ consisting of the terms in the documents below each of the 16 top-level categories. Actually, D_{jt} simply gives the total number of occurrences of term t in documents listed below category c_j of ODP.

It is obvious that we can also use other sources for creating topic-sensitive PageRank vectors. The reason why (Haveliwala 2002) uses ODP is that the ODP data is freely available. Furthermore, ODP is complied by thousands of volunteer editors, therefore, it is less susceptible to influence by any one party.

Query-Time Importance Score Computing This step is performed as query time. Given a query q, let q' be the context of q. In other words, if the query was issued by highlighting the term q in a page u, then q' consists of the terms in u. For ordinary queries not done in context, let $q' = q$. Firstly, we compute the class probabilities for each of the 16 top-level ODP classes, conditioned on q'. Let q'_i be the ith term in q'. Then given the query q, we computing the following for each c_j:

$$P(c_j \mid q') = \frac{P(c_j) \cdot P(q' \mid c_j)}{P(q')} \propto P(c_j) \cdot \prod_i P(q'_i \mid c_j).$$

$P(q'_i \mid c_j)$ is easily computed from the class term-vector $\vec{D}_j$. The quantity $P(c_j)$ is not so straightforward. In (Haveliwala 2002), $P(c_j)$ is made uniform. In other words, for some user k, we can use a prior distribution $P_k(c_j)$ that reflects the interests of user k.

Next, we compute the query-sensitive importance score of each retrieved page that contains the original query term q. Let $rank_{jd}$ be the rank of page d given by the rank vector $\overrightarrow{PR}(\alpha, \vec{v}_j)$, then the query-sensitive importance score s_{qd} for the page d is computed as follows:

$$s_{qd} = \sum_j P(c_j \mid q') \cdot rank_{jd}.$$

The results are ranked according to this composite score s_{qd}.

The above query-sensitive PageRank computation has the following probabilistic interpretation. With probability $1 - \alpha$, a random surfer on page u follows an outlink of u where the particular outlink is chosen uniformly at random. With probability $\alpha P(c_j|q')$, the surfer instead jumps to one of the pages

in T_j where the particular page in T_j is chosen uniformly at random. The long term visit probability that the surfer is at page d is exactly given by the composite score s_{qd} defined above.

4.4 Quadratic Extrapolation

As discussed before, the PageRank scores, which correspond to the principal eigenvector entries of the matrix A^T, are computed by using power iteration method. For simplicity of the following discussion, we denote A^T as A. The power method is the oldest method for computing the principal eigenvector of a matrix. The intuition behind the convergence of the power method is as follows. We assume, for simplicity, that the start vector $\vec{x}^{(0)}$ lies in the subspace spanned by the eigenvectors of A (this assumption does not affect the convergence of the method). Then $\vec{x}^{(0)}$ can be written as a linear combination of the eigenvectors of A:

$$\vec{x}^{(0)} = \vec{u}_1 + \alpha_2 \vec{u}_2 + \ldots + \alpha_m \vec{u}_m ,$$

where $\vec{u}_i$ ($i = 1, \ldots, m$) are eigenvectors of A. Since we know that the first eigenvalue of matrix A is $\lambda_1 = 1$, we have

$$\vec{x}^{(1)} = A\vec{x}^{(0)} = \vec{u}_1 + \alpha_2 \lambda_2 \vec{u}_2 + \ldots + \alpha_m \lambda_m \vec{u}_m ,$$

and

$$\vec{x}^{(n)} = A^n \vec{x}^{(0)} = \vec{u}_1 + \alpha_2 \lambda_2^n \vec{u}_2 + \ldots + \alpha_m \lambda_m^n \vec{u}_m .$$

Since $\lambda_m \leq ,\ldots, \leq \lambda_2 < 1$, $A^{(n)}\vec{x}^{(0)}$ approaches $\vec{u}_1$ as n grows large. Therefore, the power method converges to the principal eigenvector of the matrix A. It can be seen as well that if the eigengap $1 - |\lambda_2|$ is small, the convergence of the method will be slow because n must be large before λ_2^n is close to 0. Because of this, it is necessary to use faster method for computing PageRank, especially for those personalized and topic-sensitive PageRank schemes that require computing many PageRank vectors, each biased towards certain types of pages.

For this purpose, (Kamvar et al. 2003) proposed a quadratic extrapolation algorithm for the fast computation of PageRank. The algorithm is named *QuadraticPowerMethod* in which another algorithm, named *QuadraticExtrapolation*, is embedded. These two algorithms are depicted as follows (Kamvar et al. 2003):

Function $\vec{x}^{(n)}$ = *QuadraticPowerMethod* ( ) {
$\vec{x}^{(0)} = \vec{v}$;

$k = 1$;
repeat
$\vec{x}^{(k)} = A\vec{x}^{(k-1)}$;
$\delta = \| \vec{x}^{(k)} - \vec{x}^{(k-1)} \|_1$;
periodically,
$\vec{x}^{(k)} = QuadraticExtrapolation(\vec{x}^{(k-3)},...,\vec{x}^{(k)})$;
$k = k+1$;
until $\delta < \varepsilon$;
}

Function $\vec{x}^{*} = QuadraticExtrapolation(\vec{x}^{(k-3)},...,\vec{x}^{(k)})$ {
for j = $k - 2 : k$ **do**
$\vec{y}^{(j)} = \vec{x}^{(j)} - \vec{x}^{(k-3)}$;
end
$Y = (\vec{y}^{(k-2)}, \vec{y}^{(k-1)})$;
$\gamma_3 = 1$;
$\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = -Y^{+}\vec{y}^{(k)}$;
$\beta_0 = \gamma_1 + \gamma_2 + \gamma_3$;
$\beta_1 = \gamma_2 + \gamma_3$;
$\beta_2 = \gamma_3$;
$\vec{x}^{*} = \beta_0\vec{x}^{(k-2)} + \beta_1\vec{x}^{(k-1)} + \beta_2\vec{x}^{(k)}$;
}

The overhead in performing the extrapolation, function *QuadraticExtrapolation*, comes primarily from the least-squares computation of γ_1 and γ_2, i.e.

$$\begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} = -Y^{+}\vec{y}^{(k)},$$

where Y^{+} is the pseudoinverse of the matrix Y. Since Y is an $m \times 2$ matrix, we can do the least-square solution cheaply in just two iterations of the Grant-Schmidt algorithm (Meng et al. 2002). Since the detailed computational algorithm discussion is beyond our scope, for readers who are interested in details of this computation, please refer to (Kamvar et al. 2003) and (Trefethen and D 1997).

4.5 Exploring the Block Structure of the Web for Computing PageRank

Although the quadratic extrapolation method in Sect. 4.4 provides an approach of quickly computing the PageRank, the speedups from this method are modest for the parameter settings (i.e. d =0.25) typically used for PageRank (Kamvar et al. 2003). Another approach that makes use of Webpage block structure information for speeding up PageRank computation was proposed by (Kamvar et al. 2003).

It was noticed through the investigation conducted by (Kamvar et al. 2003) that the Web link graph has a nested block structure: the vast majority of hyperlinks link pages on a host to other pages on the same host, and many of those that do not link pages within the same domain. The pages in the same block are heavily linked. Actually in (Kamvar et al. 2003), dotplot were used to visualize the link matrix. In a dotplot, if there exists a link from page i to page j then point (i, j) is colored; otherwise, point (i, j) is white. Fig. 4.2 (Rennie J. and A. McCallum 1999) gives an example of a slice for Stanford/Berkeley Host Graph. It can be seen from this example that:

- There is a definite block structure to the Web.
- The individual blocks are much smaller than entire Web.
- There are clear nested blocks corresponding to domains, hosts, and subdirectories within the path.

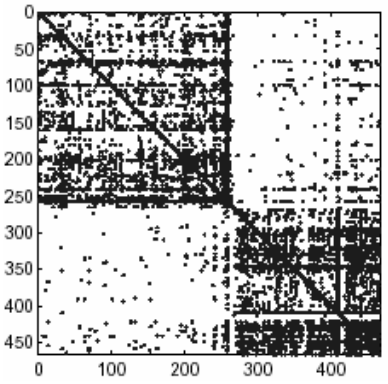


Fig. 4.2. Stanford/Berkeley Host Graph

The purpose of the algorithm in (Kamvar et al. 2003) is to directly exploit this kind of structure information to achieve large speedups compared with previous algorithms for computing PageRank.

For better understanding the idea of the algorithm, we recall the PageRank computation here in terms of matrix expression. As we know, the PageRank

computation is equivalent to the computation of principal eigenvector of the transition matrix A^T, which is defined as follow (Sect. 4.1):

$$A = (1-d)\left[\frac{1}{m}\right]_{m\times m} + dM,$$

$$\text{with } M_{i,j} = \begin{cases} \dfrac{1}{N(i)} & \text{if page } i \text{ points to page } j \\ 0 & \text{otherwise} \end{cases}$$

If we denote

$$\vec{v} = \left[\frac{1}{m}\right]_{m\times 1} \text{ and } E = [1]_{m\times 1} \times \vec{v}^T,$$

then the matrix A can be rewritten as

$$A = dM + (1-d)\,E.$$

As indicated in Sect. 4.1, in terms of the random walk, the effect of E is as follows. At each time step, with probability $(1-d)$, a surfer visiting any node will jump to a random Web page (rather than following an outlink). The destination of the random jump is chosen according to the probability distribution given in $\vec{v}$. In this original PageRank computation, the vector $\vec{v}$ is uniformed. However, if the vector $\vec{v}$ could be redefined to be non-uniform, then E adds artificial non-uniform probabilities and the resultant PageRank vector could be biased to prefer certain kinds of pages. This is the idea based on which the algorithm in (Kamvar et al. 2003) was developed. For this reason, the vector $\vec{v}$ is referred to as the *personalization vector*.

In practical situation, the power method for PageRank computation can be rewritten as the following algorithm (Kamvar et al. 2003):

Function ***pageRank***(G, $\vec{x}^{(0)}$, $\vec{v}$){
Construct matrix M from G where G is the Web page graph;
Repeat
$\vec{x}^{(k+1)} = dM^T\vec{x}^{(k)}$;
$w = \| \vec{x}^{(k)} \|_1 - \| \vec{x}^{(k+1)} \|_1$;
$\vec{x}^{(k+1)} = \vec{x}^{(k+1)} + w\vec{v}$;
$\delta = \| \vec{x}^{(k+1)} - \vec{x}^{(k)} \|_1$;
Until $\delta < \varepsilon$;
Return $\vec{x}^{(k+1)}$;
}

The overview of this algorithm that exploits the block structure information is as follow (Kamvar et al. 2003): for each host, a "local PageRank vector" is

computed, giving the relative importance of pages within a host. These local PageRank vectors can then be used to form an approximation to the global PageRank vector that is used as a starting vector for the standard Page-Rank computation. This is the basic idea behind the Block-Rank algorithm, which is summarized as the following procedures:

1. Split the Web into blocks by domain.
2. Compute the Local PageRanks for each block.
3. Estimate the relative importance, or "BlockRank" of each block.
4. Weight the Local PageRanks in each block by the Block-Rank of that block, and concatenate the weighted Local PageRanks to form an approximate Global PageRank vector $\vec{z}$.
5. Use $\vec{z}$ as a starting vector for standard PageRank computation.

The details are described below (Kamvar et al. 2003). Before that, some notations that will be used are introduced here. We will use lower-case indices (i.e. *i*, *j*) to represent indices of individual Web sites, and upper case indices (i.e. *I*, *J*) to represent indices of blocks. We use the notation $i \in I$ to denote that page $i \in$ block *I*. The number of elements in block *J* is denoted n_J. The graph of a given block *J* is given by the $n_J \times n_J$ submatrix G_{JJ} of the matrix *G*.

Computing Local PageRanks This step computes a "local PageRank vector" for each block in the Web. Since most blocks have very few links in and out of the block as compared to the number of links within the block, it seems plausible that the relative rankings of most of the pages within a block are determined by the inter-block links.

We define the *local PageRank vector* $\vec{l}_J$ of a block *J* (G_{JJ}) to be the result of the PageRank algorithm applied only on block *J*, as if block *J* represented the entire Web, and as if the links to pages in other blocks did not exist. That is:

$$\vec{l}_J = \textbf{\textit{pageRank}}(G_{JJ}, \vec{s}_J, \vec{v}_J),$$

where the start vector $\vec{s}_J$ is the $n_J \times 1$ uniform probability vector over pages in block *J* ($[1/n_J]_{nJ\times 1}$), and the personalization vector $\vec{v}_J$ is the $n_J \times 1$ vector whose elements are all zero except the element corresponding to the root node (page) of block *J*, which is also the root page of the host, whose value is 1.

Estimating the Relative Importance of Each Block This step computes the relative importance, or *BlockRank*, of each block. Assume there are *k* blocks in the Web. To compute BlockRanks, we first create the block graph *B*, where each vertex in the graph corresponds to a block in the Web graph. An edge between two pages in the Web is represented as an edge between the

corresponding blocks (or a self-edge, if both pages are in the same block). The edge weights are determined as follows: the weight of an edge between blocks I and J is defined to be the sum of the edge-weights from pages in I to pages in J in the Web graph, weighted by the local PageRanks of the linking pages in block I. That is, if A is the Web graph and l_i is the local PageRank of page i in block I, then weight of edge B_{IJ} is given by:

$$B_{IJ} = \sum_{i \in I,\, j \in J} A_{ij} \cdot l_i \,.$$

From this expression, the block matrix $B = [B_{IJ}]_{k \times k}$ can be written in matrix notation as follows. Define the local PageRank matrix L to be the $m \times k$ matrix whose columns are the local PageRank vectors $\vec{l}_J$, i.e.

$$L = \begin{pmatrix} \vec{l}_1 & \vec{0} & \cdots & \vec{0} \\ \vec{0} & \vec{l}_2 & \cdots & \vec{0} \\ \vdots & \vdots & \ddots & \vdots \\ \vec{0} & \vec{0} & \cdots & \vec{l}_k \end{pmatrix}.$$

Define the matrix S to be the $m \times k$ matrix that has the same structure as L, but whose nonzero entries are all replaced by 1. Then the block matrix B can be computed as:

$$B = L^T A S.$$

Once we have the $k \times k$ transition matrix B, we may use the standard PageRank algorithm on this reduced matrix to compute the BlockRank vector $\vec{b}$:

$$\vec{b} = \textbf{\textit{pageRank}}(B,\ \vec{v}_k,\ \vec{v}_k),$$

where $\vec{v}_k$ is the uniform k-vector $[1/k]_{k \times 1}$.

Approximating Global PageRank using Local PageRank and BlockRank This step finds an estimate to the global PageRank vector $\vec{x}$. At this point, we have two sets of rankings. Within each block J, we have the local PageRanks $\vec{l}_J$ of the pages in the block. We also have the BlockRank vector $\vec{b}$, whose elements b_J are the BlockRank for each block J, measuring the relative importance of the blocks. We may now approximate the global PageRank of a page $j \in J$ as its local PageRank l_j, weighted by the BlockRank b_J of the block in which it resides. That is,

$$x^{(0)}_{\mathrm{j}} = l_j \cdot b_J.$$

In matrix notation, this is:

$$\vec{x}^{(0)} = L\vec{b}\,.$$

Recall that the local PageRanks of each block sum to 1. Also, the BlockRanks sum to 1. It is obvious that the above approximate global PageRanks will also sum to 1 (Kamvar et al. 2003). Therefore, we may use the approximate global PageRank vector $\vec{x}^{(0)}$ as a start vector for the standard PageRank algorithm.

Computing Global PageRank In order to compute the true global PageRank vector $\vec{x}$ from the approximate PageRank vector $\vec{x}^{(0)}$, we simply use it as a start vector for standard PageRank. That is:

$$\vec{x} = \textbf{\textit{pageRank}}(G, \vec{x}^{(0)}, \vec{v}_J)$$

where G is the graph of the Web, and $\vec{v}_J$ is the uniform distribution over root nodes.

(Kamvar et al. 2003) claimed the advantages of the BlockRank algorithm as follows:

1. A major speedup of the algorithm comes from caching effects. All of the host-blocks in the crawl are small enough so that each block graph fits in main memory, and the vector of ranks for the active block largely fits in the CPU cache. As the full graph does not fit entirely in main memory, the local PageRank iterations thus require less disk i/o than the global computations.
2. In the BlockRank algorithm, the local PageRank vectors for many blocks will converge quickly; thus the computations of those blocks may be terminated after only a few iterations.
3. The local PageRank computations in step 1 of the BlockRank algorithm can be computed in a completely parallel or distributed fashion.
4. In several scenarios, the local PageRank computations (e.g., the results of Step 1) can be reused during future applications of the BlockRank algorithm.

The experiments in (Kamvar et al. 2003) showed that the BlockRank algorithm yielded significant speedup over the standard PageRank.

4.6 Web Page Scoring Systems (WPSS)

As discussed before, page ranking is a fundamental step towards the construction of effective search engines for both generic (horizontal) and focused (vertical) search. But the original PageRank algorithm only considers the situation where the Web surfer either follows the links included in the current visiting page or jump to another page (node) in the Web graph. The surfer will never go back along the reverse links from the current visiting page or

stay on the current page. In (Diligenti et al. 2002), a general framework for Web page scoring systems (WPSS) was proposed based on a random walk model. This random walk considers all possible actions the surfer may take. Therefore the original PageRank and HITS algorithms are all the special cases of this general framework. Furthermore, this general framework can derive different algorithms for effective horizontal and vertical searches.

In the WPSS, a complete model of the behavior of a user surfing the Web is considered. We assume that a Web surfer can perform one out of four atomic actions at each step of his/her traversal of the Web graph (Diligenti et al. 2002):

- j jump to a node of the graph;
- l follow a hyperlink from the current page;
- b follow a back-link (a hyperlink in the inverse direction);
- s stay in the same node.

Then the set of the atomic actions used to move on the Web is defined as $O = \{j, l, b, s\}$. From these surfer's actions, the surfer's behaviour can be modelled by a set of probabilities which depend on the current page:

- $x(l|q)$ the probability of following one hyperlink from page q,
- $x(b|q)$ the probability of following one back-link from page q,
- $x(j|q)$ the probability of jumping from page q,
- $x(s|q)$ the probability of remaining in page q.

These values must satisfy the normalization constraint

$$\sum_{o \in O} x(o \mid q) = 1$$

Most of these actions need to specify their targets. Assuming that the surfer's behavior is time-invariant, then we can model the targets for jumps, hyperlink or back-link choices by using the following parameters (Diligenti et al. 2002):

- $x(p|q, j)$ the probability of jumping from page q to page p;
- $x(p|q, l)$ the probability of selecting a hyperlink from page q to page p; this value is not null only for the pages p linked directly by page q, i.e. $p \in ch(q)$, being $ch(q)$ the set of the children of node q in the graph G;
- $x(p|q, b)$ the probability of going back from page q to page p; this value is not null only for the pages p which link directly page q, i.e. $p \in pa(q)$, being $pa(q)$ the set of the parents of node q in the graph G.

These sets of values must satisfy the following probability normalization constraints for each page $q \in G$:

$$\sum_{p\in G} x(p \mid q, j) = 1, \quad \sum_{p\in ch(q)} x(p \mid q, l) = 1, \quad \sum_{p\in pa(q)} x(p \mid q, b) = 1.$$

The above model considers a temporal sequence of actions performed by the surfer. The probability that the surfer is located in page p at time t+1, $x_p(t+1)$, can be modelled using the following equation:

$$x_p(t+1) = \sum_{q\in G} x(p \mid q) \cdot x_q(t),$$

where the probability $x(p|q)$ of going from page q to page p is obtained by considering the action which can be performed by the surfer. Considering all possible actions that surfer may take, the equation can be rewritten as

$$x_p(t+1) = \sum_{q\in G} x(p \mid q, j) \cdot x(j \mid q) \cdot x_q(t) + \sum_{q\in G} x(p \mid q, l) \cdot x(l \mid q) \cdot x_q(t) + \sum_{q\in G} x(p \mid q, b) \cdot x(b \mid q) \cdot x_q(t) + x(s \mid p) \cdot x_p(t).$$

Suppose the number of pages in the Web graph G is N. The page probabilities in G at time t can be collected in a N-dimensional vector $\boldsymbol{x}(t)$. Actually, the above probability update equations can be rewritten in a matrix form. To this end, firstly we define some matrices (Diligenti et al. 2002):

- the *forward* matrix Δ whose element (p, q) is the probability $x(p|q, l)$;
- the *backward* matrix Γ collecting the probabilities $x(p|q, b)$;
- the *jump* matrix Σ which is defined by the jump probabilities $x(p|q, j)$.

The forward and backward matrices are related to the Web adjacency matrix $\boldsymbol{A}$ whose entries are 1 if page p links page q. In particular, the forward matrix Δ has non-null entries only in the positions corresponding to 1s in matrix $\boldsymbol{A}$, and the backward matrix Γ has non null entries in the positions corresponding to 1s in $\boldsymbol{A}^T$.

Further, we can define other matrices that collect the probabilities of taking one of the possible actions from a given page q. These are $N\times N$ diagonal matrices defined as follows: $\boldsymbol{D}_j$ whose diagonal values (q, q) are the probabilities $x(j|q)$, $\boldsymbol{D}_l$ collecting the probabilities $x(l|q)$, $\boldsymbol{D}_b$ containing the values $x(b|q)$, and $\boldsymbol{D}_s$ having on the diagonal the probabilities $x(s|q)$. Hence, the above probability update equations can be written in matrix form as

$$\boldsymbol{x}(t+1) = (\Sigma \cdot \boldsymbol{D}_j)^T \boldsymbol{x}(t) + (\Delta \cdot \boldsymbol{D}_l)^T \boldsymbol{x}(t) + (\Gamma \cdot \boldsymbol{D}_b)^T \boldsymbol{x}(t) + (\boldsymbol{D}_s)^T \boldsymbol{x}(t).$$

If we define the transition matrix $\boldsymbol{T}$ that is used to update the probability distribution as follows:

$$\boldsymbol{T} = (\Sigma \cdot \boldsymbol{D}_j + \Delta \cdot \boldsymbol{D}_l + \Gamma \cdot \boldsymbol{D}_b + \boldsymbol{D}_s)^T.$$

Then the above probability update equation can be written as

$$\boldsymbol{x}(t+1) = \boldsymbol{T} \cdot \boldsymbol{x}(t). \tag{4.4}$$

Starting from a given initial probability distribution $\boldsymbol{x}(0)$, this probability update equation can be applied recursively to compute the probability distribution at a given time step t yielding

$$\boldsymbol{x}(t) = \boldsymbol{T}^{t} \cdot \boldsymbol{x}(0).$$

The absolute page rank for the pages on the Web is then the stationary probability distribution of this recursive procedure. (Diligenti et al. 2002) gave the following proposition that guarantees the convergence of the recursive operation. Readers who are interested in the proof detail of this proposition can refer to (Diligenti et al. 2002).

PROPOSITION. *If* $x(j|q) \neq 0$ *and* $x(p|q, j) \neq 0$, $\forall p, q \in G$, *then* $\lim_{t \to \infty} \boldsymbol{x}(t) = \boldsymbol{x}^*$, *where* $\boldsymbol{x}^*$ *does not depend on the initial state vector* $\boldsymbol{x}(0)$.

Uniform Jump Probabilities In the application of this general framework, some assumptions can be introduced on the probability matrices. A possible choice is to consider some actions to be independent on the current page. A first choice is the uniform jump probability that is independent on the starting page q. This choice models a surfer who decides to make random jumps from a given page to another page with uniform probability. Thus, the jump matrix Σ has all the entries equal to $x(p|q, j) = x(p|j) = 1/N$. Moreover, we also suppose that the probability of choosing a jump among the available actions does not depend on the page, i.e. $x(j|p) = d_j$, $\forall p \in G$. Under these two assumptions, the probability update equation becomes

$$x(t+1) = (d_j / N) \cdot \Lambda + (\Delta \cdot \boldsymbol{D}_l + \Gamma \cdot \boldsymbol{D}_b + \boldsymbol{D}_s)^{T} x(t)$$

since $(\Sigma \cdot \boldsymbol{D}_j)^{T} \cdot \mathrm{x(t)} = (d_j / N) \cdot \Lambda$ as $\sum_{p \in G} x_p(t) = 1$, where Λ is a N-dimensional vector whose entries are all 1s. If we denote $\boldsymbol{R} = (\Delta \cdot \boldsymbol{D}_l + \Gamma \cdot \boldsymbol{D}_b + \boldsymbol{D}_s)^{T}$, it can be proved (see (Diligenti et al. 2002)) that the probability distribution $\boldsymbol{x}(t)$ converges to

$$\boldsymbol{x}^* = (d_j / N)\, (\boldsymbol{I} - \boldsymbol{R})^{-1} \cdot \Lambda$$

and the value of $\boldsymbol{x}^*$ does not depend on the choice of the initial state vector $\boldsymbol{x}(0)$.

Multiple State Model A model based on a single variable may not capture the complex relationships among Web pages when trying to model their importance. The probabilistic framework described above can also be extended to a multivariable scheme by considering a pool of Web surfers, each described by a single variable. These surfers can be used to simulate different Web browsing actions that reveal different relationships among Web pages. In particular, each surfer is characterized by his/her own way of browsing the Web modeled by using different parameter values in each state transition equation. These parameters represent different policies in evaluating the absolute importance of the pages. Moreover, the surfers may interact by accepting *suggestions* from each other(Diligenti et al. 2002).

In order to model the activity of M different surfers, (Diligenti et al. 2002) used a set of state variables $x_q^{(i)}(t)$ ($i = 1, \ldots, M$) which represent the probability of each surfer i to be visiting page q at time t. The interaction among the surfers is modeled by a set of parameters which define the probability of the surfer k to accept the suggestion of the surfer i, thus jumping from the page he/she is visiting to the one visited by the surfer i. This interaction happens before the choice of the actions described previously. If we hypothesize that the interaction does not depend on the page the surfer k is currently visiting, the degree of interaction with the surfer i is modeled by the value $b(i|k)$ which represents the probability for the surfer k of jumping to the page visited by the surfer i. These values must satisfy the probability normalization constraint $\sum_{s=1}^{M} b(s \mid k) = 1$.

Before taking any action, the surfer i probably repositions himself/herself in page p looking at the suggestions of the other surfers. This probability is computed as

$$v_p^{(i)}(t) = \sum_{s=1}^{M} b(s \mid i) x_p^{(s)}(t).$$

Due to the action taken by the surfer i at time t, the probability distribution $x_p^{(i)}(t+1)$ is computed by replacing $x_p^{(i)}(t)$ with $v_p^{(i)}(t)$ in the original probability update equation. Thus, the transition function is defined as follows:

$$x_p^{(i)}(t+1) = [\sum_{q \in G} x^{(i)}(p \mid q, j) \cdot x^{(i)}(j \mid q) + \sum_{q \in G} x^{(i)}(p \mid q, l) \cdot x^{(i)}(l \mid q) + \sum_{q \in G} x^{(i)}(p \mid q, b) \cdot x^{(i)}(b \mid q) + x^{(i)}(s \mid p)] \cdot \sum_{s=1}^{M} b(s \mid i) x_q^{(s)}(t),$$

$$i = 1, \ldots, M.$$

These equations can also be expressed in matrix format. We define a $M \times M$ matrix $\boldsymbol{B}$ which collects the values $b(i|k)$. Matrix $\boldsymbol{B}$ is also referred as *interaction matrix*. For each surfer i, his/her own forward, backward and jump matrices are denoted as $\Delta^{(i)}$, $\Gamma^{(i)}$, $\Sigma^{(i)}$, and the action matrices are denoted as $\boldsymbol{D}_j^{(i)}$, $\boldsymbol{D}_l^{(i)}$, $\boldsymbol{D}_b^{(i)}$, $\boldsymbol{D}_s^{(i)}$ respectively. Thus the transition matrix for the surfer i is defined as

$$\boldsymbol{T}^{(i)} = (\Sigma^{(i)} \cdot \boldsymbol{D}_j^{(i)} + \Delta^{(i)} \cdot \boldsymbol{D}_l^{(i)} + \Gamma^{(i)} \cdot \boldsymbol{D}_b^{(i)} + \boldsymbol{D}_s^{(i)})^T.$$

Using these definitions, the set of M interacting surfers can be described as a set of matrix equations as follows:

$$\begin{cases} x^{(1)}(t+1) = \boldsymbol{T}^{(1)} \cdot x^{(1)}(t) \cdot \boldsymbol{B}^{(1)} \\ \vdots \\ x^{(M)}(t+1) = \boldsymbol{T}^{(M)} \cdot x^{(M)}(t) \cdot \boldsymbol{B}^{(M)} \end{cases} \tag{4.5}$$

From this expression, it can be easily seen that if the surfers are independent on each other, i.e. $b(i|i) = 1$ and $b(i|j) = 0$ for $i \neq j$, this model is then reduced to the original model (4).

The two models (4) and (5) can be used to derive various algorithms for page ranking. In (Diligenti et al. 2002) two kinds of page ranking situations were considered: horizontal WPSS and vertical WPSS. Horizontal WPSSs do not consider any information on the page contents and produce the rank vector using just the topological characteristics of the Web graph. Vertical WPSSs aim at computing a relative ranking of pages when focusing on a specific topic. As indicated in (Diligenti et al. 2002) the original PageRank and HITS algorithms, which are the representatives of horizontal WPSS, are all the special cases of the general framework (4) and (5).

As the two representatives of horizontal WPSS, PageRank and HITS have their own advantages and disadvantages. The PageRank is stable, it has a well defined behavior because of its probabilistic interpretation and it can be applied to large page collections without canceling the influence of the smallest Web communities. For HITS, the hub and authority model can capture more than PageRank the relationships among Web pages. On the other hand, PageRank is sometimes too simple to take into account the complex relationships of Web page citations. HITS is not stable, only the largest Web community influences the ranking, and this does not allow the application of HITS to large page collections. It is amazing that the above general probability framework (4) and (5) can derive a new algorithm that includes the advantages of both approaches. In (Diligenti et al. 2002), it was called PageRank-HITS model. The derivation of this model is described as follows (Diligenti et al. 2002).

We employ two surfers, each one implementing a bidirectional PageRank surfer. We assume that surfer 1 either follows a back-link with probability $x^{(1)}(b|p) = d^{(1)}$ or jumps to a random page with probability $x^{(1)}(j|p) = 1 - d^{(1)}$, $\forall p \in G$. Whereas surfer 2 either follows a forward link with probability $x^{(2)}(l|p) = d^{(2)}$ or jumps to a random page with probability $x^{(2)}(j|p) = 1 - d^{(2)}$ $\forall p \in G$. The interaction between the surfers is described by the matrix

$$\boldsymbol{B} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

The interpretation of the interactions represented by this matrix is that surfer 1 consider surfer 2 as an expert in discovering authorities and always moves to the position suggested by that surfer before acting. On the other hand, surfer 2 considers surfer 1 as an expert in finding hubs and always moves to the position suggested by that surfer before choosing the next action. In this case, the equation (5) becomes

$$\begin{cases} \boldsymbol{x}^{(1)}(t+1) = \dfrac{1-d^{(1)}}{N}\boldsymbol{\Lambda} + d^{(1)}(\boldsymbol{\Gamma}^{(1)})^{T} \cdot \boldsymbol{x}^{(2)}(t), \\ \boldsymbol{x}^{(2)}(t+1) = \dfrac{1-d^{(2)}}{N}\boldsymbol{\Lambda} + d^{(2)}(\boldsymbol{\Delta}^{(2)})^{T} \cdot \boldsymbol{x}^{(1)}(t). \end{cases}$$

If we assume the independence of parameters $x^{(1)}(p|q, b)$ and $x^{(2)}(p|q, l)$ on the page p, then it holds that $\Delta^{(1)\mathrm{T}} = \boldsymbol{A}^{\mathrm{T}} \cdot \Theta$, $\Gamma^{(2)\mathrm{T}} = \boldsymbol{A} \cdot \Omega$, where Ω is the diagonal matrix with element (p, p) equals to $1 / pa(p)$ and Θ is the diagonal matrix with the element (p, p) equals to $1/ ch(p)$. Therefore, we have

$$\begin{cases} \boldsymbol{x}^{(1)}(t+1) = \dfrac{1-d^{(1)}}{N}\boldsymbol{\Lambda} + d^{(1)}\mathbf{A} \cdot \boldsymbol{\Omega} \cdot \boldsymbol{x}^{(2)}(t), \\ \boldsymbol{x}^{(2)}(t+1) = \dfrac{1-d^{(2)}}{N}\boldsymbol{\Lambda} + d^{(2)}\mathbf{A}^{T} \cdot \boldsymbol{\Theta} \cdot \boldsymbol{x}^{(1)}(t). \end{cases}$$

As indicated in (Diligenti et al. 2002), this page rank is stable, the score sum up to 1 and no normalization is required at the end of each iteration. Moreover, the two state variables can capture and process more complex relationships among pages. In particular, if we set $d^{(1)} = d^{(2)} = 1$, the above equations will yield a normalized version of HITS.

For vertical WPSS, it relies on the representation of the page content with set of features, such as a set of keywords, and on a classifier which is used to assess the degree of relevance of the page with respect to the topic of interest. In (Diligenti et al. 2002), this vertical WPSS was also called focused PageRank. The idea of it is as follow (Diligenti et al. 2002). In the PageRank framework, when choosing to follow a link in a page q each link has the same probability $1 / ch(q)$ to be followed. Instead of the *random* surfer model, in the focused domain we can consider the more realistic case of a surfer who follows the links according to the suggestions provided by a page classifier. If the surfer is located at page q and the pages linked by page q have scores $s(ch_1(\mathrm{q}))$, …, $\mathrm{s}(ch_{hq}(q))$ by a topic classifier where h_q is the number of children of q, the probability of the surfer to follow the i-th link is defined as

$$x(ch_i(q) \mid q, l) = \frac{s(ch_i(q))}{\sum_{j=1}^{hq} s(ch_j(q))}. \tag{4.6}$$

Thus the forward matrix Δ will depend on the classifier outputs on the target pages. Hence, the modified equation to compute the combined page scores using a PageRank-like scheme is

$$x_p(t+1) = \frac{(1-d)}{N} + d \sum_{q \in pa(p)} x(p \mid q, l) \cdot x_q(t),$$

where $x(p|q, l)$ is computed as in the above equation (6).

This scoring system removes the assumption of complete randomness of the underlying Web surfer. In this case, the surfer is aware of what he/she is searching, and he/she will trust the classifier suggestions following the links with a probability proportional to the score of the page the links leads to. This allows us to derive a topic-specific page rank.

It can be seen from the above discussions that the general probability framework (4) and (5) model considers all actions that a Web surfer may take, and many page ranking algorithms such as the original PageRank and HITS can be derived from this framework. The experimental results in (Diligenti et al. 2002) support the effectiveness of this framework.

4.7 The Voting Model

The original PageRank algorithm, as well as other improved ones, was originally based on a surfing model as discussed before. Actually, there are also other models that can be used to derive various Web page rank algorithms. Particularly, if a new derived algorithm from other models can also subsume the original PageRank, this new algorithm can then be considered as the extension of the PageRank algorithm and will have wider application areas.(Lifantsev 2000) proposed such a model that ranks Web pages from a different angle. This model was named voting model in(Lifantsev 2000). It subsumes the surfing model providing natural trivially converging computation of more different types of ranks, and provides a new meaning for the parameters and the results of the surfing model.

This voting model is mainly for ranking authority pages. The idea of this model is described as follows in (Lifantsev 2000). The nodes/pages in the Web page graph have a certain number of votes at their disposal. A node can distribute its votes among other nodes (including itself) either by assigning a portion of the votes to a destination node, or by giving a portion of the votes to a destination node for the latter to decide how to distribute it. The authority rank of a node is then the number of votes eventually assigned to it. The votes can be fractional. The reason why this model is called voting model is that it is a generalization of direct and representative voting system: a node can use

its votes directly or can give them to a representative to manage on behalf of the node; any representative in turn can also use all the votes it gets either directly or indirectly by giving them to another representative (Lifantsev 2000).

Suppose the set of nodes in the Web page graph is N. The voting model assigns each node a certain number of initial votes. We denote the total number of votes as $|N|$, and represent these initial votes by the column initial votes vector $\mathbf{G} = (g_n)$ such that $\sum_{n \in N} g_n = |N|$, $g_n \in [0, |N|]$ for $n \in N$. We assume $g_n = 1$ for $n \in N$ if not stated otherwise, i.e. we usually assume that all the nodes are equal at the beginning.

The votes a node has are then divided and transmitted along the links between the nodes. The votes are either assigned to a node or trusted to a node to manage. The vote propagations can be represented by the vote assignment matrix $\boldsymbol{V_A} = (a_{nn'})$ and the vote trust matrix $\boldsymbol{V_T} = (t_{nn'})$ such that $a_{nn'} \in [0, 1]$ is the fraction of votes assigned to node n by node n', $t_{nn'} \in [0, 1]$ is the fraction of votes trusted to n by n', and they satisfy $\sum_{n \in N} (a_{nn'} + t_{nn'}) \in [0,1]$ for any $n' \in N$. In other words, for each node the trust and assignment fractions on its out-links form an incomplete probability distribution. These two matrices are also called *vote distribution matrices*. Please note that the incomplete matrices correspond to the cases when some nodes waste some fraction of the votes they are given. It is possible that a node assigns some votes it has to itself, and it is also possible that a node trusts some votes it has back to itself.

Then the computation of the total votes eventually assigned to the nodes, which is represented by the column vote vector $\boldsymbol{V}$, can be conducted in the following way (Lifantsev 2000):

$$\boldsymbol{V_0} = \boldsymbol{0}$$
$$\boldsymbol{U_0} = \boldsymbol{G}$$
$$\boldsymbol{V_{i+1}} = \boldsymbol{V_i} + \boldsymbol{V_A} \cdot \boldsymbol{U_i},$$
$$\boldsymbol{U_{i+1}} = \boldsymbol{V_T} \cdot \boldsymbol{U_i},$$

where $\mathbf{0}$ is the column vector of all zeros and $\boldsymbol{U_i}$ is the vector representing the number of votes given to each node to manage on the ith stage of the computation. Theoretically, the vote vector $\boldsymbol{V}$, which is also the authority rank vector, is equal to $\boldsymbol{V_\infty}$.

One possible problem with this computation is that if there is a loop in the graph induced by $\boldsymbol{V_T}$ with all the edges marked with 1, we will not be able to reach a state where $\boldsymbol{V_\infty} = \mathbf{0}$, which is required for convergence. To remedy this problem, we can take one of the following three ways (Lifantsev 2000):

1. We can introduce a vote transfer degradation factor d that is slightly less than 1 and replace $\boldsymbol{U_{i+1}} = \boldsymbol{V_T} \cdot \boldsymbol{U_i}$ with $\boldsymbol{U_{i+1}} = d\, \boldsymbol{V_T} \cdot \boldsymbol{U_i}$ in the above compu-

tation. Such degradation factor is also good if we want to encourage nodes to assign the votes directly rather than entrust them.
2. We can stop the computation when $\boldsymbol{U}_{i+1}$ gets close enough to $\boldsymbol{U}_{\mathrm{i}}$.
3. We can also require that $\boldsymbol{V}_{\boldsymbol{T}}$ should not have elements that are equal to 1. It is reasonable in practice, i.e. a node does not trust all its votes to just one other node, it can assign at least some fraction of votes to itself.

As indicated in (Lifantsev 2000), this computation scheme has very easy to ensure and verify convergence conditions. The reason is that we just transfer and assign the initial votes according to $\boldsymbol{V}_{\boldsymbol{T}}$ and $\boldsymbol{V}_{\boldsymbol{A}}$ until the not yet assigned votes $\boldsymbol{U}_i$ get close enough to $\mathbf{0}$. It is different from the surfing model where all the surfers get reshuffled at each iteration and we have to reach a fix point.

There is a connection between the ranks computed using the above voting model and the surfing model. Before introducing the theorem regarding this connection, firstly we summarize the surfing model that is used in the original PageRank algorithm.

Surfing Model In this model, there are surfers that go from one node to another using the links; the more surfers a node 'has,' the higher its authority rank. Initially, we suppose there are |N| surfers that can split themselves into fractional surfers to follow different paths. They are initially distributed evenly among all the nodes if we assume that all nodes are a priori equally popular. These initial surfers can be represented by a column start surfers vector $\boldsymbol{S} = (s_n)$ such that $\sum_{n \in N} s_n = |N|$, $s_n \in [0, |N|]$ for $n \in N$. we assume $s_n = 1$ for $n \in N$ if not stated otherwise.

The surfers then travel along the links dividing themselves according to propagation fractions associated with the links. The propagation fractions are represented by a surfer propagation matrix $\boldsymbol{P} = (p_{nn'})$ such that $p_{nn'} \in [0, 1]$ is the fraction of surfer directed from node n' to n and $\sum_{n \in N} p_{nn'} \in [0,1]$ for any $n' \in N$. Therefore, for each node the propagation fractions of its out-links form an incomplete probability distribution. Such matrices are also called *column distribution matrices*. Please note that it is possible that a node directs its surfers back to itself, and that it directs its surfers equally or non-equally along all of its out-links. As we know, the original PageRank algorithm divides all the surfers equally among all the outgoing links.

We represent the authority ranks by the column rank vector $\boldsymbol{R}$, then the authority ranks are computed as follows:

$$\boldsymbol{R_0 = S}$$

$$\boldsymbol{R_{i+1} = e\, P \cdot R_i + (1 - e)\, S}$$

where e is the exploration probability of surfers.

As in the PageRank algorithm, we always reach a fixpoint such that $\boldsymbol{R}_\infty$ (theoretically) is the final page rank of pages, while $\boldsymbol{R}_\infty$ is the principal eigenvector of matrix $e\,\boldsymbol{P} + (1-e) / |N|\,\boldsymbol{S}\cdot \boldsymbol{1}^{\mathrm{T}}$ with the eigenvalue of 1, where $\boldsymbol{1}$ is the vector consisting of all 1s.

When $\boldsymbol{P}$ is incomplete column distribution matrix, i.e. $\sum_{n\in N} p_{nn'} < 1$, we can still compute the ranks as the principal eigenvector of the same matrix as above, except that its eigenvalue is going to be less than 1. It then requires that the $\boldsymbol{R}_{\mathrm{i}}$ is to be normalized up at each iteration step, so that we can prevent the fixpoint $\boldsymbol{R}_\infty$ from becoming a zero vector. Therefore, the algorithm is improved as

$$\boldsymbol{R}_0 = \boldsymbol{S}$$

$$\boldsymbol{R}_{\mathrm{i}+1} = \alpha_i\,(e\,\boldsymbol{P}\cdot\boldsymbol{R}_{\mathrm{i}} + (1-e)\,\boldsymbol{S}\,),$$

where α_i is such as to make $\|\boldsymbol{R}_{\mathrm{i}+1}\|_1 = |N|$.

This method can be slightly modified as follows:

$$\boldsymbol{R}_{\boldsymbol{0}} = \boldsymbol{S}$$

$$\boldsymbol{R}_{\boldsymbol{i+1}} = \boldsymbol{e}\,\boldsymbol{P}\cdot\boldsymbol{R}_{\boldsymbol{i}} + \alpha_i\,\boldsymbol{S},$$

where α_i is such as to make $\|\boldsymbol{R}_{\mathrm{i}+1}\|_1 = |N|$. The difference here is that we compensate for the loss of unpropagated surfers by adding surfers proportionally to S, rather than by consistently increasing the number of surfers in each node by the same factor. This algorithm is based on the surfers who surf the Web along the links, so it is named surfing model.

(Lifantsev 2000) indicated the strong connection between the ranks computed by the surfing and voting models:

Theorem *If* $\boldsymbol{P}$ *is a column distribution matrix, then the surfer rank vector* $\boldsymbol{R} = \boldsymbol{P}\cdot\boldsymbol{R}_\infty$, *where* $\boldsymbol{R}_\infty$ *is omputed using the modified eigenvalue method with exploration probability* $e \in (0, 1)$ *and some start surfers vector* $\boldsymbol{S}$, *is equal to the vote vector* $\boldsymbol{V}_\infty$ *which is computed for* $\boldsymbol{V}_T = e\boldsymbol{P}$ *and* $\boldsymbol{V}_A = (1-e)\boldsymbol{P}$ *with* $\boldsymbol{G} = \boldsymbol{S}$ *in the voting model.*

Readers who are interested in the proof details of this theorem please refer to (Lifantsev 2000).

From this theorem, it can be seen that (Lifantsev 2000):

- The surfing model is a sub-case of the voting model.
- The exploration probability e is the fraction of the votes (authority ranks) that a node trust the destinations of its out-links to manage, whereas $1 - e$ is the fraction of the votes (ranks) a node assigns directly to the destinations of its out-links.
- In the case of the surfing model, the trust-assignment split is set for all the nodes uniformly by the model. In the voting model, each node can itself potentially set the trust-assignment split individually for each of its out-links. In practice, this can possibly be implemented by extracting some

hints about the intended trust-assignment split from the text surrounding the links).

- The surfing model is a bit unnatural in that it corresponds to the voting model where each node trusts some fraction of the votes it has back to itself to redistribute these votes again and again. Hence, we can construct a more natural model that uses exactly the same data as the surfing model by letting $\boldsymbol{V_T} = e\boldsymbol{P}$ *and* $\boldsymbol{V_A} = (1 - e)\boldsymbol{P}$ except that we clean $\boldsymbol{V_T}$'s diagonal adding those "self-trusts" to $\boldsymbol{V_A}$'s diagonal.

In general, the voting model seems to be better suited for collection of various statistics (refer to (Lifantsev 2000)) as it relies on simple direct iterative computation where at each iteration smaller and smaller corrections are added to the result, whereas the surfing model is based on fixpoint iterative computation with non-trivial convergence criteria where at each iteration all the results are recomputed.

4.8 Using Non-Affliated Experts to Rank Popular Topics

The original PageRank algorithm, as well as other improved or derived page ranking algorithms discussed before, tries to globally rank Web pages. This ranking is query independent. Although the absolute page ranking is helpful in improving Web searching, in practical use, users would more likely to get the page rank with respect to user queries or a given page. Since PageRank is query-independent it cannot by itself distinguish between pages that are authoritative in general and pages that are authoritative on the query or given page topic. Therefore, it is necessary to consider page ranking that is related to user queries or a given page. In this section and next section, we will introduce two approaches that were proposed to rank Web pages with respect to the user query and a given page. This kind of page ranking algorithm can be considered as a complimentary to the original page ranking algorithms.

In (Bharat and GA 2001), an approach named Hilltop, was proposed to rank Web pages that are to be related to a user query topic. The idea of this approach is based on the same assumptions as other connectivity algorithms such as HITS and PageRank, that is the number and quality of the sources referring to a page are a good measure of the page's quality. The difference between this approach and other algorithms is that it only uses "expert" sources/pages to measure the page's quality. An expert page is the page that is about a certain topic and has links to many non-affiliated pages on that topic. The expert pages are created with the specific purpose of directing people towards resources.

For a user's query, this approach first computes a list of the most relevant experts on the query topic. Then relevant links are identified within the selected set of experts, and target Web pages are identified as well by following these relevant links. The targets are then ranked according to the number and relevance of non-affiliated experts that point to them. Thus, the score of a target page reflects the collective opinion of the best independent experts on the query topic. Accordingly, the Hilltop algorithm consists of two phases: expert lookup and query processing.

Expert Lookup An expert page needs to be objective and diverse: that is, its recommendations should be unbiased and point to numerous *non-affiliated* pages on the subject. Therefore, in order to find the experts, it is necessary to detect when two sites belong to the same or related organizations. (Bharat and GA 2001) defined two hosts as affiliated if one or both of the following is true:

- They share the same first three octets of the IP address.
- The rightmost non-generic token in the hostname is the same.

Tokens are considered as the substrings of the hostname delimited by "." (period). A suffix of the hostname is considered generic if it is a sequence of tokens that occur in a large number of distinct hosts. For instance, ".com" and ".co.uk" are domain names that occur in a large number of hosts and are hence generic suffixes. Given two hosts, if the generic suffix in each case is removed and the subsequent rightmost token is the same, we consider them to be affiliated. For example, given two hostname "www.ibm.com" and "ibm.co.mx", we remove the generic suffixes ".com" and ".co.mx" respectively. The resulting rightmost token is "ibm", which is the same in both cases. Hence they are considered to be affiliated. Optionally, we could require the generic suffix to be the same in both cases.

The affiliation relation is transitive: if A and B are affiliated and B and C are affiliated then we can infer that A and C to be affiliated even if there is no direct evidence of the fact. In practice, this may cause some non-affiliated hosts to be classified as affiliated. This may also happen, for example, if multiple, independent Web sites are hosted by the same service provider. However, this is acceptable since this relation is intended to be conservative (Bharat and GA 2001).

In a preprocessing step, the Hilltop algorithm (Bharat and GA 2001) constructs a host-affiliation lookup. Using a union-find algorithm we group hosts that either share the same rightmost non-generic suffix or have an IP address in common into sets. Every set is given a unique identifier. The host-affiliation lookup maps every host to its set identifier or to itself when there is no set. This is used to compare hosts. If the lookup maps two hosts to the same value then they are affiliated; otherwise they are nonaffiliated.

After the preprocessing step, we can turn to selecting experts from a search engine's database of pages. In (Bharat and GA 2001), the *AltaVista*'s crawl from April 1999 was used. This is done as follows:

> "Considering all pages with out-degree greater than a threshold, k (e.g., $k = 5$), we test to see if these URLs point to k distinct *nonaffiliated* hosts. Every such page is considered an expert page."

If a broad classification (such as *Arts*, *Science*, *Sports,* etc.) is known for every page in the search engine database then we can additionally require that most of the k non-affiliated URLs discovered in the previous step point to pages that share the same broad classification. This allows us to distinguish between random collections of links and resource directories.

To locate expert pages that match user queries, Hilltop (Bharat and GA 2001) created an inverted index to map keywords to experts on which they occur. In doing so only text contained within "key phrases" of the expert is indexed. A key phrase is a piece of text that qualifies one or more URLs in the page. Every key phrase has a scope within the document text. URLs located within the scope of a phrase are said to be "qualified" by it. For example, the title, headings (e.g., text within a pair of <H1> </H1> tags) and URL anchor text within the expert page are considered key phrases. The title has a scope that qualifies all URLs in the document. A heading's scope qualifies all URLs until the next heading of the same or greater importance. An anchor's scope only extends over the URL it is associated with. The inverted index is organized as a list of match positions within experts. Each match position corresponds to an occurrence of a certain keyword within a key phrase of a certain expert page. All match positions for a given expert occur in sequence for a given keyword. At every match position we also store:

1. An identifier to identify the phrase uniquely within the document.
2. A code to denote the kind of phrase it is (title, heading or anchor).
3. The offset of the word within the phrase.

In addition, for every expert Hilltop maintains the list of URLs within it (as indexes into a global list of URLs), and for each URL the identifiers of the key phrases that qualify it are also maintained. To avoid giving long key phrases an advantage, the number of keywords within any key phrase is limited, such as to 32.

Query Processing The first step that responses to a user query is to determine a list of N (e.g. $N = 200$) experts that are the most relevant for that query. Then we rank results by selectively following the relevant links from these experts and assigning an authority score to each such page. The details of this query processing procedure are presented as follows (Bharat and GA 2001).

For an expert to be useful in response to a query, the minimum requirement is that there is at least one URL which contains all the query keywords in the key phrases that *qualify* it. A fast approximation is to require all query keywords to occur in the document.

The score of an expert is computed as a 3-tuple of the form (S_0, S_1, S_2). Let k be the number of terms in the input query q. The component S_i of the score is computed by considering only key phrases that contain precisely $k - i$ of the query terms. For example, S_0 is the score computed from phrases containing all the query terms. That is

$$S_i = \Sigma_{\{\text{key phrases } p \text{ with } k - i \text{ query terms}\}} \mathit{LevelScore}(p) * \mathit{FullnessFactor}(p, q),$$

where *LevelScore*(*p*) is a score assigned to the phrase by virtue of the type of phrase it is. For example, in (Bharat and GA 2001) a *LevelScore* of 16 was used for title phrases, 6 for headings and 1 for anchor text. This was based on the assumption that the title text is more useful than the heading text, which is more useful than an anchor text match in determining what the expert page is about. *FullnessFactor*(*p, q*) is a measure of the number of terms in p covered by the terms in q. Let *plen* be the length of p. Let m be the number of terms in p which are not in q (i.e., surplus terms in the phrase). Then, *FullnessFactor*(*p, q*) is computed as follows:

- If m <= 2, FullnessFactor(p, q) = 1.
- If m > 2, FullnessFactor(p, q) = 1 – (m – 2) / plen.

The goal of the expert score is to prefer experts that match all of the query keywords over experts that match all but one of the keywords, and so on. Hence experts are ranked first by S_0. We break ties by S_1 and further ties by S_2. The score of each expert is converted to a scalar by the weighted summation of the three components:

$$\mathit{Expert\_Score} = 2^{32} * S_0 + 2^{16} * S_1 + S_2.$$

From the top N (e.g. $N = 200$) experts selected from their expert scores, we can then examine the pages they point to. These pages are called *targets*. It is from this set of targets that we select top ranked pages regarding the user query. For a target to be considered it must be pointed to by at least two experts on hosts that are mutually non-affiliated and are not affiliated to the target. For all qualified targets we compute a target score T in three steps (Bharat and GA 2001):

1. For every expert E that points to target T we draw a directed edge (E,T). Consider the following "qualification" relationship between key phrases and edges:

- The title phrase *qualifies* all edges coming out of the expert.

- A heading *qualifies* all edges whose corresponding hyperlinks occur in the document *after* the given heading and *before* the next heading of equal or greater importance.
- A hyperlink's anchor text *qualifies* the edge corresponding to the hyperlink.

For each query keyword *w*, let *occ*(*w, T*) be the number of distinct key phrases in *E* that contain *w* and *qualify* the edge (*E,T*). We define an "edge score" for the edge (*E,T*) represented by *Edge_Score*(*E,T*), which is computed as:

- If *occ*(*w, T*) is 0 for any query keyword then the *Edge_Score*(*E,T*) = 0
- Otherwise, *Edge_Score*(*E,T*)=*Expert_Score*(*E*)*({query keywords w} occ(w, T).

2. We next check for affiliations between expert pages that point to the same target. If two affiliated experts have edges to the same target T, we then discard one of the two edges. Specifically, we discard the edge which has the lower Edge_Score of the two.
3. To compute the Target_Score of a target, we sum the Edge_Scores of all edges incident on it.

Then the list of targets is ranked by Target_Score. Optionally, this list can be filtered by testing if the query keywords are present in the targets. Optionally, we can match the query keywords against each target to compute a Match_Score using content analysis, and combine the Target_Score with the Match_Score before ranking the targets. (Bharat and GA 2001) evaluated this Hilltop algorithm to demonstrate its advantages. Interested readers can refer to this paper.

4.9 A Latent Linkage Information (LLI) Algorithm

The latent linkage information (LLI) algorithm was proposed in (Hou and Zhang 2003), which is for ranking Web pages that are closely related to a given page. The most commonly used methods for this kind of query comes from scientific literature co-citation index algorithm (Garfield 1972; Garfield 1979; Dean and Henzinger 1999; Hou and Zhang 2003). Although co-citation based algorithms are simple and easy to implement, they are unable to reveal the deeper relationships among the pages as they only take into consideration the count of common hyperlinks among pages, page importance is not considered. LLI algorithm tried to reveal deeper relationships among pages for finding more semantically related pages.

The first step of this algorithm is to construct a Web page space (page source) for the given page u from link topology on the Web. Ideally, the page source that is a page set from which the relevant pages are selected should have the following properties:

1. The size of the page source (the number of pages in the page source) is relatively small.
2. The page source is rich in relevant pages.

The *best* relevant pages of the given page, based on the statement in (Dean and Henzinger 1999), should be those that address the same topic as the original page and are semantically relevant to the original one.

For a given page u, its semantic details are most likely to be given by its in-view and out-view (Mukherjea and Hara 1997). The in-view is a set of parent pages of u, and out-view is a set of child pages of u. In other words, the relevant pages with respect to the given page are most likely to be brought into the page source by the in-view and out-view of the given page. The page source for finding relevant pages, therefore, should be derived from the in-view and out-view of the given page, so that the page source is rich in the related pages.

The page source is constructed as a directed graph with edges indicating hyperlinks and nodes representing the following pages:

1. Page u,
2. Up to B parent pages of u, and up to BF child pages of each parent page that are different from u,
3. Up to F child pages of u, and up to FB parent pages of each child page that are different from u.

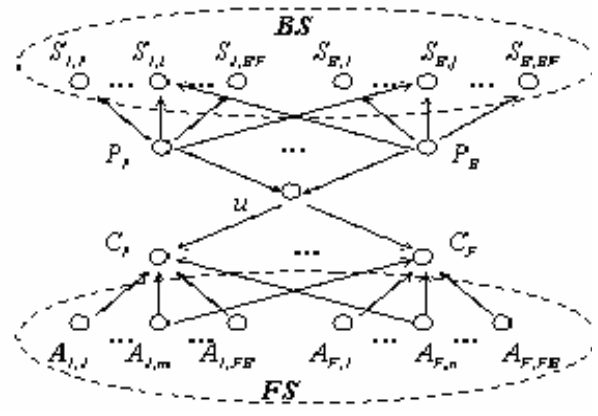


Fig. 4.3. Page source structure for the given page u

The parameters *B*, *F*, *BF* and *FB* are used to keep the page source to a reasonable size. In (Hou and Zhang 2003), $B = FB = 200$, $F = BF = 40$. This page source structure is presented intuitively in Fig. 4.3. Let P_u be a set of parent pages of u, C_u be a set of child pages of u. Page set *BS* composes of children of pages in P_u. The maximum number of children each page in P_u can bring into *BS* is *BF*. Similarly, page set *FS* composes of parents of pages in C_u. The maximum number of parents each page in C_u can bring into *FS* is *FB*.

Suppose the size of *BS* is m (e.g. the number of pages in *BS* is m) and size of P_u is n, the sizes of *FS* and C_u are p and q respectively. Without loss of generality, we also suppose $m > n$ and $p > q$. The topological relationships between the pages in *BS* and P_u are expressed in a linkage matrix A, and the topological relationships between the pages in *FS* and C_u are expressed in another linkage matrix B. The linkage matrices A and B are concretely constructed as follows:

$A = (a_{ij})_{m \times n}$ where

$$a_{ij} = \begin{cases} 1 & \text{when } page\ i \text{ is a child of } page\ j,\ page\ i \in BS,\ page\ j \in P_u, \\ 0 & \text{otherwise} \end{cases}$$

$B = (b_{ij})_{p \times q}$ where

$$b_{ij} = \begin{cases} 1 & \text{when } page\ i \text{ is a parent of } page\ j,\ page\ i \in FS,\ page\ j \in C_u, \\ 0 & \text{otherwise} \end{cases}$$

These two matrices imply more beneath their simple definitions. In fact, the ith row of matrix A can be viewed as the coordinate vector of *page i* (*page* $i \in BS$) in an n-dimensional space spanned by the n pages in P_u, and the ith row of matrix B can be viewed as the coordinate vector of *page i* (*page* $i \in FS$) in a q-dimensional space spanned by the q pages in C_u. Similarly, the jth column of matrix A can be viewed as the coordinate vector of *page j* (*page j* $\in P_u$) in an m-dimensional space spanned by the m pages in *BS*. The meaning is similar for the columns in matrix B. In other words, the topological relationships between pages are transferred, via the matrices A and B, to the relationships between vectors in different multi-dimensional spaces.

Since A and B are real matrices, there exist SVDs of A and B: $A = U_{m \times m} \Sigma_{m \times n} V^T_{n \times n}$, $B = W_{p \times p} \Omega_{p \times q} X^T_{q \times q}$. As indicated above, the rows of matrix A are coordinate vectors of pages of *BS* in an n-dimensional space. Therefore, all the possible inner products of pages in *BS* can be expressed as AA^T, i.e. $(AA^T)_{ij}$ is the inner product of *page i* and *page j* in *BS*. Because of the orthogonal properties of matrices U and V, we have $AA^T = (U\Sigma)(U\Sigma)^T$.

Matrix $U\Sigma$ is also an $m \times n$ matrix. It is obvious from this expression that matrix $U\Sigma$ is equivalent to matrix A, and the rows of matrix $U\Sigma$ could be viewed as coordinate vectors of pages in *BS* in *another* n-dimensional space. Since the SVD of a matrix is not a simple linear transformation of the matrix (Golub and Loan 1993), it reveals statistical regulation of matrix elements to some extent. Accordingly, the coordinate vector transformation from one space to another space via SVD makes sense. For the same reason, the rows of matrix $V\Sigma^T$, which is an $n \times m$ matrix, are coordinate vectors of pages in P_u in *another* m-dimensional space. Similarly, for matrix B, the rows of matrix $W\Omega$ are coordinate vectors of pages in *FS* in another q-dimensional space, and the rows of matrix $X\Omega^T$ are coordinate vectors of pages in C_u in another p-dimensional space.

Next, we discuss matrices A and B separately. For the SVD of matrix A, matrices U and V can be denoted respectively as $U_{m\times m} = [u_1, u_2, ..., u_m]_{m\times m}$ and $V_{n\times n} = [v_1, v_2, ..., v_n]_{n\times n}$, where u_i (i = 1, … , m) is a m-dimensional vector $u_i = (u_{1,i}, u_{2,i}, ..., u_{m,i})^T$ and v_i (i = 1, … , n) is a n-dimensional vector $v_i = (v_{1,i}, v_{2,i}, ..., v_{n,i})^T$. Suppose *rank(A)* = r and singular values of matrix A are as follows:

$$\sigma_1 \geq \sigma_2 \geq ... \geq \sigma_r > \sigma_{r+1} = ... = \sigma_n = 0.$$

For a given threshold ε ($0 < \varepsilon \leq 1$), we choose a parameter k such that $(\sigma_k - \sigma_{k+1}) / \sigma_k \geq \varepsilon$. Then we denote $U_k = [u_1, u_2, ..., u_k]_{m\times k}$, $V_k = [v_1, v_2, ..., v_k]_{n\times k}$, $\Sigma_k = diag(\sigma_1, \sigma_2, ..., \sigma_k)$, and $A_k = U_k \Sigma_k V_k^T$.

From the SVD theorem in Chap. 2, the best approximation matrix A_k contains main linkage information among the pages and makes it possible to filter those irrelevant pages, which usually have fewer links to the parents of given u, and effectively find relevant pages. In this algorithm, the relevance of a page to the given page u is measured by the similarity between them. For measuring the page similarity based on A_k, we choose the ith row R_i of the matrix $U_k\Sigma_k$ as the coordinate vector of *page* i (*page* $i \in$ *BS*) in a k-dimensional subspace S:

$$R_i = (u_{i1}\sigma_1, u_{i2}\sigma_2, ..., u_{ik}\sigma_k), \quad i = 1, 2, ..., m. \tag{4.7}$$

For the given page u, since it is linked by every parent page, it is represented as a coordinate vector with respect to the pages in P_u: $u = (g_1, g_2, ..., g_n)$ where $g_i = 1$, $i \in [1, n]$. The projection of coordinate vector u in the k-dimensional subspace S is represented as

$$u' = uV_k\Sigma_k = (g_1', g_2', ..., g_k') \ , \tag{4.8}$$

where $g_i' = \sum_{t=1}^{n} g_t v_{ti}\sigma_i$, $i = 1, 2, ..., k$.

The equations (7) and (8) map the pages in *BS* and the given page *u* into the vectors in the same *k*-dimensional subspace *S*, in which it is possible to measure the similarity (relevance degree) between a page in *BS* and the given page *u*. The commonly used cosine similarity measurement is used for this purpose, i.e. for two vectors $x = (x_1, x_2, ..., x_k)$ and $y = (y_1, y_2, ..., y_k)$ in a *k*-dimensional space, the similarity between them is defined as

$$sim(x, y) = \frac{| x \cdot y |}{\| x \|_2 \| y \|_2}, \text{ where } x \cdot y = \sum_{i=1}^{k} x_i y_i \ , \ \| x \|_2 = \sqrt{x \cdot x} \ .$$

In this way, the similarity between a *page i* in *BS* and the given page *u* is defined as

$$BSS_i = sim(R_i, u') = \frac{| R_i \cdot u' |}{\| R_i \|_2 \| u' \|_2}, \ i = 1, 2, ..., m. \tag{4.9}$$

For the given selection threshold δ, the relevant pages in *BS* with respect to the given page *u* is the set

$$BSR = \{ p_i \mid BSS_i \geq \delta \ , p_i \in BS, i = 1, 2, ..., m \}.$$

In the same way, for the SVD of matrix $B = W_{p\times p}\Omega_{p\times q}X^T_{q\times q}$, we suppose *rank(B)* = *t* and singular values of matrix *B* are $\omega_1 \geq \omega_2 \geq ... \geq \omega_t > \omega_{t+1} = ... = \omega_q = 0$. For a given threshold ε ($0 < \varepsilon \leq 1$), we choose a parameter *l* such that $(\omega_l - \omega_{l+1}) / \omega_l \geq \varepsilon$. Then we denote $B_l = W_l\Omega_l X_l^T$, where

$$W_l = [w_{i,j}]_{p\times l}, \ X_l = [x_{i,j}]_{q\times l}, \ \Omega_l = diag(\omega_1, \omega_2, ..., \omega_l) \ .$$

The *i*th row R'_i of the matrix $W_l\Omega_l$ is the coordinate vector of *page i* (*page i* $\in$ *FS*) in a *l*-dimensional subspace *L*:

$$R_i' = (w_{i1}\omega_1, w_{i2}\omega_2, ..., w_{il}\omega_l), \quad i = 1, 2, ..., p. \tag{4.10}$$

The projection of coordinate vector *u* in the *l*-dimensional subspace *L* is represented as

$$u'' = uX_l\Omega_l = (g_1'', g_2'', ..., g_l'') \ , \tag{4.11}$$

where

$$g_i'' = \sum_{j=1}^{q} g_j x_{ji} \omega_i , \quad i = 1, 2, ..., l.$$

Therefore, the similarity between a *page i* in *FS* and the given page *u* is

$$FSS_i = sim(R_i', u'') = \frac{| R_i' \cdot u'' |}{\| R_i' \|_2 \| u'' \|_2}, \quad i = 1, 2, ..., p. \tag{4.12}$$

For the given selection threshold δ, the relevant pages in *FS* with respect to the given page *u* is the set

$$FSR = \{ p_i \mid FSS_i \geq \delta \ , p_i \in FS, i = 1, 2, ..., p \}.$$

Finally, the relevant pages of the given page (URL) *u* are a page set $RP = BSR \cup FSR$.

As indicated in (Hou and Zhang 2003), the complexity or computational cost of the LLI is dominated by the SVD computation of the linkage matrices *A* and *B*. Without loss of generality, we suppose $m = max(m, p)$ and $n = max(n, q)$. Then the complexity of the LLI algorithm is $O(m^2 n + n^3)$ (Golub and Loan 1993). If $n << m$, the complexity is approximately $O(m^2 n)$. Since the number of pages in the page source can be controlled by the algorithm, and this number is relatively very small compared with the number of pages on the Web, the LLI algorithm is feasible for application. For experimental results of this algorithm, please refer to (Hou and Zhang 2003).

Up to now, the HITS and PageRank algorithms we discussed, as well as their variants, are all based on Web page hyperlink analysis although they focus on different goals (i.e. globally rank Web pages and locally rank pages). It is clear from these discussions that hyperlinks among Web pages convey semantic information that can be used to reveal deeper relationships among pages. It is also clear from these discussions that much effort has been made trying to unify hyperlink analysis within a uniform model. Another example of this effort can be found in (Ding et al. 2002), which used matrices to model hyperlink analysis in a general form. From this general form, HITS and PageRank can be derived as two extreme end cases of this form. For more details of this general form, please refer to (Ding et al. 2002) and other related papers. It can be seen that whatever efforts are being made, hyperlink analysis could incorporate with other Web page information analysis techniques, such as page content and Web access log file analysis, to improve various Web based applications.

5 Affinity and Co-Citation Analysis Approaches

In this chapter, we present several Web community analysis approaches, such as affinity, co-citation etc, for capturing underlying relationships among Web pages. In Sect. 5.1, we start presenting a new Web page similarity measurement, which incorporates hyperlink transitivity and page importance. Then, Sect. 5.2 gives a hierarchical clustering algorithm based on page correlation matrix. In Sect. 5.3, we adapt a concept of affinity and utilize it to represent the relationship between two pages. The permutation of affinity matrix is utilized to achieve highest global affinity. Moreover, the affinity-based clustering algorithms are given in this section. The Co-Citation algorithm is discussed in detail in Sect. 5.4 and an extended algorithm is described as well.

5.1 Web Page Similarity Measurement

A Web page similarity usually refers to a certain page space. Since we are concerned about clustering Web-searched results in this work, we focus on a page space that is related to the user's query topics. The ideas and analysis techniques in the following sections, however, could also be used to other concerned Web spaces, such as those in (Weiss et al. 1996) and (Pitkow and Pirolli 1997). In this section, we firstly establish a page source (space) that is related to the query topics. Within this page source, we incorporate hyperlink transitivity and page importance to propose a new page similarity measurement.

5.1.1 Page Source Construction

The page source construction is based on the Web-searched results. For users, they are usually concerned about a part of searched results; say the first r highest-ranked pages returned by the search engine. From the hyperlink analysis point of view, the pages that link to or are linked to these r highest-ranked pages are also related to the query topics to some extent. Therefore, the page source S with respect to user's query topics is constructed as follow:

Step 1: Select r highest-ranked pages from the searched results to form a root page set *R*.
Step 2: For each page *p* in *R*, select up to *B* pages, which point to *p* and whose domain names are different from that of *p*, and add them to the back vicinity set *BV* of *R*.
Step 3: For each page *p* in *R*, select up to *F* pages, which are pointed to by *p* and whose domain names are different from that of *p*, and add them to the forward vicinity set *FV* of *R*.
Step 4: Page source S is constructed by uniting sets *R*, *BV*, *FV* and adding original links between pages in *S*.

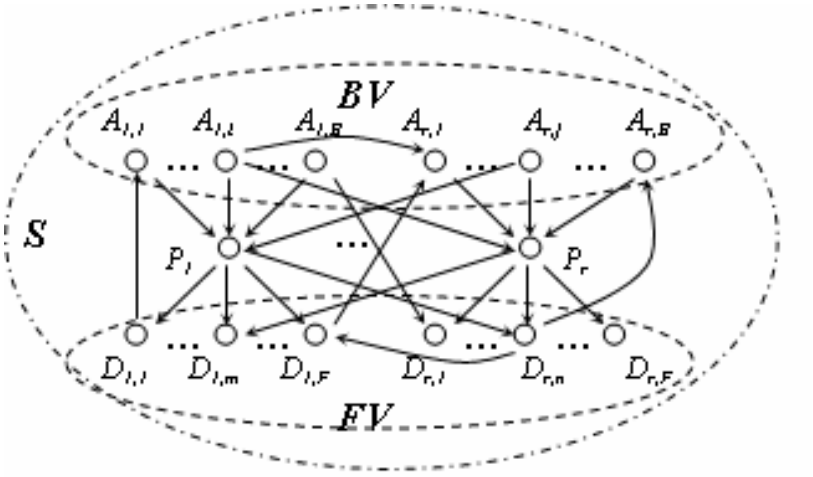


Fig. 5.1. Structure of the page source *S*

In the above page source construction algorithm, parameters *B* and *F* are used to guarantee that the page source *S* is of a reasonable size. For example, we choose the value 200 for *B* and *F* from our experiment experience. When constructing sets *BV* and *FV*, it is required that for each page *p* in *R*, the domain names of its parent pages and child pages are different from the domain name of the page *p*. This requirement filters those parent and child pages coming from the same Website where the page *p* is located. The reason, as indicated in (Bharat and Henzinger 1998; Wang and Kitsuregawa 2001), is that the links within the same Website are more likely to reveal the inner structure than to imply a certain semantic relationship.

During the page source construction procedure, it is possible to bring some mirror pages into the page source. There are several reasons for not being required to remove these mirror pages. Firstly, there is no standard currently to identify whether two pages are mirror pages or not just from their linkage

analysis, and identifying mirror pages will add extra computing cost. Secondly, if two pages are mirror pages, they have the same hyperlink structure and are most likely to be clustered into one cluster, in which the user or an algorithm can identify them easily. Therefore, keeping a proper mirror page redundancy in the page source *S* is reasonable.

It is worth indicating that the Web pages and their linkage information required for page source construction could be obtained in many ways. For example, the child pages of a certain page and their links can be directly obtained from that page, while the parent pages of that page and their links can be found by the functions provided by some Web browsers, such as the search function *link:URL* provided by *AltaVista* and *Google*. Bharat et al (Bharat et al. 1998) proposed a specific system to obtain linkage information from the Web. Usually, Web search engines use crawlers (spiders) to obtain the Web pages with hyperlink information, and the obtained information is stored in a specific database for further use (Brin and L. Page 1998).

5.1.2 Page Weight Definition

The role each page plays in similarity measurement is different in a concerned page source *S*. For instance, two kinds of pages need to be noticed. The first one is the page whose *out-link contribution* to *S* (i.e. the number of pages in *S* that are pointed to by this page) is greater than the average out-link contribution of all the pages in the page source *S*. Another kind is the page whose *in-link contribution* to *S* (i.e. the number of pages in *S* that point to this page) is greater than the average in-link contribution of all the pages in the page source *S*. The pages of the first kind are called *index* pages in (Botafogo and Shneiderman 1991) (*hub* pages in (Kleinberg 1999)), and those of the second kind are called *reference* pages in (Botafogo and Shneiderman 1991) (*authority* pages in(Kleinberg 1999)). These pages are most likely to reflect certain topics related to the query within the concerned page source. If two pages are linked by or linking to some pages of these kinds, these two pages are more likely to be located in the same topic group and have higher similarity.

It also needs to be noticed that index Web pages in common sense, such as personal bookmark pages and index pages on some special-purpose Web sites, might not be the index pages in the concerned page source *S* if their out-link contribution to *S* is below the average out-link contribution in *S*. For the same reason, some pages with high in-degrees on the Web, such as home pages of commonly used search engines, might not be the reference pages in the concerned page source *S*. For simplicity, we filter the home pages of commonly used search engines (e.g. *Yahoo!*, *AltaVista*, *Google* and *Excite*) from the concerned page source *S*, since these pages are not related to any

specific topics. To measure the importance of each page *within the concerned page source*, we define a weight for each page.

For each page P_i in the page source S, similar to the HITS algorithm in (Kleinberg 1999), we associate a non-negative *in-weight* $P_{i,in}$ and a non-negative *out-weight* $P_{i,out}$ with it. Due to the hyperlink transitivity in the page source, the *in-weight* and *out-weight* for the page P_i in S are iteratively calculated as follows (Kleinberg 1999):

$$P_{i,in} = \sum_{P_j \in S, P_j \to P_i} P_{j,out} ,$$

$$P_{i,out} = \sum_{P_j \in S, P_i \to P_j} P_{j,in} .$$

In order to guarantee the convergence of the above iterative operations, it is required that the in-weight vector and out-weight vector are normalized after each iteration, i.e.

$$\sum_{P_i \in S} P_{i,in}^2 = 1 , \sum_{P_i \in S} P_{i,out}^2 = 1 .$$

We denote the average in-weight of S as μ, and the average out-degree of S as λ. That is

$$\mu = \sum_{P_i \in S} P_{i,in} / size(S) , \ \lambda = \sum_{P_i \in S} P_{i,out} / size(S) ,$$

where $size(S)$ is the number of pages in S. Then the page weight for P_i is defined as

$$w_i = 1 + \max((P_{i,in} - \mu)/(M_{in} - m_{in}), (P_{i,out} - \lambda)/(M_{out} - m_{out})) \tag{5.1}$$

where M_{in}, m_{in}, M_{out} and m_{out} are defined as follow:

$$M_{in} = \max_{P_j \in S}(P_{j,in}) , \quad m_{in} = \min_{P_j \in S}(P_{j,in}) ,$$

$$M_{out} = \max_{P_j \in S}(P_{j,out}) , \ m_{out} = \min_{P_j \in S}(P_{j,out}) .$$

The page weight definition in (1) indicates that if a page's in-weight and out-weight in S are below their corresponding average values μ and λ, its weight will be less than 1, which means its influence to the similarity measurement is relatively less. For the same reason, if a page's in-weight or out-weight in S is above the average value (e.g. an index page or a reference page), its weight will be greater than 1 and its influence to the similarity measurement is relatively greater. In other words, the page weight defined in (1) reflects the importance of each page's role in the concerned page source. This page importance will be incorporated in the page similarity measurement.

5.1.3 Page Correlation Matrix

For each Web page, its correlation with other pages, via linkages, is expressed in two ways: one is *out-links* from it, another is *in-links* to it. In this work, the similarity between two pages is measured by their own correlations with other pages in the page source S, rather than being derived directly from the links between them. For measuring the page correlation, we firstly give the following definitions.

Definition 1. If page A has a direct link to page B, then the length of path from page A to page B is 1, denoted as $l(A,B) = 1$. If page A has a link to page B via n other pages, then $l(A,B) = n+1$. The distance from page A to page B, denoted as $sl(A,B)$, is the shortest path length from A to B, i.e. $sl(A,B) = \min(l(A,B))$. The length of path from a page to itself is zero, i.e. $l(A,A) = 0$. If there are no links from page A to page B (direct or indirect), then $l(A,B) = \infty$.

It can be inferred from this definition that $l(A,B) = \infty$ does not imply $l(B,A) = \infty$, because there might still exist links from page B to page A in this case.

Definition 2. The correlation weight between two pages i and j ($i \neq j$), denoted as $w_{i,j}$, is the maximal weight of their weights, i.e. $w_{i,j} = \max(w_i,w_j)$ where w_i and w_j are the page weights for pages i and j respectively. If $i = j$, $w_{i,j}$ is defined as 1.

The following definition defines how much two pages correlate with each other if there is a direct link between them.

Definition 3. Correlation factor, denoted as F, $0<F<1$, is a constant that measures the correlation rate between two page with direct link, i.e. if page A has a direct link to page B, then the correlation rate from page A to page B is F.

How to determine the value of this correlation factor F to more precisely reflect the correlation relationship between pages is beyond the scope of this work. Further research could be done in this area. In this work, similar to the work in (Weiss et al. 1996), the value of F is chosen as 1/2. That means if page A has a direct link to page B, the correlation from page A to page B is 50%. It is argued that not every pair of pages that are hyperlinked has 50% semantic relationship with each other. However, in the context of the Web, the research focuses on finding certain statistical regularities from a large number of pages. Therefore, certain imprecise relationship descriptions are permitted as in (Weiss et al. 1996). For general purpose, we still use F in the following algorithm to represent this correlation factor. It can be seen that hyperlinks between pages are used to measure the correlations between pages, rather than to directly measure the similarities.

With the above definitions, a correlation degree between any two pages can be defined. This correlation degree depends on the value of correlation factor F, the distance between the two pages (the farther the distance, the less the correlation degree), and the correlation weights of involved pages along the shortest path. The following definition gives this function.

Definition 4. The *correlation degree* from page i to page j, denoted as c_{ij}, is defined as

$$c_{ij} = w_{i,k1} w_{k1,k2} \cdots w_{kn,j} F^{sl(i,j)}, \tag{5.2}$$

where F are correlation factor, $sl(i,j)$ is the distance from page i to page j, and $w_{i,k1}$, $w_{k1,k2}$, …, $w_{kn,j}$ are correlation weights of the pages i, $k1$, $k2$, …, kn, j that form the distance $sl(i,j)$, i.e. $i \to k1 \to k2 \to \ldots \to kn \to j$. If $i = j$, then c_{ij} is defined as 1.

For the concerned page source S, we suppose the size of the root set R is m, the size of the vicinity set $V = BV \cup FV$ is n. Then the correlation degrees of all the pages in S can be expressed in a $(m+n)\times(m+n)$ matrix $C = (c_{ij})_{(m+n)\times(m+n)}$, called *correlation matrix*. This correlation matrix C is a numerical format that converts the hyperlinks (direct or indirect) between pages in S into the correlation degrees, incorporating the hyperlink transitivity and page importance.

The key for computing the correlation degree c_{ij} in (2) is the distance $sl(i,j)$ between any two pages i and j in S. This distance can be computed via some operations on the matrix elements of a special matrix called *primary correlation matrix*. The primary correlation matrix $A = (a_{ij})_{(m+n)\times(m+n)}$ is constructed as follows:

$$a_{ij} = \begin{cases} F & \text{if there is a direct link from } i \text{ to } j, i \neq j \\ 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

Based on this primary correlation matrix, the algorithm for computing the distance $sl(i,j)$ between any two pages i and j is described as follows:

Step 1: For each page $i \in S$, choose $factor = F$ and go to Step 2;
Step 2: For each element a_{ij}, if $a_{ij} = factor$, then set $k = 1$ and go to Step 3. If there is no element a_{ij} ($j = 1, \ldots, m+n$) such that $a_{ij} = factor$, then go back to Step 1;
Step 3: If $a_{jk} \neq 0$ and $a_{jk} \neq 1$, calculate $factor*a_{jk}$;
Step 4: If $factor*a_{jk} > a_{ik}$, then replace a_{ik} with $factor*a_{jk}$, change $k = k+1$ and go back to Step 3. Otherwise, change $k = k+1$ and go back to Step 3;
Step 5: Change $factor = factor*F$ and go to Step 2 until there are no changes to all element values a_{ij} ;

Step 6: Go back to Step 1 until all the pages in S have been considered.

After element values of matrix A are updated by the above algorithm, the distance from page i to page j is

$$sl(i, j) = [\log a_{ij} / \log F] .$$

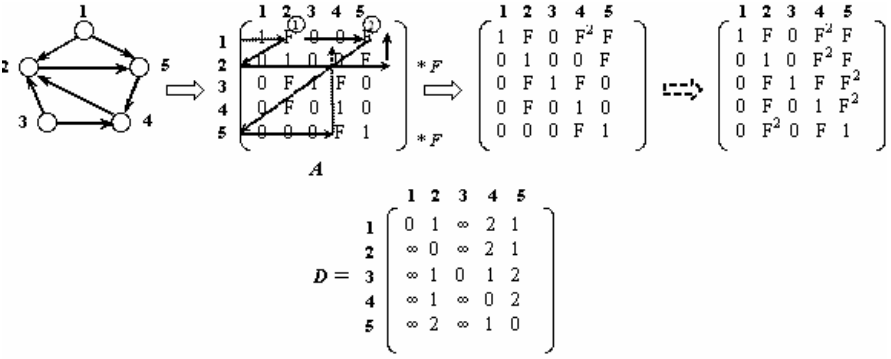


Fig. 5.2. Example of computing distance between pages

The example in Fig. 5.2 gives an intuitive execution demonstration of the above algorithm. In this example, five pages (numbered 1 to 5) and their linkages are represented as a directed graph. Their primary correlation matrix A is also shown in the figure. The dashed arrows in matrix A show the first level operation sequence (*factor* = F) of the above algorithm for page 1. The procedure of other level operations for other pages is similar except for changing the values of variable *factor* according to the above algorithm. The final updated primary correlation matrix and the corresponding distance matrix D are presented in the figure. It is clear from these results that although there are several paths from page 1 to page 4, the distance from page 1 to page 4 is 2, which is consistent with the real situation. The situation is the same for page 3 and page 5 in this example.

This distance computation algorithm could be adapted for computing the correlation degrees (5.2). The above algorithm also provides a numerical method to find the shortest path between any two nodes in a directed graph. If page correlation weights are not considered in computing the correlation degrees c_{ij}, the above algorithm could be directly used to produce correlation matrix C.

5.1.4 Page Similarity

In this work, we focus on clustering Web-searched pages in the root set R with a new page similarity measurement. The new page similarity is measured by the page correlation degrees within the concerned page source. For simplicity and better understanding of this new similarity, we divide the correlation matrix C into four blocks (sub-matrices) as follows:

$$C = (c_{ij})_{(m+n)\times(m+n)} = \begin{array}{c} \\ \mathrm{R} \\ \mathrm{V} \end{array} \begin{array}{c} \begin{array}{cc} \mathrm{R} & \mathrm{V} \end{array} \\ \begin{bmatrix} \textcircled{1} & \textcircled{2} \\ \textcircled{3} & \textcircled{4} \end{bmatrix}_{(m+n)\times(m+n)} \end{array}$$

The elements in sub-matrix 1 represent the correlation relationships between the pages in R. Similarly, the elements in sub-matrices 2 and 3 represent the correlation relationships between the pages in R and V, and sub-matrix 4 gives the correlation relationships between the pages in V. It can be seen that the correlation degrees related with the pages in R are located in three sub-matrices 1, 2 and 3. Therefore, the similarity measurement for the pages in R only refers to the elements in these three sub-matrices.

Note: If the similarity between any two pages in the whole source space S is to be measured, the whole correlation matrix C will be used and the similarity definition is the same as follows.

In the correlation matrix C, the row vector that corresponds to each page i in R is in the form of

$$row_i = (c_{i,1}, c_{i,2}, ..., c_{i,m+n}), \quad i = 1,2,...,m .$$

From the construction of matrix C, it is known that row_i represents *out-link* relationship of page i in R with all the pages in S, and element values in this row vector indicate the correlation degrees of this page to the linked pages. Similarly, the column vector that is in the form of

$$col_i = (c_{1,i}, c_{2,i}, ..., c_{m+n,i}), \quad i = 1,2,...,m ,$$

represents *in-link* relationship of page i in R with all the pages in S, and its element values indicate the correlation degrees from the pages in S to page i.

Each page i in R, therefore, is represented as two correlation vectors: row_i and col_i. For any two pages i and j in R, their *out-link similarity* is defined as

$$sim_{i,j}^{out} = \frac{(row_i, row_j)}{\| row_i \| \cdot \| row_j \|},$$

where

$$(row_i, row_j) = \sum_{k=1}^{m+n} c_{i,k} c_{j,k}, \qquad \| row_i \| = (\sum_{k=1}^{m+n} c_{i,k}^2)^{1/2}.$$

Similarly, their *in-link similarity* is defined as

$$sim_{i,j}^{in} = \frac{(col_i, col_j)}{\| col_i \| \cdot \| col_j \|}.$$

Then the similarity between any two pages i and j in R is defined as

$$sim(i, j) = \alpha_{ij} \cdot sim_{i,j}^{out} + \beta_{ij} \cdot sim_{i.j}^{in}, \tag{5.3}$$

where α_{ij} and β_{ij} are the weights for out-link and in-link similarities respectively.

The similarity weights α_{ij} and β_{ij} are determined dynamically as:

$$\alpha_{ij} = \frac{\| row_i \| + \| row_j \|}{MOD_{ij}}, \quad \beta_{ij} = \frac{\| col_i \| + \| col_j \|}{MOD_{ij}},$$

where $MOD_{ij} = \| row_i \| + \| row_j \| + \| col_i \| + \| col_j \|$. As a special case, for any pair of pages *i* and *j* in *R*, if their out-link modes $\|row\|$ and in-link modes $\|col\|$ are approximately the same, the weights α_{ij} and β_{ij} could be simply chosen as $(\alpha_{ij}, \beta_{ij}) = (1/2, 1/2)$.

It is argued that hyperlink transitivity would bring noise factors into the page similarity measurement. One source of the noise factors is the noise pages that are not query topic related but are densely linked with each other in the page source. The noise pages will have unreasonably high page weights and mislead the page correlation degrees. These noise pages, however, can be eliminated from the page source by many existing algorithms, such as (Bharat and Henzinger 1998; Hou and Zhang 2002; Hou et al. 2002). Therefore, in this work, we can reasonably assume that the pages in the page source are query topic related. Another noise factor source is taking every path between two pages into consideration in the page correlation degree measurement. Under this situation, minor page correlations between two pages could be accumulated such that the final correlation degrees are unreasonably increased in some cases. In this work, however, the page correlation degree only takes the shortest path between two pages into account, so the noise factors are omitted. On the other hand, the page correlation degree decreases quickly with the increase of the shortest path length (distance). Therefore, the contribution of hyperlink transitivity to the page correlation degree is minor if there exists a long distance between two pages. This would also eliminate many noise factors in the page similarity measurement.

The above page similarity measurement is derived from the page correlation degrees, rather than the direct hyperlinks between the pages. It seems that

this idea comes from the co-citation analysis. The intension of this new similarity, however, is different from that of co-citation analysis based similarities. In the co-citation analysis, the influence of each page to the similarity measurement is the same, and the similarity between any two pages only depends on the number of direct common pages (common parent and child pages). In this new similarity measurement (5.3), the influence of each page to the similarity is different, which is reflected by the page weight. Furthermore, this new similarity not only depends on the number of direct common pages, but also depends on the number of indirect common pages and the correlation degrees of the involved pages. Fig. 5.3 gives an example that shows these intrinsic differences.

In this example, the values for the weights α_{ij} and β_{ij} are simply chosen as $(\alpha_{ij}, \beta_{ij}) = (1/2, 1/2)$. The number in a pair of parentheses beside a page number is the weight of that page, and the number beside a link arrow indicates the correlation degree between the two pages if the correlation factor $F = 1/2$. For the situation (a) in this example, if the co-citation analysis is applied, the similarity of pages 1 and 2 is the same as that of pages 2 and 3. But, if the similarity measurement (5.3) is applied to this situation, we get $sim(1,2) = 0.09$ and $sim(2,3) = 0.17$. The similarity $sim(2,3)$ is greater than $sim(1,2)$ because the common page 5 of pages 2 and 3 are more important than common page 4 of pages 1 and 2. The simple co-citation analysis, however, is unable to reflect this difference.

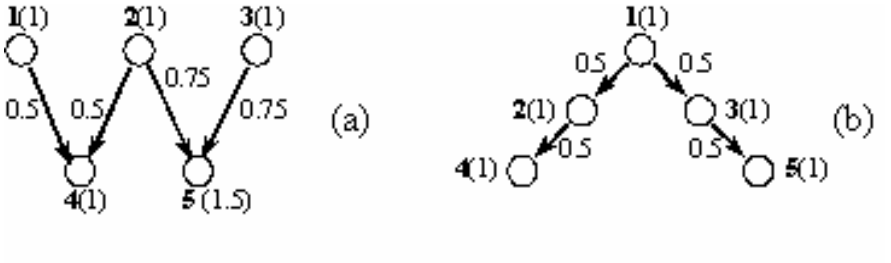


Fig. 5.3. Example of the similarity measurement

For the situation (b), the similarity between pages 4 and 5 is zero if the co-citation analysis is applied, because they have no direct common (parent) pages. Actually, there still exists a relative weak relationship between them via page 1, and their similarity should not be zero. By applying (5.3) to this situation, we get $sim(4,5) = 0.02$, which reflects the influence of the indirect common pages to the page similarity measurement. If pages 2 and 3 have higher page weights, $sim(4,5)$ would be higher.

5.2 Hierarchical Web Page Clustering

With the page similarity measurement and the correlation matrix C, a hierarchical Web page clustering algorithm could be established. This hierarchical clustering algorithm consists of two phases. The first one is single layer clustering, in which the pages in R are clustered at the same level without hierarchy. The second phrase is hierarchical clustering, in which the pages in the clusters produced by the first phase are clustered further to form a cluster hierarchical structure. Fig. 5.4 gives this hierarchical clustering diagram.

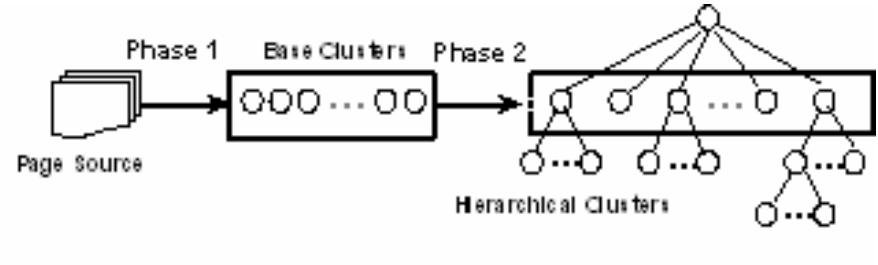


Fig. 5.4. Hierarchical clustering diagram

The details of the hierarchical clustering algorithm are described as follows.

Phase 1: Single Layer Clustering
[**Input**]: A set of Web pages $R = \{p_1, p_2, \ldots, p_m\}$, clustering threshold T.
[**Output**]: A set of clusters $CL = \{CL_i\}$.
[**Algorithm**]: *BaseCluster*(R, T)
Step 1. Select the first page p_1 as the initial cluster CL_1 and the centroid of this cluster, i.e. $CL_1 = \{p_1\}$ and $CE_1 = p_1$.
Step 2: For each page $p_i \in R$, calculate the similarity between p_i and the centroid of each existing cluster $sim(p_i, CE_j)$.
Step 3: If $sim(p_i, CE_k) = \max_j(sim(p_i, CE_j)) > T$, then add p_i to the cluster CL_k and recalculate the centroid CE_k of this cluster that consists of two vectors

$$CE_k^{row} = \frac{1}{|CL_k|}\sum_{j \in CL_k} row_j \,, CE_k^{col} = \frac{1}{|CL_k|}\sum_{j \in CL_k} col_j \,,$$

where $|CL_k|$ is the number of pages in CL_k. Otherwise, p_i itself initiates a new cluster and is the centroid of this new cluster.

Step 4: If there are still pages to be clustered (i.e. pages that have not been clustered or a page that itself is a cluster), go back to Step 2 until all cluster centroids no longer change.
Step 5: Return clusters $CL = \{CL_i\}$.

The above phase 1 of the clustering algorithm produces a set of single layer clusters called *base clusters*. Recursively applying the above algorithm, with increasing clustering threshold T, to each base cluster would produce downward hierarchical clusters. This procedure is stopped when the number of pages in each leaf cluster is below a certain predefined threshold NP. Then the whole hierarchical cluster structure is produced. The procedure is described as phase 2 of the clustering algorithm.

Phase 2: Hierarchical Clustering
[**Input**]: A set of base clusters $CL = \{CL_i\}$, parameter NP and clustering threshold T in phase 1.
[**Output**]: Hierarchical clusters $HCL = \{HCL_i\}$.
[**Algorithm**]: *HierarchyCluster*(CL, NP, T)
Step 1: Set $HCL = CL$, and let CL to be the set of clusters at layer 1 (base layer), i.e. $CL^1 = \{CL_i^1\} = \{CL_i\}$. Assign $l = 1$ and $T' = T$.
Step 2: Recursively increase T', l and call algorithm *BaseCluster*(CL_i^l, T') for those clusters CL_i^l in CL^l that contain more than NP pages. Add the clusters at each layer to HCL.
Step 3: Return the produced set of hierarchical clusters HCL.

The clustering threshold T in the algorithm is determined by practical requirement. It should guarantee that the pages are clustered into a reasonable number of clusters. For example, T could be chosen as the average page similarity of all the pages in R. The increase rate for the hierarchical clustering threshold T' could be chosen as a certain percentage of the threshold T.

The parameter NP (e.g. 10) is used to control the number of downward levels of the hierarchical cluster structure. If the number of pages in a cluster $\leq NP$, this cluster should not be divided into some smaller clusters (at a lower level) any more. If the hierarchical cluster structure is for Web page navigation, the value of NP is usually determined by the number of pages in a cluster that users can tolerate for navigation. Proper NP value would also be able to reduce the execution cost of the algorithm.

It can be inferred from the phase 1 of the algorithm that a page in R only belongs to a cluster. In practice, a page might belong to multiple clusters. This requirement can be easily met by only changing the clustering condition in the step 3 of the phase 1, i.e. changing the condition " If $sim(p_i, CE_k) =$

$\max_j(sim(p_i, CE_j)) > T$" to " If $sim(p_i, CE_k) > T$". For computation simplicity, we still assume that a page only belongs to a cluster.

As stated in (Wen et al. 2001), for this kind of hierarchical clustering algorithm, it has been proven (Wang 1997) that the algorithm is independent of the order in which the pages are presented to the algorithm if the pages are properly normalized. Since the page normalization is guaranteed in the similarity measurement (3), the above hierarchical clustering algorithm is independent of the page order. It is not difficult to prove that the complexity of this algorithm is $O(M*N*logN)$, where M is the number of generated clusters and N is the number of pages to be clustered.

5.3 Matrix-Based Clustering Algorithms

In this work, we still focus on the page source constructed in Sect. 5.1.1, and adopt the concepts and symbols in that section. For the concerned page source S, we suppose the size of the root set R is m, the size of the vicinity set $V = BV \cup FV$ is n. With the new page similarity measurement (3), a new $m \times m$ symmetric matrix SM, called *similarity matrix* for R, can be constructed as $SM = (sm_{i,j})_{m \times m}$ for all the pages in the root set R, where

$$sm_{i,j} = \begin{cases} sim(i, j) & \text{if } i \neq j, \\ 1 & \text{if } i = j. \end{cases}$$

The matrix-based Web page clustering is implemented by partitioning the page similarity matrix. With the partition of the similarity matrix, the pages are accordingly clustered into clusters. To guarantee the effectiveness of matrix-based algorithms in page clustering, it is needed to conduct similarity matrix permutation before partition.

5.3.1 Similarity Matrix Permutation

The similarity matrix permutation is to put those closely related pages together in the similarity matrix SM, such that the page position in the matrix more reasonably reflects the relevance between pages within the whole range of concerned pages. For measuring how close two pages are related, we define the *affinity* of two pages i and $j \in R$ as:

$$AF(i, j) = \sum_{k=1}^{m} sm_{i,k} \times sm_{j,k}$$

The corresponding affinity matrix is denoted as *AF*. Two pages with higher affinity would be more related with each other and should have more of a chance to be put in the same cluster. However, since the pages in the matrix have mutual effects, the final page positions of the similarity matrix should be determined within the whole range of concerned pages in the matrix. For globally optimising the page position, we define the *global affinity* of matrix *SM* as

$$GA(SM) = \sum_{i=1}^{m} \sum_{j=1}^{m} AF(i, j)[AF(i, j-1) + AF(i, j+1)] \quad (5.4)$$

where $AF(i,0) = AF(i, m+1) = 0$. $GA(SM)$ contains all the affinities of pages in *R* with their neighbouring pages. The higher the $GA(SM)$, the more likely the closely related pages are put together as neighbouring pages. The purpose of the similarity matrix permutation is to get the highest $GA(SM)$, under which the close related pages are located closely to each other in the matrix.

The highest $GA(SM)$ can be obtained by swapping the positions of every pair of columns (accordingly rows) in matrix *AF*. In fact, we denote the permuted affinity matrix as *PA*. Similar to the work in (Özsu and Valduriez 1991), the algorithm for generating *PA* with the highest $GA(SM)$ consists of three steps:

1. **Initiation**. Place and fix one of the columns of *AF* arbitrarily into *PA*.
2. **Iteration**. Pick each of the remaining *m-i* columns (where *i* is the number of columns already placed in *PA*) and try to place them in the remaining *i*+1 positions in the *PA*. Choose the placement that makes the greatest contribution to the global affinity. Continue this step until no more columns remain to be placed.
3. **Row ordering**. Once the column ordering is determined, the placement of the rows should also be changed so that their relative positions match the relative positions of the columns.

When the highest $GA(SM)$ is achieved, the page positions in *SM* are permuted according to the actual page positions in the permuted affinity matrix *PA*. As a result, the closely related pages are located closely to each other in the new permuted similarity matrix. For simplicity, hereafter, we still denote this permuted similarity matrix as *SM*.

Fig. 5.5 gives an example of the similarity matrix permutation. There are nine pages (marked P_1, P_2, … , P_9) in this example. The original and permuted similarity matrices are shown in Fig. 5.5(a) and (b) separately. It can be seen that the closely related pages are located closely to each other in the permuted similarity matrix (b) with the highest global affinity.

5.3.2 Clustering Algorithm from a Matrix Partition

Matrix-based page clustering is implemented by decomposing the permuted matrix *SM* into four sub-matrices along its main diagonal, i.e.

$$SM = (sm_{i,j})_{m\times m} = \begin{pmatrix} SM_{1,1} & SM_{1,2} \\ SM_{2,1} & SM_{2,2} \end{pmatrix}_{m\times m}.$$

Since the rows (or columns) of the permuted similarity matrix *SM* correspond to the pages to be clustered, the pages corresponding to the sub-matrices $SM_{1,1}$ and $SM_{2,2}$ form two clusters, while the elements of sub-matrix $SM_{1,2}$ (or $SM_{2,1}$, since $SM^T_{2,1} = SM_{1,2}$) represent similarities between the pages that separately belong to these two clusters.

It is clear that the partition of matrix *SM* is equivalent to finding a dividing point *D* along the main diagonal of *SM*. To find this dividing point *D*, we define a measurement for the sub-matrix $SM_{p,q}$ ($1\le p, q \le 2$) as

$$M(SM_{p.q}) = \sum_{i=(p-1)*d+1}^{d+(m-d)*(p-1)} \sum_{j=(q-1)*d+1}^{d+(m-d)*(q-1)} sm_{i,j}\ ,\ 1\le p, q \le 2,$$

where *d* stands for the row (and column) number of *D*. The dividing point *D* is selected such that the following function is maximized

$$.F_D = M(SM_{1,1})* M(SM_{2,2}) - M(SM_{1,2})* M(SM_{2,1}) \qquad (5.5)$$

Therefore, the determination of the dividing point *D* makes the pages with high affinity to be located in the same cluster (sub-matrix), and the similarity between the clusters to be low. Once the dividing point *D* is determined, two clusters $SM_{1,1}$ and $SM_{2,2}$ are settled down. For instance, the pages in the example of Fig. 5.5 are clustered into two clusters: $SM_{1,1} = \{P_1, P_5, P_7, P_2\}$, $SM_{2,2} = \{P_4, P_9, P_3, P_8, P_6\}$, while the row (an column) number of *D* is 4.

This matrix partition could be recursively applied to the matrices $SM_{1,1}$ and $SM_{2,2}$ until the number of pages in every new produced cluster is less than or equal to a preferred number *pn* (e.g. 20). All clusters produced during this procedure hierarchically cluster the Web pages. Fig. 5.6 shows this clustering diagram. The clustering procedure is depicted as the following Algorithm1, *Clustering1*, where | *SM* | stands for the number of rows (columns) of the square matrix *SM*.

	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9
P_1	1	0	0	0	.47	0	.6	0	0
P_2	0	1	0	0	0	0	.49	0	0
P_3	0	0	1	0	0	0	0	.7	.66
P_4	0	0	0	1	0	0	0	0	.42
P_5	.47	0	0	0	1	0	.9	0	0
P_6	0	0	0	0	0	1	0	.92	.45
P_7	.6	.49	0	0	.9	0	1	0	0
P_8	0	0	.7	0	0	.92	0	1	0
P_9	0	0	.66	.42	0	.45	0	0	1

(a)

	P_1	P_5	P_7	P_2	P_4	P_9	P_3	P_8	P_6
P_1	1	.47	.6	0	0	0	0	0	0
P_5	.47	1	.9	0	0	0	0	0	0
P_7	.6	.9	1	.49	0	0	0	0	0
P_2	0	0	.49	1	0	0	0	0	0
P_4	0	0	0	0	1	.42	0	0	0
P_9	0	0	0	0	.42	1	.66	0	.45
P_3	0	0	0	0	0	.66	1	.7	0
P_8	0	0	0	0	0	0	.7	1	.92
P_6	0	0	0	0	0	.45	0	.92	1

(b)

Fig. 5.5. (a) A similarity matrix. (b) The permuted matrix of (a)

[**Algorithm1**] *Clustering1* (*SM, pn*)
[**Input**] *SM*: similarity matrix; *pn*: preferred page number in each cluster;
[**Output**] $CL = \{CL_i\}$: a set of hierarchical clusters;
Begin

Set $CL = \varnothing$; Permute *SM* such that (5.4) is maximized;
Decompose *SM* such that (5.5) is maximized;
If $| SM_{1,1} | \le pn$, **then** do
converting $SM_{1,1}$ into the next CL_i; $CL = CL \cup \{CL_i\}$;
else do
converting $SM_{1,1}$ into the next CL_i; $CL = CL \cup \{ CL_i \}$;
Clustering1 ($SM_{1,1}$, *pn*);
If $| SM_{2,2} | \le pn$, **then** do
converting $SM_{2,2}$ into the next CL_i; $CL = CL \cup \{CL_i\}$;
else do
converting $SM_{2,2}$ into the next CL_i; $CL = CL \cup \{ CL_i \}$;
Clustering1 ($SM_{2,2}$, *pn*);
Return *CL*;

End

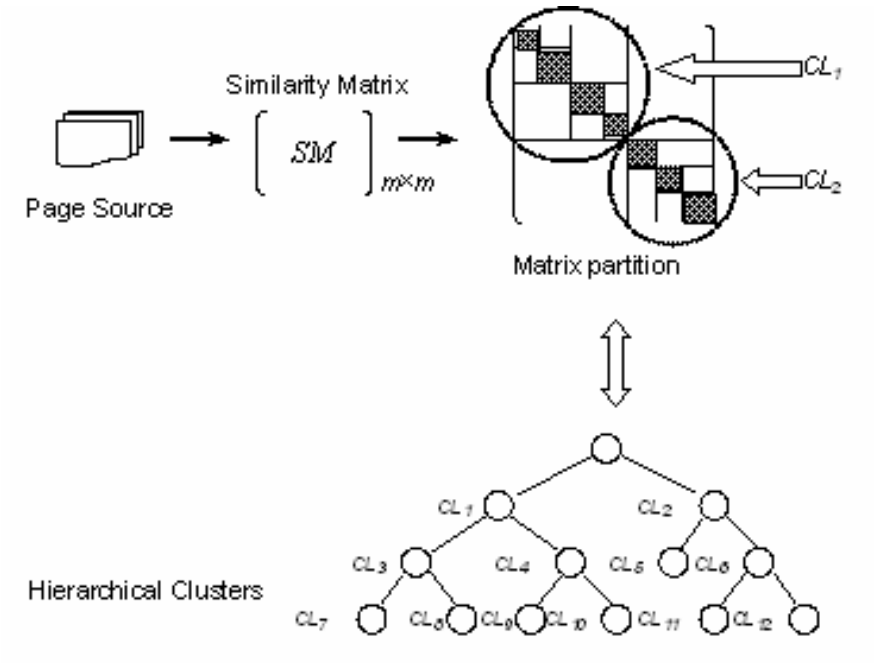


Fig. 5.6. Matrix-based hierarchical clustering diagram

5.3.3 Cluster-Overlapping Algorithm

For the above algorithm *Clustering1*, there exists no overlapping among the clusters that are produced at the same level. Each page belongs to only one of the clusters at the same level. In practice, however, it is reasonable that a page might belong to several same level clusters. On the other hand, the non-zero element values in $SM_{1,2}$ or $SM_{2,1}$ represent the similarity between two pages, named *cross-related pages*, that belong to two different clusters. If a cross-related page in one cluster has a higher similarity with another cluster, it is possible for this page to be added to another cluster (sub-matrix) to form a new cluster in case it will be missed out. For these reasons, the hierarchical clustering algorithm *Clustering 1* could be improved such that the cluster overlapping among the same level clusters is permitted.

To determine whether a cross-related page in one cluster could be added to another cluster, we define a *centroid* of the cluster $SM_{p,p}$ ($1 \leq p \leq 2$) as $CE(SM_{p,p}) = \{CE^{row}(SM_{p,p}), CE^{col}(SM_{p,p})\}$, which consists of two vectors

(row and column vectors) that are constructed from the correlation matrix C as

$$CE^{row}(SM_{p,p}) = \frac{1}{|SM_{p,p}|}\sum_{j\in SM_{p,p}} row_j \,, CE^{col}(SM_{p,p}) = \frac{1}{|SM_{p,p}|}\sum_{j\in SM_{p,p}} col_j \tag{5.6}$$

The centroid of a cluster is a logical page representing this cluster. For a pair of cross-related pages $p\in SM_{1,1}$ and $q\in SM_{2,2}$, if $sim(p, CE(SM_{2,2})) \geq t$, then page p could be added to $SM_{2,2}$ to form a new cluster (sub-matrix) $SM'_{2,2}$ with the dimension being increased by 1, where t is a threshold defined as the average non-zero similarities in $SM_{1,2.}$ The page q could be treated in the similar way. The following Algorithm2 *Extending* depicts this cross-related page treatment.

[**Algorithm2**] *Extending* (*SM*)
[**Input**] *SM*: similarity matrix with sub-matrices $SM_{1,1}$, $SM_{2,2}$, $SM_{1,2}$ and $SM_{2,1}$;
[**Output**] $SM'_{1,1}$, $SM'_{2,2}$: new sub-matrices (clusters) with some added cross-related pages;
Begin
Compute the centroids $CE(SM_{1,1})$ and $CE(SM_{2,2})$ according to (5.6);
Compute the threshold t, which is the average non-zero similarities in $SM_{1,2}$;
Set N_1=[|$SM_{1,1}$|*0.15]; N_2 = [|$SM_{2,2}$| * 0.15]; $N = \min(N_1, N_2)$;
Construct page set P={p|at least one sm_p, $j\neq 0$, $1\leq p\leq d$, $d+1\leq j\leq m$};
Construct page set Q={q|at least one sm_i,$q\neq 0$, $1\leq i\leq d$, $d+1\leq q\leq m$};
Compute P_SM_{22} = {$sim(p, CE(SM_{2,2})|p\in P$, $sim(p, CE(SM_{2,2}) \geq t$};
Compute Q_SM_{11} = {sim(q, $CE(SM_{1,1})|q\in Q$, sim(q, $CE(SM_{1,1}) \geq t$};
Add up to N pages in $SM_{2,2}$ that correspond to the N highest values in Q_SM_{11} into $SM_{1,1}$ to form a new sub-matrix $SM'_{1,1}$;
Add up to N pages in $SM_{1,1}$ that correspond to the N highest values in P_SM_{22} into $SM_{2,2}$ to form a new sub-matrix $SM'_{2,2}$;
Return $SM'_{1,1}$ and $SM'_{2,2}$;
End

The parameter d is the row (or column) number of the dividing point D in *SM*. The parameter N in this algorithm is used to restrict the number of pages to be added to $SM_{1,1}$ and $SM_{2,2}$, which guarantees the recursive execution of the matrix partition. This parameter, on the other hand, also guarantees that the added cross-related pages could not change (or dominate) the main property of the original clusters, i.e. the most number of cross-related pages added to a cluster is less than or equal to 15% of the original page number in this

cluster. This percentage could be adjusted according to the practical requirements.

When n pages that belong to cluster $SM_{2,2}$ are added into cluster $SM_{1,1}$, the corresponding new sub-matrix $SM'_{1,1}$ is formed by adding n columns of $SM_{1,2}$ and n rows of $SM_{2,1}$ into the original $SM_{1,1}$ with the dimension being increased by n. These added columns and rows correspond to these n added pages. The main diagonal elements of the newly produced $n \times n$ lower-right sub-matrix of $SM'_{1,1}$ are set to 1, and other elements in this sub-matrix are set to 0. The construction of $SM'_{1,1}$ is intuitively shown in Fig. 5.7 For the construction of $SM'_{2,2}$, the procedure is the same.

Based on the above cluster overlapping treatment algorithm *Extending*, the matrix-based hierarchical clustering algorithm with cluster overlapping is depicted as the following Algorithm3 *Clustering2*.

[**Algorithm3**] *Clustering2* (*SM, pn*)
[**Input**] *SM*: similarity matrix; *pn*: preferred page number in each cluster;
[**Output**] $CL = \{CL_i\}$: a set of hierarchical clusters;
Begin
Set $CL = \varnothing$; Permute *SM* such that (5.4) is maximized;
Decompose *SM* such that (5.5) is maximized;
$\{SM'_{1,1}, SM'_{2,2}\}$ = *Extending* (*SM*);
If $| SM'_{1,1} | \le pn$, **then** do
converting $SM'_{1,1}$ into the next CL_i; $CL = CL \cup \{CL_i\}$;
else do
converting $SM'_{1,1}$ into the next CL_i; $CL=CL\cup\{CL\}$;
Clustering2 ($SM'_{1,1}$, *pn*);
If $| SM'_{2,2} | \le pn$, **then** do
converting $SM'_{2,2}$ into the next CL_i; $CL = CL\cup\{CL_i\}$;
else do converting $SM'_{2,2}$ into the next CL_i; $CL=CL\cup\{CL_i\}$;
Clustering2 ($SM'_{2,2}$, *pn*);
Return *CL*;
End

This clustering algorithm enables some pages in R to be clustered into several same level clusters, which is reasonable in practice and enables users to find some pages from different paths in the hierarchical cluster structure. It is not difficult to prove that the complexity of the above clustering algorithms is $O(m^2)$, where m is the number of pages to be clustered.

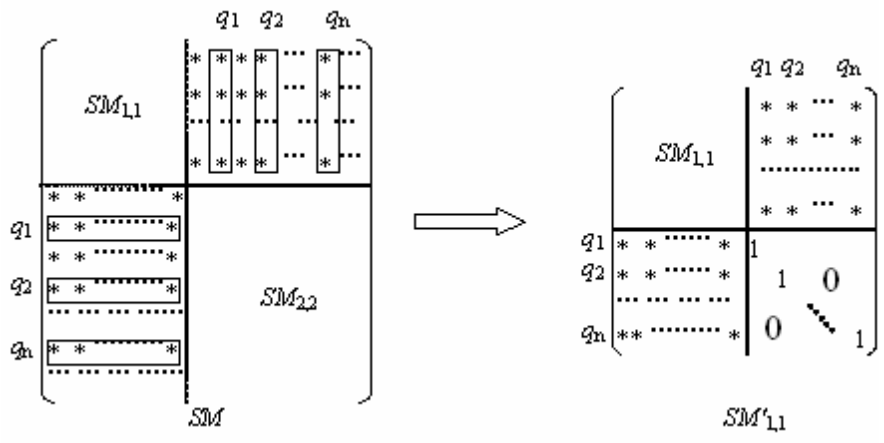


Fig. 5.7. Construction of new sub-matrix $SM'_{1,1}$

5.4 Co-Citation Algorithms

The citation and co-citation analysis were originally developed for scientific literature index and clustering, and then extended to the Web page analysis. For better understanding of the algorithms to be proposed, we firstly present some background knowledge of the citation and co-citation analysis, and then give the Extended Co-Citation algorithm for relevant page finding.

5.4.1 Citation and Co-Citation Analysis

The citation analysis was developed in information science as a tool to identify core sets of articles, authors, or journals of particular fields of study (Larson 1996). The research has long been concerned with the use of citations to produce quantitative estimates of the importance and impact of individual scientific articles, journals or authors. The most well-known measure in this field is Garfield's *impact factor (Garfield 1972)*, which is the average number of citations received by papers (or journals) and was used as a numerical assessment of journals in Journal Citation Reports of the Institution for Scientific Information.

The co-citation analysis has been used to measure the similarity of papers, journals or authors for clustering. For a pair of documents p and q, if they are both cited by a common document, documents p and q is said to be *co-cited*. The number of documents that cite both p and q are referred to as *co-citation degree* of documents p and q. The similarity between two documents is

measured by their co-citation degree. This type of analysis has been shown to be effective in a broad range of disciplines, ranging from author co-citation analysis of scientific subfields to journal co-citation analysis. For example, Chen and Carr (Chen and Carr 1999) used author co-citation analysis to cluster the authors, as well as the the research fields. In the context of the Web, the hyperlinks are regarded as citations beween the pages. If a Web page p has a hyperlink to another page q, page q is said to be cited by the page p. In this sense, citation and co-citation analyses are smoothly extended to the Web page hyperlink analysis. For instance, Larson (Larson 1996), Pitkow and Pirolli (Pitkow and Pirolli 1997) have used the co-citation to meaure the Web page similarities.

The above co-citation analyses, whether for scientific literatures or for Web pages, is mainly for the purpose of clustering, and the page source to which the co-citation analysis is applied is usually a pre-known page set or a Web site. For example, the page source in (Pitkow and Pirolli 1997) was the pages in a Web site of Georgia Institute of Technology, and the page source in (Larson 1996) was a set of pages in *Earth Science* related Web sites. When the co-citation analysis is applied for relevant page finding, however, the situation is different. Since there exists no pre-known page source for the given page and co-citation analysis, the success of co-citation analysis mainly depends on how to effectively construct a page source with respect to the given page. Meanwhile, the constructed page source should be rich in related pages with a reasonable size.

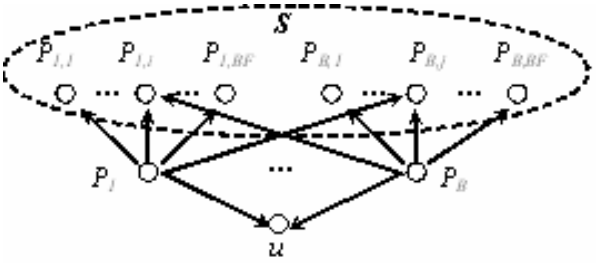


Fig. 5.8. Page source S for the u in *DH Algorithm*

Dean and Henzinger (Dean and Henzinger 1999) proposed a co-citation algorithm to find the relevant pages. Hereafter, we denote it as the *DH Algorithm*. In their work, for a given page (URL) u, the page source S with respect to u is constructed in the following way: the algorithm firstly chooses up to B (e.g. 2000) arbitrary parents of u; for each of these parents p, it adds to S up to BF (e.g. eight) children of p that surround the link from p to u. The elements of S are siblings of u as indicated in Fig. 5.8. Based on this page source S, the co-citation algorithm for finding relevant pages is as follow: for each page s

in *S*, the co-citation degree of *s* and *u* is determined; the algorithm finally returns the 10 pages that have the highest co-citation degrees with *u* as the relevant pages.

Although the *DH Algorithm* is simple and the page source is of a reasonable size (controlled by the parameters *B* and *BF*), the page source construction only refers to the parents of the given page *u*. It is actually based on an assumption that the possible related pages fall into the set of siblings of *u*. Since the child pages of *u*, and accordingly the page set derived from these child pages, are not taken into account in the page source construction, many semantically related pages might be excluded in the page source and the final results may be unsatisfactory. This is because the semantic relationship conveyed by the hyperlinks between two pages is mutual. If a page *p* is said to be semantically relevant (via hyperlinks) to another page *q*, page *q* could also be said to be semantically relevant to page *p*. From this point of view, the children of the given page *u* should be taken into consideration in the page source construction.

5.4.2 Extended Co-Citation Algorithms

For a given page *u*, its semantic details are most likely to be given by its in-view and out-view (Mukherjea and Hara 1997). The in-view is a set of parent pages of *u*, and out-view is a set of child pages of *u*. In other words, the relevant pages with respect to the given page are most likely to be brought into the page source by the in-view and out-view of the given page. The page source for finding relevant pages, therefore, should be derived from the in-view and out-view of the given page, so that the page source is rich in the related pages.

Given a Web page *u*, its parent and child pages could be easily obtained. Indeed, the child pages of *u* can be obtained directly by accessing the page *u*; for the parent pages of *u*, one way to obtain them is to issue an *AltaVista* query of the form *link: u*, which returns a list of pages that point to *u (Bharat and Henzinger 1998)*. The parent and child pages of the given page could also be provided by some professional servers, such as the Connectivity Server (Bharat et al. 1998). After the parent and child pages of *u* are obtained, it is possible to construct a new page source for *u* that is rich in related pages. The new page source is constructed as a directed graph with edges indicating hyperlinks and nodes representing the following pages:

1. Page *u*,
2. Up to *B* parent pages of *u*, and up to *BF* child pages of each parent page that are different from *u*,

3. Up to F child pages of u, and up to FB parent pages of each child page that are different from u.

The parameters B, F, BF and FB are used to keep the page source to a reasonable size. In practice, we choose $B = FB = 200$, $F = BF = 40$. This new page source structure is presented intuitively in Fig. 5.9. Before giving the Extended Co-Citation algorithm for finding relevant pages, we firstly define the following concepts.

Definition 1: Two pages p_1 and p_2 are *back co-cited* if they have a common parent page. The number of their common parents is their *back co-citation degree* denoted as $b(p_1, p_2)$. Two pages p_1 and p_2 are *forward co-cited* if they have a common child page. The number of their common children is their *forward co-citation degree* denoted as $f(p_1, p_2)$.

Definition 2: The pages are *intrinsic pages* if they have same page domain name.

Definition 3: (Dean and Henzinger 1999): Two pages are *near-duplicate pages* if (a) they each have more than 10 links and (b) they have at least 95% of their links in common.

Based on the above concepts, the complete Extended Co-Citation algorithm to find relevant pages of the given Web page u is as follow:

Step 1: Choose up to B arbitrary parents of u.

Step 2: For each of these parents p, choose up to BF children (different from u) of p that surround the link from p to u. Merge the intrinsic or near-duplicate parent pages, if they exit, as one whose links are the union of the links from the merged intrinsic or near-duplicate parent pages, i.e. let P_u be a set of parent pages of u,

$P_u = \{p_i \mid p_i$ *is a parent page of u without intrinsic and near-duplicate pages,* $i \in [1, B]\}$,

let $S_i = \{s_{i,k} \mid s_{i,k}$ *is a child page of page* p_i, $s_{i,k} \neq u$, $p_i \in P_u$, $k \in [1, BF]\}$, $i \in [1, B]$.

Then Step 1 and 2 produce the following set

$$BS = \bigcup_{i=1}^{B} S_i .$$

Step 3: Choose first F children of u.

Step 4: For each of these children c, choose up to FB parents (different from u) of c with highest in-degree. Merge the intrinsic or near-duplicate child pages, if they exist, as one whose links are the union of the links to the merged intrinsic or near-duplicate child pages, i.e. let C_u be a set of child pages of u,

$C_u = \{c_i \mid c_i$ *is a child page of u without intrinsic and near-duplicate pages,* $i \in [1, F]\}$,

let $A_i = \{a_{i.k} \mid a_{i,k}$ *is a parent page of page* c_i, $a_{i,k}$ *and u are neither intrinsic nor near-duplicate pages*, $c_i \in C_u$, $k \in [1,FB]\}$, $i \in [1, F]$.

Then Step 3 and 4 produce the following set

$$FS = \bigcup_{i=1}^{F} A_i \ .$$

Step 5: For a given selection threshold δ, select pages from *BS* and *FS* such that their back co-citation degrees or forward co-citation degrees with *u* are greater than or equal to δ. These selected pages are relevant pages of *u*, i.e., the relevant page set *RP* of *u* is constructed as:

$$RP = \{\ p_i \mid p_i \in BS \text{ with } b(p_i, u) \geq \delta \ \text{ OR } \ p_i \in FS \text{ with } f(p_i, u) \geq \delta\ \}.$$

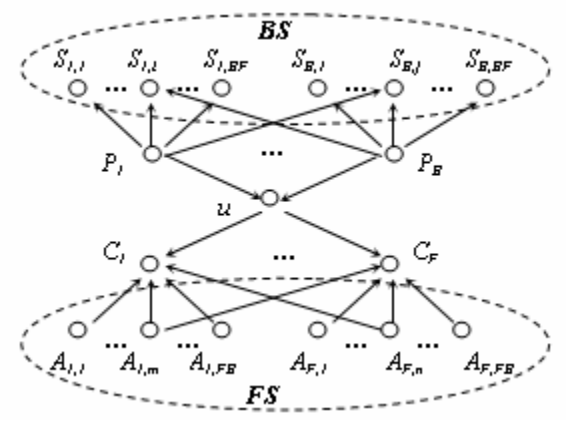


Fig. 5.9. Page source structure for the Extended Co-Citation algorithm

It can be seen from this algorithm that, in the parent page set P_u and child page set C_u of *u*, the intrinsic or near-duplicate pages are merged as one. This treatment is necessary for the success of the algorithm. Firstly, this treatment can prevent the searches from being affected by malicious hyperlinks. In fact, for the pages in a Web site (or server) that are hyperlinked purposely to maliciously improve the page importance for Web search, if they are imported into the page source as the parent pages of the given page u, their children (the siblings of *u*) most likely come from the same site (or server), and the back co-citation degrees of these children with *u* would be unreasonably increased. With the merger of the intrinsic parent pages, the influence of the pages from the same site (or server) is reduced to a reasonable level (i.e. the back co-citation degree of each child page with u is only 1) and the malicious hyperlinks are shielded off. For example, in Fig. 5.10, suppose the parent pages P_1, P_2, P_3 and their children $S_{1.1}$, …, $S_{3.2}$ be intrinsic pages. In situation

(a), the back co-citation degree of page $S_{2.2}$ with u is unreasonably increased to 3, which is the ideal situation the malicious hyperlink creators would like. The situation is the same for the pages $S_{1.2}$ and $S_{3.1}$. With the above algorithm, the situation (a) is treated as the situation (b) where P is a logic page representing the union of P_1, P_2, P_3, and the contribution of each child page from the same site (or server) to the back co-citation degree with u is only 1, no matter how tightly these intrinsic pages are linked together.

Secondly, for those pages that are really relevant to the target page u and located in the same domain name, such as those in Web sites that are concerned about certain topics, the above intrinsic page treatment would probably decrease their relevance to the given page u. However, since we consider the page relevance to the given page within a local Web community (page source), not just within a specific Web site or server, this intrinsic page treatment is still reasonable in this sense. Under this circumstance, there exists a trade-off between avoiding malicious hyperlinks and keeping as much useful information as possible. Actually, if such pages are still considered as relevant pages within the local Web community, they would be finally identified by the algorithm. The above reasons for intrinsic parent page treatment are the same for the intrinsic child page treatment, as well as the near-duplicate page treatment.

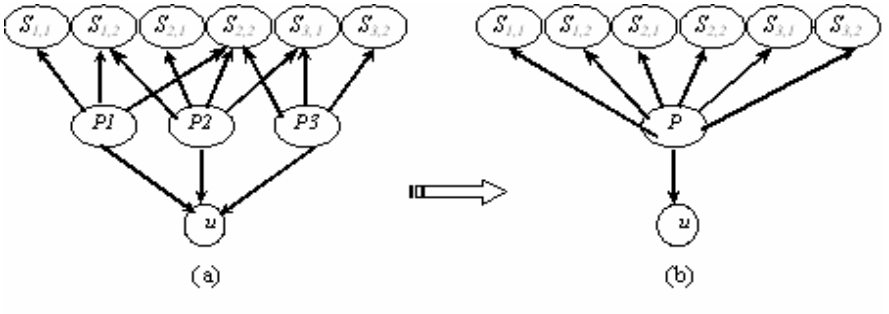


Fig. 5.10. An example of intrinsic page treatment

It is also worth noting that even if the given page u contains active links (i.e. links to hub pages that are also cited by other pages), the algorithm, especially the pages set A_i, can also shield off the influence of malicious hyperlinks from the same site or server or mirror site of u. On the other hand, however, this page set A_i would probably filter those possible relevant pages that come from the same domain name of u. The trade-off between avoiding malicious hyperlinks and keeping useful information still exists in this circumstance. If the algorithm is only used within a specific Web site or domain name, it can be simplified without considering the intrinsic page treatment. In

other words, in the Extended Co-Citation algorithm, the influence of each Web site (or server) to the page relevance measurement is reduced to a reasonable level, and a page's relevance to the given page is determined within a local Web community (page source), rather than only within a specific Web site or server.

6 Building a Web Community

In this Chapter, we mainly discuss algorithms that construct Web communities. We introduce Web communities as complete directed bipartite graphs in Sect. 6.1. In Sect. 6.2, we briefly introduce the notion of small world which indicates that such subgraphs do exist on Web. Algorithms for finding such complete directed bipartite graphs will be discussed in Sect. 6.3. Sect. 6.4 discusses finding Web communities as dense directed bipartite graphs which is an approach to relax the conditions imposed on the complete directed bipartite graphs. Sect. 6.5 introduces two approaches to find Web communities in arbitrary shapes. In Sect. 6.6 and 6.7, we discuss the algorithms on finding connections among Web communities and exploring the Web community evolution patterns. In Sect. 6.8, a graph-theoretical approach is introduced, which tries to answer the question when a found Web community can be determined as unique.

6.1 Web Community

Web community is a community on the Web, and is a collection of Web pages created by those who share the same interests or specific topics. In (Gibson et al. 1998), Gibson et al.. presented a view on Web communities that can be found using the *HITS* algorithm. In the abstract of their paper (Gibson et al. 1998), it states: *The communities can be viewed as containing a core of central, "authoritative" pages linked together by "hub pages"; and they exhibit a natural type of hierarchical topic generalization that can be inferred directly from the pattern of linkage.* In (Kumar et al. 1999), Kumar et al.. also give two different kinds of Web communities: explicitly-defined and implicitly-defined communities. Explicitly-defined Web communities are widely-known communities and can be found in *Yahoo* or *AOL* Web sites, with newsgroups, mailing-list, and a long list of resources linking to all related Web sites. Implicitly-defined communities may have rather too-specific interest or may be too detail that is typically difficult for Web portals to identify and provide enough resources for the users interested. Both kinds of Web communities serve the same purposes: to provide linkages among groups of

people who are possibly far away from each other; and to share their information and knowledge. In fact, as pointed out in (Kumar et al. 1999), the number of implicitly-defined communities can be much larger and outnumber the explicitly-defined communities. One unique feature of implicitly-defined communities is that implicitly-defined communities may appear as an emerging Web community for some specific topic or new event or new business partners and may disappear later. Web community finding is a technique to identify such Web communities systematically.

What does a Web community look like? In (Gibson et al. 1998), Gibson et al. did not define any particular types of graphs as the Web community to find. They explored the possibilities of discovering the Web communities using *HITS* with the issues in mind such as how the Web communities can be found (converged) with various of root sets and with different iterations. In (Kumar et al. 1999), Kumar et al. observed that a Web community as a graph may have different shapes which are difficult to be identified in general. However, a Web community must at least have a community core. They formalized it as a complete directed bipartite graph.

Definition 6.1. *A directed bipartite graph* is a directed graph $G(V, E)$ where V is a set of nodes that is divided into two disjoint subsets, V_f and V_c, and E is a set of edges where each edge is from a node in V_f to a node in V_c.

Definition 6.2. *A complete directed bipartite graph* $G_C(V, E)$ is a directed bipartite graph, where V $V_f \cup V_c$, such as every node in V_f has an edge to every node in V_c. A complete directed bipartite graph is denoted as $G_{C_{f,c}}$ where $|V_f| = f$ is V_c c.

An example of complete directed bipartite graph, $G_{C_{3,3}}$, is shown in Fig. 6.1. It has a set of nodes which is divided into two disjoint sets $V_f = \{f_1, f_2, f_3\}$ and $V_c = \{c_1, c_2, c_3\}$. Every node f_i in V_f, for i , links to every node c_j in V_c, for $j = 1, 2, 3$.

Definition 6.3. *A Web community (or Web community core)* is a complete directed bipartite graph, $G_C(V, E)$. The nodes in V_f and V_c, are called fans and centers, respectively.

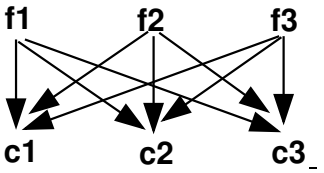


Fig. 6.1. A complete directed bipartite graph

In a Web community, a center represents a Web page the Web community in question is interested in, and a fan represents a Web page that links to the Web pages as a center. As shown above, instead of specifying an arbitrary Web community using a set of parameters, which are uneasy to be determined, now it specifies a Web community, as a complete directed bipartite graph, $G_{C_{f,c}}$, using only two parameters, namely, the number of fans (f) and the number of centers (c). Consequently, Web community finding becomes a problem of finding complete directed bipartite graphs. The hypothesis here is that a Web community contains relatively dense bipartite subgraphs, which in turn leads to the existence of a complete directed bipartite graph. The hypothesis is strongly supported by the *HITS* for its excellent performance on searching a wide range of topics on Web.

Why is Web community a completed directed complete graph at the beginning? In other words, why do fans not have edges to other fans, and why do centers not have edges to other centers? Kumar et al. in (Kumar et al. 1999) give the following reasons. They can be competitors in a Web community. They do not want to have links from themselves to their competitors because it may redirect their own customers to their competitors, even though all of them as a whole are a part of the same community. The centers may have totally opposite views on the same issue of, for example, on gun control, abortion rights, death penalty, elections, etc. On the other hand, the fans may not have links to each other, simply because they as fans can be anywhere and may not know and do not need to know each other. The ignorance of other parties can also be seen at the center side.

6.2 Small World Phenomenon on the Web

Before we discuss the other types of Web communities and the algorithms for finding Web communities, we discuss one related interesting issue called *small world phenomenon*. The small world phenomenon addresses that the length of the path between any two nodes in a graph is small on average. The implication of the existence of such rather short paths implicitly suggests that

there are possibly many small clusters of Web-pages and there are many possible Web communities, like complete directed bipartite graphs.

The small world problem was first studied by Stanley Milgran et al. in the field of social networking to study the short chains of acquaintances between any two citizens, who do not know each other in United States. One well-known experiment was conducted as follows. A citizen (source) in Nebraska delivered a letter to another citizen (target) in Massachusetts. The source was only given the basic information about the target, including the address and occupation to deliver the letter. Everyone, including the source on the chain, should try best to deliver the letter to the target, but could only pass the letter to the next person known on a first-name basis. Surprisingly, over many trials, the average number of the length of the chains was between five and six.

Kleinberg in (Kleinberg 2000; Kleinberg 2002) studied the algorithmic perspective of the small world problem, which addresses how people find these chains when they know little about the target. We show the model and the decentralized algorithm below.

Kleinberg's model follows the previous work done by Watts and Strogatz (Duncan and Watts 1998) by considering two factors, short-range (local) connections and long-range connections. To model a network that exhibits the richness of local connections and a few random long-range connections, Kleinberg used a two-dimensional grid where the size of the grid is $n \times n$, and the edges connecting two points on the grid are directed edges. In other words, a point on the grid has many connections to all its neighbor points in all directions, but a few connections to points in a rather long distance. Let (i, j) denote a point on the grid. The distance between two points (i, j) and (k,l) is measured as $dist((i, j),(k,l)) = |k-i| + |l-j|$. For local connections, a parameter $p(\geq 1)$ is introduced. The parameter p specifies the neighbors of a point on the grid. For example, if $p = 1$, then all the points that are within a distance 1 of a specific point are the neighbors of the specific point on the grid. For long-range connections, two parameters q and r are introduced. Here, the parameter q specifies how many long-range connections a point has, and the parameter r specifies how they are linked using independent random trials. Literally, the *i*-th directed edge from a point p_1 at position (x, y) to a point p_2 at position , is with probability proportional to $dist\ p_1, p_2\ ^{-r}$ for $r \geq 0$. The probability distribution is obtained with a normalization by dividing $dist\ p_1, p_2\ ^{-r}$ by the total sum of $\sum_{p_1} [dist(p_1, p_2)]^{-r}$. The decentralized algorithm follows a simple strategy: every message-holder chooses a contact that is as close to the target as possi-

ble in terms of the distance measurement. The message-holder does not know anything about how the messages have been delivered so far. Theorem 3 of (Kleinberg 2002) is given below for reference.

Theorem 6.4. *(Theorem 3 of (Kleinberg 2002)): (a) Let* $0 \le r < 2$. *There is a constant* α, *depending on* p, q, r, *but independent of* n, *so that the expected delivery time of any decentralized algorithm is at least* $\alpha n(2-r)/3$. *(b) Let* $r > 2$. *There is constant* α, *depending on* p, q, r, *but independent of* n, *so that the expected delivery time of any decentralized algorithm is at least* $\alpha n(r-2)/(r-1)$.

Another Theorem in *(Kleinberg 2002)* states that only when $r = 2$, independent of n, there is a decentralized algorithm that can produce chains whose length is a polynomial in $\log n$. In other words, there exists such a decentralized algorithm but decentralized algorithms cannot be found in other settings. Some other works on small world and graph structures in the Web can be found in (Duncan and Watts 1998; Adamic 1999; Broder et al. 2000; Dill et al. 2001).

The Theorem 6.4 together with other results in (Kleinberg 2002) showed that it is possible for people, following local information (the nearby Web pages connected locally), to link to a remote Web page she/he is interested in.

6.3 Trawling the Web

As the first attempt, Kumar and his co-workers used actual large Web archive to investigate whether the Web communities (complete directed bipartite graphs) exist in (Kumar et al. 1999; Kumar et al. 1999).

Kumar and his co-workers conducted a two week experimental studies using an 18 month Web archive obtained from Alexa, which was a company that archived the state of the Internet (Kumar et al. 1999). The archived Web data contained over 200 million Web pages. First, Kumar et al. identified the potential fans as specialized hubs using *HITS*. With human interactions, they determined that a potential fan (Web page) has links to at least 6 different Websites where a Website is the first field of the URL. Out of the 200 million Web pages, 24 million pages were used as potential fans. They further adopted an aggressive mirror-elimination strategy to remove 60% of the 24 million pages. Second, they maintained the number of potential centers as roughly 30 times of the number of potential fans. They also adapted a pruning strategy by in-degree to prune the centers that were highly referenced such as

Yahoo or *Altavista*, based on the observation that a large number of pages linking to the Web portals may not necessarily have anything to do with the content of the pages. They pruned the Web pages as centers that had more than 50 in-degree. In other words, the pages referenced by more than 50 Web pages were pruned. Finally, they tested $G_{C_{f,c}}$, for f is taken from $\{3,4,5,6\}$ and c is taken from $\{3,5,7,9\}$. The numbers of non-nepotistic Web communities, denoted as $|G_{C_{f,c}}|$, found in Kumar's experimental study are listed below. Here, non-nepotistic links are the links from a hub, as the fans, to other sites. $|G_{C_{3,3}}|=38,887$, $|G_{C_{3,5}}|=30,299$, $G_{C_{3,7}}$ 26 800 and $|G_{C_{3,9}}|=$, . The Web communities do exist.

Based on the results in (Kumar et al. 1999), Kumar et al. extended their experimental study and proposed a graph model in (Kumar et al. 1999). They considered how users create a Web page on the Internet to attract more potential visitors to visit their Web pages. A typical case they observed was that users added more recreational sailing pages obtained from the well-known Websites to attract visitors. Following the observation, adding links from a Web page, v, can be modeled as picking an existing Web page, u, and copying some links from the Web page, u, to the Web page v. This also helps to model how Web pages with a wide range of topics are created. Initially, there are only a few Web pages about an emerging topic. When more people show their interest in the new topic, the number of Web pages that reference the topic increases significantly. It may trigger more users to reference these Web pages in their own Web pages.

A class of graph models are specified by four stochastic processes at a time step t, namely, a creation process for node-creation, a creation process for edge-creation, a deletion process for node-deletion, and a deletion process for edge-deletion. First, node-creation and node-deletion are with probability $\alpha_c\ t$ and $\alpha_d\ t$, respectively. Second, edge-creation is modeled using a probability β. Consider a node, v, to be randomly selected to add edges. With probability β, it adds k edges from v to other nodes independently and uniformly selected. With probability $1-\beta$, it chooses another node u randomly, and copies k out-going edges of u to v. That is, an edge (u,w) of node u is copied to node v as a new edge (,). Third, edge deletion is modeled as to randomly delete an edge with a probability $\delta(t)$. Finally, they also showed that both in-degree distribution of nodes and out-degree distribution of nodes are of Zipfian distribution, when $\alpha_c(t)=1$, $\alpha_d\ t =$, and $\delta(t)=0$ (a node is created at all times, and no node-deletion and no edge-

deletion occur at all times). In brief, let $p_{i,t}$ and $q_{i,t}$ be the fraction of nodes at time t with in-degree i and out-degree i, then

$$p_{i,t} = 1/i^{1/(1-\alpha)} \tag{6.1}$$

$$q_{i,t} = 1/i^{1/(1-\beta)} \tag{6.2}$$

Here, α and β are two probabilities for creating a new edge (u, v). The following is the description of the edge creation process given in (Kumar et al. 1999): – *The "destination" v is set based on the α coin: if it comes up heads, v is the newest page, and otherwise, v is the destination of a random link. The "source" u is set based on the β coin: if it comes up heads, u is the newest page, and otherwise, u is the source of a random link.* When $\alpha = 0.52$ and $\beta = 0.58$, the probability of a node that has in-degree i is $i^{-2.1}$ and the probability of a node that has out-degree i is $i^{-2.38}$. They remarked that both match the reality of Web. Greco et al. also studied a stochastic approach for modeling and computing Web communities (Greco et al. 2002).

6.3.1 Finding Web Communities Based on Complete Directed Bipartite Graphs

A Web community can be obtained from a complete directed bipartite graph, $G_{C_{f,c}}$, from a Web graph G as follows, based on the *HITS* algorithm. Note: the nodes of a Web community (a complete directed bipartite graph) are divided into two sets, V_c (centers) and V_f (fans). First, the root set, R, as the preparation step for using *HITS*, are obtained as follows.

$$\begin{aligned} R = V_c \cup V_f \cup \{v_j \mid v_i \in V_f \wedge (v_i, v_j) \in G\} \cup \\ \{v_j \mid v_i, v_k \in V_c \wedge (v_j, v_i) \in G \wedge (v_j, v_k) \in G\} \end{aligned} \tag{6.3}$$

As seen above, the root set includes the nodes in $G_{C_{f,c}}$ and the nodes being pointed by the nodes V_f and the nodes pointing to at least two nodes in V_c. Second, using the *HITS* algorithm, a set of authorities and a set of hubs can be identified which represent the centers and the fans of the corresponding Web community.

6.4 From Complete Bipartite Graph to Dense Directed Bipartite Graph

In Sect. 6.3.1, we showed how Kumar et al. find complete directed bipartite graphs as Web communities existing in Web communities. Recall: a complete directed bipartite graph $G_{C_{f,c}}$ have f fans and c centers, and every fan must have a directed link to every center. A natural question that arises is whether the condition is too strong. Can it miss any interesting Web communities? In order to relax the condition imposed on complete bipartite graph, Reddy and Kitsuregawa in (Reddy and Kitsuregawa 2001) proposed a dense directed bipartite graph, which we will discuss mainly in this section. It is interesting to know that the relaxation of bicliques is also discussed in data clustering (Mishra et al. 2003).

Definition 6.5. *A dense directed bipartite graph* $G_D\ V, E$ *is a directed bipartite graph, where* $V = V_f \cup V_c$. *A node in* V_f *must link to at least* $\gamma_c \quad \leq \gamma_c \leq |V_c|$ *nodes in* V_c, *and at least* γ_f $(1 \leq \gamma_f \leq |V_f|)$ *nodes in* V_f *link to every node in* V_c. *A dense directed bipartite graph is denoted as* $G_{D_{\gamma_c,\gamma_f}}$.

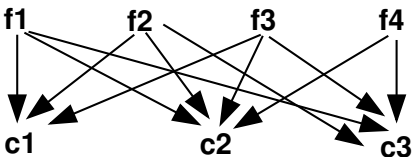


Fig. 6.2. A dense directed bipartite graph

An example of dense directed bipartite graph, $G_{D_{2,3}}$, is shown in Fig. 6.2. It has a set of nodes which are divided into two disjoint sets, $V_f = \{f_1, f_2, f_3, f_4\}$ and $V_c = \{c\ , c\ , c\ \}$. A node f_i in V_f links to at least two nodes in V_c, and there are three nodes (f_1, f_2, f_3) that link to every node c_j in V_c, for $j = 1, 2, 3$. It is important to know that $G_{C_{\cdot}} \subseteq G_{D_{\cdot}}$. In this example, a $G_{C_{\cdot}}$ can be a set of nodes $\{f_1, f_2, c_1, c_2, c_3\}$ and the edges connected them in $G_{D_{2,3}}$.

A Web community is then defined as a dense directed bipartite graph in (Reddy and Kitsuregawa 2001). Comparing Definition 6.5 with Definition

6.2, the condition in Definition 6.2 is weakened, because it does not request that every fan must link to every center. Following Definition 6.5, a complete directed bipartite graph is a special case of a dense directed bipartite graph. Therefore, given a dataset, the resulting set of complete directed bipartite graphs found are contained in the resulting set of dense directed bipartite graphs. In other words, given a dataset, W, and two parameters γ_f and γ_c. Let $G_C = \{G_{C_{f,c}} \mid G_{C_{f,c}} \subseteq W \wedge f \geq \gamma_f \wedge c \geq \gamma_c\}$ and $G_D = \{G_{D_{f,c}} \mid G_{C_{f,c}} \subseteq W \wedge f \geq \gamma_c \wedge c \geq \gamma_f\}$. $G_C \subseteq G_D$.

Definition 6.6. *A Web community* is a dense directed bipartite graph, $G_D(V, E)$

Furthermore, in (Reddy and Kitsuregawa 2001), Reddy and Kitsuregawa defined a Web community hierarchy. It is worth noting that a Web community is treated as a set of nodes in this Web community hierarchy. In other words, the fans at the level $i-1$ become centers at the level i. Upon a specific level i, a Web community is a dense directed bipartite graph.

Definition 6.7. *A Web community* C_{ij} is the j-th community at the level i, where $C_{ij} = V_f$ of a $G_{D_{f,c}}$ at level i – 1 such as the sizes of f and c, for $G_{D_{f,c}}$, are greater than or equal to the minimum required numbers of fans (γ_f) and centers (γ_c) respectively, and $|V_c| \leq \beta \cdot \gamma_c$, where β is a positive number as a parameter to control the relaxation of the Web community as dense directed bipartite graphs.

As shown above, there are three parameters to relax the conditions imposed on complete directed bipartite graphs, γ_f, γ_c and β. Both γ_f and γ_c relax the condition that a Web community must be a complete directed bipartite graph. In other words, not all centers need to be pointed to by fans and not all fans need to point to centers. The last parameter β further enlarges the pool of centers ($|V_c|$) up to $\beta \cdot \gamma_c$. Note: β is a positive number.

6.4.1 The Algorithm

In following, we introduce the algorithm in (Reddy and Kitsuregawa 2001) to extract Web communities as dense directed bipartite graphs. We call it *Comm-DBG*, which is shown in Algorithm 6.1. The algorithm takes five pa-

rameters, a large graph G for the Web dataset, θ for specifying a dense directed bipartite graph along with γ_f and γ_c, and β for controlling the relaxation. *Comm-DBG* will return a set of Web communities on which the Web community hierarchy can be easily built. There are two main steps in *Comm-DBG*. As shown in Algorithm 6.1, the first step (line-1) is to generate a set of candidates on which Web communities can be found by calling a procedure *Gen* (Algorithm 6.2). The second step is to find dense directed bipartite graphs as Web communities (line-2) by calling a procedure *Find-DBG* (Algorithm 6.3).

Algorithm 6.1 *Comm-DBG*($G, \theta, \gamma_f, \gamma_c, \beta$)

Input: a directed graph G, a cocited threshold $\theta\ (>0)$, and three positive numbers, γ_f, γ_c and β.
Output: a set of dense directed bipartite graphs;
1. $U \leftarrow$ *Gen*(G, θ, K); K is a pregiven parameter for controling the maximum number of iterations.
2. $E \leftarrow$ *Find-DBG*(U, G, γ_f, γ_c, β);
3. **return** E;

Generating Candidate Sets

Algorithm 6.2 illustrates how to generate a set of candidate sets for the Web dataset, as a large directed graph G. A for-loop statement is used to generate a candidate set for a node $_i \in$, as shown in line 2-12. The candidate set for v_i is U_1, which is initialized as only to contain v_i (line 3), and will be inserted into the set of U at line 12, where U is the set of candidate sets to be returned at the end of the procedure (line 13). A relationship between a node and a set of nodes serves the basis to generate a candidate set. The relationship is called *Cocited*, and is defined in Definition 6.8. The Cocited relationship is an extension of the cocited (co-citation) defined for citation analysis and bibliometrics. Given two papers, v_i and v_j. The papers, v_i and v_j, are cocited if they cite the same papers (at least one). In other words, given two nodes, v_i and v_j in a graph G, the two nodes, v_i and v_j are cocited if the intersection of $child(v_i)$ and $child(v_j)$ is not empty, where $child(v_k)$ is the set of child nodes of v_k. Note: the cocited relationship between two nodes can be used to generate candidate sets for constructing complete directed bipartite graphs, but is too strong to be used to generate candidate sets for constructing dense directed bipartite graphs. In order to

generate such candidate sets for dense directed bipartite graphs, Cocited relationship is defined in Definition 6.8 in (Reddy and Kitsuregawa 2001).

Definition 6.8. Given a set of nodes, V, and a node, v_i ($\notin V$), in a directed graph G. Let $Cocited(v_i, V) = child(v_i) \cap (\cup_{v_j \in V} child(v_j))$ where $child(v_k)$ is the set of child nodes of v_k. The node v_i and the set V are cocited, if $|Cocited(v_i, V)| \geq \theta$.

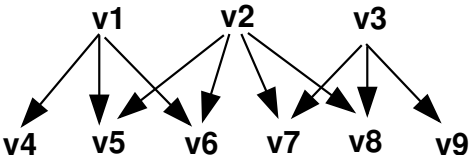


Fig. 6.3. Cocited

As can be seen above, the Cocited relationship is defined as a relationship between a node and a set of nodes, rather than between two nodes. A node v_i is in a Cocitation relationship with a set V if v_i points (cites) to some nodes pointed (cited) by some nodes in V. A threshold, θ, is introduced in Definition 6.8. A node v_i and V are in a co-citation relationship if v_i points at least θ nodes pointed by some nodes in V. Fig. 6.3 shows three nodes v_1, v_2 and v_3 are Cocited, when $\theta = 2$. First, let $V = \{v_1\}$, v_2 is Cocited with V, because $|Cocited(v_2, V)| \geq 2$. Second, let $V = \{v_1, v_2\}$, v_3 is Cocited with V (v_1 and v_2), because $|Cocited(v_3, V)| \geq 2$. Therefore, the three nodes, v_1, v_2 and v_3, are Cocited. Note: v_1 and v_3 do not have any common child nodes.

In Algorithm 6.2, at line 8, it checks if a node v_j is Cocited with a set of nodes U_1. U_1 will be enlarged K times. K is needed as shown in Fig. 6.3. Let $V = \{v_1\}$. Suppose it first checks whether v_3 is Cocited with V (v_1). Obviously, v_3 is not Cocited with v_1 because $|Cocited(v_3, V)| = 0$, which is less than $\theta = 2$. With the parameter K, v_3 can possibly be Cocited with v_1 after v_2 is first Cocited with v_1.

Algorithm 6.2 *Gen(G , θ , K)*

Input: a directed graph $G(V, E)$, a cocited threshold $\theta\ (>0)$, and controling of iteration $K > 0$.
Output: a set of sets U.

1. $U \leftarrow \varnothing$;
2. **for** each $v_i \in V$ **do**
3. $U_1 \leftarrow \{v_i\}$;
4. $k \leftarrow 1$;
5. **while** $k \leq K$ **do**
6. $U_2 \leftarrow \varnothing$;
7. **for** each $v_j \in V$ **do**
8. **if** Cocited(v_j, U_1) $\geq \theta$ **then**
9. $U_2 \leftarrow U_2 \cup \{v_j\}$;
10. $U_1 \leftarrow U_1 \cup U_2$;
11. $k \leftarrow k+1$;
12. $U \leftarrow U \cup \{U_1\}$;
13. **return** U.

Algorithm 6.3 *Find-DBG*(U, G, γ_f, γ_c, β)

Input: a set of sets U, a graph G, and three non-negative integers γ_f, γ_c and β.
Output: a set of dense directed bipartite graphs.

1. $E \leftarrow \varnothing$;
2. **for** each $V_i \in U$ **do**
3. $c \leftarrow \gamma_c$;
4. $f \leftarrow \gamma_f$;
5. $E_i \leftarrow \varnothing$;
6. **for** each $v_i \in V_i$ **do**
7. insert the edge of (v_i, v_j) into E_i if $v_j \in child(v_i)$;
8. again: for a goto statement below
9. **repeat**
10. sort E_i based on the destination of edges;
11. delete (v_i, v_j) from E_i if $parent(v_j) < c$;
12. sort E_i based on the source of edges;
13. delete (v_i, v_j) from E_i if $child(v_i) < f$;
14. **until** E_i is converged
15. $\leftarrow \{\ \ |(\ ,\ ) \in\ \ \}$;
16. **if** $\beta \cdot$ **then**
17. f f 1;
18. **goto** again;
19. $E \leftarrow E \cup \{E_i\}$; E_i represents a dense directed bipartite graph
20. **return** E.

Finding Web Communities

Algorithm 6.3 illustrates how to find Web communities as dense directed bipartite graphs. The parameter U is the set of candidate sets generated by *Gen* (Algorithm 6.2). For each candidate set $V_i \in U$, *Find-DBG* attempts to find a Web community, E_i, in three steps. First, lines 6-7 create an initial E_i, based on V_i, including all edge pairs (v_i, v_j) for $v_i \in V_i$ and $v_j \in child(v_i)$. Second, lines 9-14 remove edges, (v_i, v_j), from E_i, either if v_j as a center does not have enough fans ($| parent(v_i) | < \gamma_f$) or if v_i as a fan does not have enough centers to point to ($| child(v_i) | < \gamma_c$). The idea behind this is to remove those that can not be participated in a dense directed bipartite graph. The second step will continue until E_i is converged. The removing process can be done using two sorting methods. One is to sort based on v_i as the source, and the other is to sort based on v_j as the destination of edges. The third step, lines 15-17 are to control the relaxation using β. This step first checks whether the number of centers is greater than $\beta \cdot c$ (Refer to Definition 6.3). If so, it increases the number of fans allowed and goes to "again" to repeat the second and the third steps. *Find-DBG* will return a set of such E_i's.

6.5 Maximum Flow Approaches

In the previous sections, a Web community is based on either a complete directed bipartite graph or a dense directed bipartite graph. They all need to specify the sizes of the two disjoint subsets of nodes using two parameters. A question that arises is whether it is possible to find Web communities of arbitrary shapes or diameters. Maximum flow is one of the possible alternative to identify Web communities. Flake et al. studied such an approach in (Flake et al. 2000; Flake et al. 2002), and Imafuji and Kitsuregawa improved the quality the Flake and his co-workers work in (Imafuji and Kitsuregawa 2003; Imafuji and Kitsuregawa 2004).

6.5.1 Maximum Flow and Minimum Cut

We introduce the maximum flow and minimum cut below, as the background of this section. The details can be found in (Cormen et al. 1990; Papadimitriou and Steiglitz 1998).

Definition 6.9. Let $N(s,t,V,E,c)$ be a flow network. Here, V is a set of nodes, and E is a set of directed edges $E \subseteq V \times V$ of a directed graph $G(V,E)$. And, s and t are two distinctive nodes in V, called source and terminal, respectively. Each edge $(v_i,v_i) \in E$, is associated with a capacity c, denoted $c(v_i,v_j)$. The capacity specifies the upper bound of data that can flow from v_i to v_j via the edge (v_i,v_j).

Definition 6.10. Given a flow network $N(s,t,V,E,c)$, a flow is a function, denoted as $f(v_i,v_j)$, for every edge $(v_i,v_j) \in E$, satisfying three properties:

- $0 \le f(v_i,v_j) \le c(v_i,v_j)$.
- For all nodes $v \quad V \quad \{s \ t\}$, $\sum_{(u_i,v)\in E} f(u_i,v) = \sum_{(v,v_j)\in E} f(v,u_j)$.

The $s - t$ maximum flow problem over N is to find the maximum value of $\sum_{v\in V} \quad (\ ,\ )$.

In Definition 6.10, the first condition states that a flow on an edge cannot exceed its capacity (its upper bound). The second condition states that, except for the source s (with a zero in-coming flow) and the terminal t (with a zero out-going flow), the in-coming flow and out-going of any node must be the same. Note: $c(v_i,v_j)$ and $f(v_i,v_j)$ be zero if $(v_i,v_j) \notin E$.

Definition 6.11. Given a flow network $N \ s,t,V,E,c$. The $s - t$ minimum cut over N is to find the minimum value of a cut set, $C \ V_s,V_t$, when dividing the set of nodes V into two disjoint sets, V_s and V_t, such as $s \in V_s$ and $t \in V_t$. The cut set is given as $C(V_s,V_t) = \{(v_i,v_j) \mid (v_i,v_j) \in E \wedge v_i \in V_s \wedge v_j \in V_t\}$, the capacity of a cut set is denoted as $val(C) \quad \sum_{(v_i,v_j)\in C(V_s,V_t)} c(v_i,v_j)$, and the size of a cut set is the number of edges in the cut set, denoted as $|C|$.

The s-t maximum flow problem is identical to the $s - t$ minimum cut problem over the same flow network N, as the theorem of Ford and Fulk-

erson (Cormen et al. 1990; Papadimitriou and Steiglitz 1998). Based on the theorem of Ford and Fulkerson, many efficient algorithms were proposed. In (Cormen et al. 1990), Cormen et al. named all the algorithms that follow the theorem of Ford and Fulkerson, as *Ford-Fulkerson-Method*. The *Ford-Fulkerson-Method* initializes all flows $f(v_i, v_j)$ to be zero, and iteratively increase the flow value along a path, called augmenting path, if more flows can be pushed into the path. Flake, Lawrence and Giles proposed an algorithm called *ISA* for incremental shortest augmentation, which is given in (Flake et al. 2000), and Imafuji and Kitsuregawa used the Edmonds-Karp algorithm in (Cormen et al. 1990).

6.5.2 FLG Approach

In (Flake et al. 2000; Flake et al. 2002), Flake, Lawrence and Giles considered a flow network $N(s,t,V,E,c)$ with one modification such as the graph $G(V,E)$ is a undirected graph, rather than a directed graph as originally defined in the flow network. We denote the modified flow network as $N(s,t,V,E,c)$ below. Flake et al. defined a Web community with the flow network in mind as follows. The graph $G(V,E)$ is the undirected graph in the modified network flow N.

Definition 6.12. *A Web community is a set of nodes $V_c \subseteq V$ over the graph $G(V,E)$ such as every node v in V_c has more edges connecting to the other nodes in V_c than the nodes in $V - V_c$.*

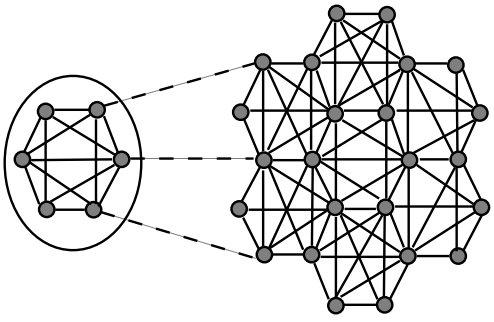


Fig. 6.4. A Web Community

An example Web community (Definition 6.12) is shown as the left side dense subgraph in Fig. 6.4 (Fig. 1 of (Flake et al. 2000)). The three dashed

edges form the cut between the left and right side subgraphs. In a theorem in (Flake et al. 2000), Flake et al. showed that a Web community V_c (a set of nodes) can be identified over a flow network N by finding the $s-t$ minimum cut, C, of N on the condition that $|C| < \#(s)$ and $|C| < \#(t)$, where $|C|$ is the size of a cut set over the flow network N, $\#(s)$ is the number of edges between s and other nodes in V_c, and $\#(t)$ is the number of edges between t and $V - V_c - \{t\}$. The Web community is the set of nodes, after the minimum cut, that are still reachable from the source node s. The theorem reemphasizes the fact that every node in a Web community has more connections with other nodes in the same Web community than the connections with the nodes outside of the Web community (the other size of the cut), because the cut is minimum to separate the community from the rest part of the graph. The conditions that both $\#(s)$ and $\#(t)$ are greater than the cut size ensure that the Web community found is meaningful.

Following Definition 6.12, a Web community is a set of nodes of graph, $G(V, E)$, which is a part of a flow network N. The question is how to identify the source, s, and, the terminal, t. Flake et al. selected two virtual nodes as the source and terminal.

The source, s, is a virtual node that connects to a set of user given seed nodes, $S \subset V$. The virtual source s is assigned to an infinite capacity to each of the seed node. The set of seed nodes together provides good opportunities to identify a Web community with higher accuracy. Because the virtual source s has infinite capacities to all of the seed nodes, all the seed nodes have high potential to be reachable from the source s after the cut – to cut the Web community from the rest of the graph.

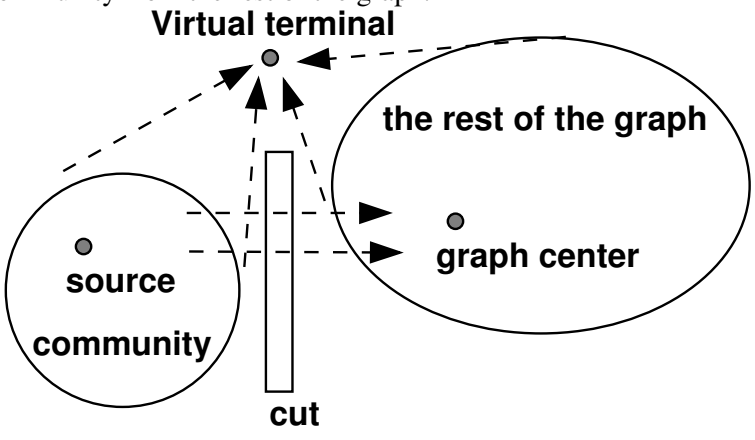


Fig. 6.5. A cut

The choice of the terminal t has great impacts on the Web community and is intuitively difficult to obtain. Flake et al. gave a smart way of selecting a virtual terminal and proved the correctness. They formalize the problem as whether the cut set remains the same given two different terminals, the best, t, and the heuristic $\hat{t}$. As illustrated in Fig. 6.5 (Based on Fig. 3 of (Flake et al. 2000)), the best terminal, t, is considered as the graph center, the virtual terminal, $\hat{t}$, is a node newly added. The virtual terminal $\hat{t}$ is connected with all nodes in V, except the seeds, s and itself, with the minimum capacity 1. The rectangle in Fig. 6.5 shows the cut set between the Web community and the rest of the graph.

Let us consider two flow networks over a Web graph $G(V,E)$ with a set of seed nodes $S \subseteq V$.

- $N_1 = (s,t,V_1,E_1,c_1)$: Here, $V_1 = V \cup \{s\}$, and $E_1 = E \cup E'$ where $E' = \{(s,v) \mid v \in S\}$. c_1 includes the capacity for each edge (s,v) in E', which are all infinite, $c(s,v) = \infty$, as well as the capacity for each edge in E. t is the graph center and is in V.
- $N_2 = (s,\hat{t},V_2,E_2,c_2)$. Here, as the virtual s, $\hat{t}$ is a virtual node. $V_2 = V \cup \{s,t\}$, and $E_2 = E \cup E' \cup E''$ where $E' = \{(s,v) \mid v \in S\}$, and $E'' = \{(v,t) \mid v \in V\}$. c_2 includes the capacity for each edge, (s,v) in E', which are all infinite, $c(s,v) = \infty$, the capacity for each edge (v,t) in E'', $c(v,t) = 1$, as well as the capacity for each edge in E.

Suppose that there are two cut sets, after applying s – t minimum cut, $C_1(V_{1_s}, V_{1_t})$ and $C_2(V_{2_s}, V_{2_t})$ found from N_1 and N_2, respectively. Both V_{1_s} and V_{2_s} include the source s. The questions are: can the two cut sets, C_1 and C_2, possibly be the same? And what conditions can make it possible? Flake et al. showed that it is possible in a theorem. We introduce their founding below.

Theorem 6.13. (Theorem 2 of (Flake et al. 2000)) Given two flow networks, $N_1 = (s,t,V_1,E_1,c_1)$ and $N_2 = (s,\hat{t},V_2,E_2,c_2)$. Let $C_1(V_{1_s}, V_{1_t})$ and $C_2(V_{2_s}, V_{2_t})$ be the minimum cut sets for N_1 and N_2, respectively. If the condition $1 < k < |V_{1_t}| / val(C_1)$ holds, then $C_2 = C_1 \cup C'$ such as C' only includes the edges $(v,\hat{t}) \in E_2$ for $v \in V_{2_s}$.

Flake, Lee and Giles sketched the proof as follows. First, increasing the capacity of all nodes equally by a factor of k cannot change the minimum cut set. Second, if all nodes in V_{2_t} are connected to the terminal $\hat{t}$, the minimum cut set of N_1 will be still effective in N_2, if $|V_{1_t}| > k \cdot val(C_1)$. Third, if all nodes, v, in V_{2_s} are connected to $\hat{t}$ with a unit capacity edge, it is more efficient to cut the edge between such a v and $\hat{t}$ than to remove v from the community, V_{2_s}. Fourth, $|V_{1_s}| + k \cdot val(C_1) < |V_{1_s}| + |V_{1_t}|$ as implied by $k < |V_{1_t}| / val(C_1)$. Note: for the two cuts, C_1 and C_2, $V_{1_s} = V_{2_s}$ and $V_{1_t} = V_{2_t} - \{\hat{t}\}$.

Flake, Lee and Giles proposed two algorithms, called *Exact-Flow-Community* (Algorithm 6.4) and *Approximate-Flow-Community* (Algorithm 6.5). Both find Web communities based on a given set of source nodes, S, as seeds. The *Exact-Flow-Community* algorithm assumes that the whole Web data as a graph is archived and exists. On the other hand, The *Approximate-Flow-Community* algorithm does not assume that the whole Web data already exists in hand, and uses a subset of the entire Web data.

The *Exact-Flow-Community* algorithm takes three inputs, a set of seed nodes, S, and the entire Web data as a graph, G, and an integer k. In Algorithm 6.4, line 1-11, it creates a flow network with two virtual nodes, source s and terminal t. All edges are associated with a capacity. Then, line 12, it calls a *Ford-Fulkerson-Method* to identify a minimum cut set, (v_s, v_t) over the flow network. In line 13, the nodes still connected to s after the cut will be returned as the Web community identified by the set of seed nodes.

The *Approximate-Flow-Community* algorithm (Algorithm 6.5) takes three inputs, a set of seed nodes, S, and two integers, d and m. Note: it does not have the entire Web as an input into the algorithm. It attempts to enlarge the set of seed nodes, S, in an iterative way. In each iteration, line 3, it crawls from the set of seed nodes, S, with a given depth d, and constructs a partial Web data set as a graph G. The parameter k (See Theorem 6.13) is set to be $|S|$ (line 4). In line 5, it calls *Exact-Flow-Community* to get a set of nodes, V_s. Then, it ranks all the nodes in V_s by the number of edges in V_s (line 6). And in line 7, it adds the highest ranked non-seed nodes into the set of seed nodes, S. This procedure will repeat m times.

Flake et al. conducted the theoretical analysis and reported their findings in (Flake et al. 2004).

Algorithm 6.4 *Exact-Flow-Community*(S , G , k)

Input: a set of seed nodes, S , a graph $G(V, E)$ where $S \subseteq V$, and a positive number k .
Output: a set of nodes
1. Let s and t be two additional virtual nodes;
2. $V \leftarrow V \cup \{s,t\}$;
3. **for** each $v \in S$ **do**
4. add (s,v) into E with an infinite capacity, $c(s,v) \leftarrow \infty$;
5. **for** each $(u,v) \in E$ **do**
6. $c(u,v) \leftarrow k$;
7. **if** $(v,u) \notin E$ **then**
8. add (v,u) into E with $c(v,u) \leftarrow k$;
9. **for** each $v \in V$ such as $v \notin S \cup \{s,t\}$ **do**
10. add (v,t) into E with $c(v,u) \leftarrow 1$;
11. let $N(s,t,V,E,c)$ be a flow network;
12. $C(V_s,V_t) \leftarrow$ Ford-Fulkerson-Method(N);
13. **return** all $v \in V_s$ still connected to s ;

Algorithm 6.5 *Approximate_flow_community*(S , d , m)

Input: a set of seed nodes, S , and two positive numbers, d and m .
Output: a set of nodes
1. $n \leftarrow 1$;
2. **while** $n < m$ **do**
3. $G(V,E) \leftarrow$ a graph crawl from S with a depth d ;
4. $k = |S|$
5. $V_c \leftarrow$ Exact-Flow-Community(S , G , k);
6. rank all v in V_c by the number of edges in V_c ;
7. add highest ranked non-seed nodes to S ;
8. $n \leftarrow n+1$;
9. **return** all $v \in V$ still connected to S ;

6.5.3 IK Approach

One of the key issues in the FLG approach is the edge capacity, c , and the parameter k used in Algorithm 6.4. As shown in Theorem 6.13, the parameter k plays an important role in making the minimum cut remains unchanged with virtual source and terminal. Flake et al. did not show how to determine the edge capacity.

Imafuji and Kitsuregawa show the impacts of the edge capacity on the quality of the Web community being found in (Imafuji and Kitsuregawa

2003; Imafuji and Kitsuregawa 2004). They conducted the testing using a real Web archive. Three observations have been made. First, increasing edge capacity can increase the size of Web community. Second, it is difficult to determine proper edge capacity. For example, the worst case is that there only exists a so-called quantum jump point of edge capacity. When the edge capacity is less than the quantum jump point, the Web community, based on a set of seed nodes, is too small. When the edge capacity is greater than or equal to the quantum jump point, the whole graph will be the Web community. Third, the noises in the Web community cannot be easily removed by controlling the edge capacity. Here, noises are the Web pages in the Web community but are not related to the main topic of the Web community.

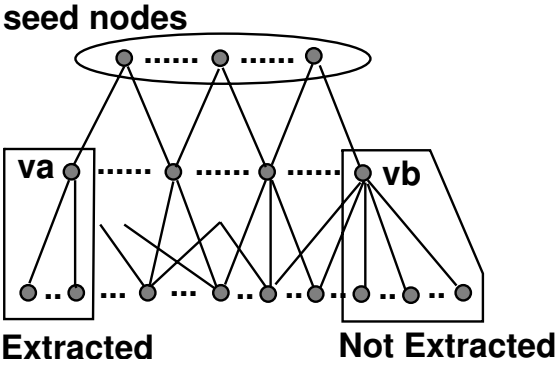


Fig. 6.6. The impacts of edge capacity

Below, we introduce the analysis made in (Imafuji and Kitsuregawa 2004) for the third observation, using an example shown in Fig. 6.6 (Fig. 4 of (Imafuji and Kitsuregawa 2004)). As shown in Fig. 6.6, the set of seed nodes is illustrated as the top level nodes. The second layers are the nodes that connected to the seeds, as the immediate neighbors. Consider the nodes at the second layers. There are two cases.

- Case-1: A second layer node, v, connects many third layer nodes, $u_1, u_2, \cdots$. The number of the nodes, u_i, that have degree 1, is less than the edge capacities.
- Case-2: A second layer node, v, connects many third layer nodes, $u_1, u_2, \cdots$. The number of the nodes, u_i, that have degree 1, is much greater than the edge capacities.

Which one will be included in the Web community? In Fig. 6.6, v_a is the first case, provided that the nodes pointed to by the node v_a in the rectangle

are the nodes that have degree 1. All of them will be included in the Web community, because the number of such nodes is less than the edge capacity and therefore the flow going into v_a is greater than the flow going out of the rectangle. v_b is the second case, provided that the nodes pointed to by the node v_b in the polygon are the nodes that have degree 1. All of them will not be included in the Web community, because the number of such nodes is much greater than the edge capacity and therefore the flow going into v_b is less than the flow going out of the polygon. Note the nodes that are pointed by v_b and some other nodes at the second layers may be included in the Web community.

Will the nodes in the first case be noises? The nodes in the rectangle pointed by v_a can be noise, if they are pointed by many nodes that are not in the Web community. Increasing the edge capacity will increase the opportunities to include them as noises. Note: using a small edge capacity may make the entire Web community unreasonable small.

Should the nodes in the second case be included in the Web community? The nodes in the polygon should not be included in the Web community, because such a node v_b serves as a hub pointing to too many pages which are most likely not to be related to the Web community.

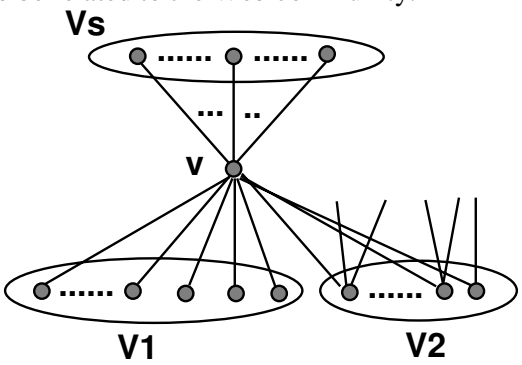


Fig. 6.7. The edge capacity

The Algorithm

Imafuji and Kitsuregawa proposed an algorithm, denoted *IK*. The key issue is how to assign edge capacities to edges. They proposed a *HITS* based approach. They attempted to estimate the quantum jump point, q, that ensures a reasonable size of the Web community. In other words, the quantum jump

point is the minimum largest edge capacity that allows all edges to be unsaturated. Let all the edge capacities be $q-1$. Consider a node v that has the largest degree. The total possible flow going into the node v can be assumed as smaller than flow going out the node v. Such a node v is most likely to be connected to the seed nodes, when the graph G is constructed in *Approximate-Flow-Community* using a depth 2 (Algorithm 6.5, line 3). An example is shown in Fig. 6.7 (Based on Fig. 5 of (Imafuji and Kitsuregawa 2004)) , where v is the node that has the largest degree, d. In Fig. 6.7, V_s represents the set of seed nodes, V_1 represents the set of nodes that are connected with v and are with degree 1, and V_2 represents the set of nodes that are connected with v and are with a degree which is greater than 1. Therefore, the degree of v is

$$d = |V_s| + |V_1| + |V_2|$$

And

$$(q-1)\cdot|V_s| \leq |V_1| + 1 \leq q\cdot|V_s|$$

Here, $|V_1|+1$ represents the edges from v to all nodes in V_1 and an edge to the virtual terminal node. Consequently,

$$q = \left\lfloor \frac{|V_1|}{|V_s|} + 1 \right\rfloor$$

In (Imafuji and Kitsuregawa 2004), the edge capacity of an edge (v_i, v_j) is set as

$$c(v_i, v_j) = \left\lfloor \frac{r_h . h_{v_i} + r_a . a_{v_j}}{2} \right\rfloor \tag{6.4}$$

Here, a_{v_l} and h_{v_k} are the authority score and hub score obtained using *HITS (Kleinberg 1998)*. And, $r_h = \max(a_{v_k}) / \max(h_{v_l})$, $r_a = q$. Both r_h and r_a are needed to make the edge capacity be an integer, because a_{v_l} and h_{v_k} are a real number between zero and 1.

Algorithm 6.6 *IK(S)*

Input: a set of seed nodes, S.
Output: a set of nodes
1. repeat
2. $G(V,E) \leftarrow$ a graph crawl from S with a depth 2;

3. calculate all authority scores, a_{v_i}, and hub scores, h_{v_i}, for all nodes v_i in G, using HITS (Kleinberg 1998).
4. Let s and t be two additional virtual nodes;
5. $V \leftarrow V \cup \{s,t\}$;
6. **for** each $v \in S$ **do**
7. add (s,v) into E with an infinite capacity, $c(s,v) \leftarrow \infty$;
8. **for** each $(u,v) \in E$ **do**
9. $c(u,v)$ is assigned using Eq. (4).
10. if $(v,u) \notin E$ do
11. add (v,u) into E with $c(v,u) = c(u,v)$;
12. **for** each $v \in V$ such as $v \notin S \cup \{s,t\}$ **do**
13. add (v,t) into E with $c(v,u) \leftarrow 1$;
14. let $N(s,t,V,E,c)$ be a flow network;
15. $C(V_s,V_t) \leftarrow$ Ford-Fulkerson-Method(N);
16. let V_c include all $v \in V_s$ that are still connected to S;
17. rank all v in V_c by the number of edges in V_c;
18. add highest ranked non-seed nodes to S;
19. **until** V_c is converged
20. **return** V_c;

IK is outlined in Algorithm 6.6. Different from *Approximate-Flow-Community*, it only takes a set of seed nodes. The flow network is constructed for a set of seed nodes S. The edge capacity is set using Eq. (6.4) based on *HITS (Kleinberg 1998)* (line 2-14). It then calls a *Ford-Fulkerson-Method* to obtain a minimum cut $C(V_s,V_t)$ (line 15). The set of nodes that are still connected to the seeds after the minimum cut is maintained in V_c (line 16). The ranking of all nodes, v_i, in V_c, line 17, is done as follows:

$$rank(v_i) = a_{v_i} \cdot in(v_i) + h_{v_i} \cdot out(v_i).$$

Here, $in(v_i)$ and $out(v_i)$ are the number of edges going into v_i from a node in V_c and the number of edges going out from v_i to a node in V_c. The process will repeat until V_c is converged, (line 1-19). In line 20, the Web community is returned as V_c.

6.6 Web Community Charts

The previous sections discussed Web communities. In this section, we introduce Web community charts (Toyoda and Kitsuregawa 2001; Toyoda and Ki-

tsuregawa 2003). A Web community chart is a graph $G(V, E)$ where V is a set of Web communities and E is a set of edges from a Web community to another Web community. We show an example taken from (Toyoda and Kitsuregawa 2003), as shown in Fig. 6.8 (Based on Fig. 2 of (Toyoda and Kitsuregawa 2003)). In this example, there are two Web communities, a vendor community and a user community. The vendor community consists of companies, IBM, Toshiba and Sony as the centers, as well as the fans which point to them. The user community consists of Sony-PC-fan and Sony-PC-user as the centers, as well as the fans which point to them. Fig. 6.8 (a) shows the two Web communities. There are links from some fans of the user community to some centers of the vendor community. Fig. 6.8 (b) shows the Web community chart. In order to find Web community charts, the notion of Web community needs to be refined, because the centers (authorities) and fans (hubs) are used to specify the Web communities themselves and the relationships among Web communities.

We first discuss the notion of symmetric/asymmetric related pages. On Web, two Web pages, v_i and v_j, are symmetric related if v_i derives v_j and v_j derives v_i. As observed by Toyoda and Kitsuregawa, on Web, it implies that there are other Web pages that point to both v_i and v_j. Two pages are asymmetrically related if either v_i derives v_j or v_j derives v_i, but not both. The derivation relationship between two Web pages is dependent on an algorithm, which derives a Web from another Web page based on some ranking. For example, the algorithm A can be *HITS (Kleinberg 1998)*, *Companion (Dean and Henzinger 1999)*, or *Companion- (Toyoda and Kitsuregawa 2001)*. An example of the asymmetric related Web pages is given in (Toyoda and Kitsuregawa 2001). A fan page of a baseball team, v_i, may derive an official home page of the team, v_j, by an algorithm A. But, with the same algorithm A, the official home page of the team, v_j, may not derive the fan page of the baseball team, v_i. The reasons are, i) the official home page v_j is most likely linked by many hub pages together with other official home pages of baseball teams, and ii) the number of hubs linking to the official home pages of baseball teams (including v_j) is greater than the number of hubs linking to the fans of the baseball teams (including v_i).

Definition 6.14. Given two nodes, v_i and v_j in a graph G. Consider an algorithm, A, which, for a given node v_i, can derive a node v_j. It is said that

v_i derives v_j. If v_i derives v_j and v_j derives v_j, the two nodes v_i and v_j are said to be symmetrically related. If v_i derives v_j but v_j cannot derive v_j, the two nodes v_i and v_j are said asymmetric related.

Definition 6.15. *A Web community* is a dense directed bipartite graph $G(V, E)$ where V is divided into two disjoint subsets, V_1 and V_2. In addition, V_i, for $i = 1, 2$, is a set of nodes that are strongly connected by the symmetric relationship (symmetric related).

Fig. 6.8(c) shows that every two nodes among either fans or centers are symmetric related. In the following, we give the details of the *Companion-* (Toyoda and Kitsuregawa 2001; Toyoda and Kitsuregawa 2003) following the results in (Toyoda and Kitsuregawa 2001; Toyoda and Kitsuregawa 2003) that *Companion-* gives better precision than *HITS* and *Companion* when finding related pages.

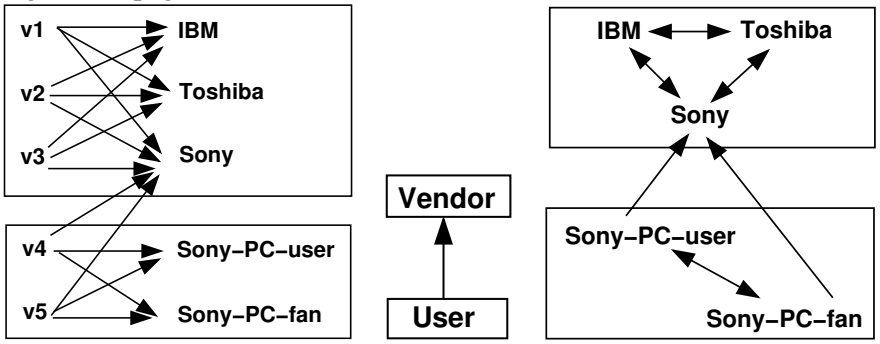


Fig. 6.8. A Web community chart

6.6.1 The Algorithm

The Web community chart shown in Fig. 6.8 is found with a set of seeds including IBM, Toshiba Sony, Sony-PC-fan `and` Sony-PC-user using an algorithm called *Build-Chart* which constructs a Web community chart $G(V, E)$ from Web data.

Algorithm 6.7 outlines the *Build-Chart* algorithm proposed in (Toyoda and Kitsuregawa 2001; Toyoda and Kitsuregawa 2003). *Build-Chart* takes a set of seed pages, V_a, Web data as a graph $G_w(V_w, E_w)$, and a positive number N which is used to derive symmetric related pages. There are four main steps.

First, in line 1-6, it constructs an authority derivation graph, $G_a(V_a, E_a)$, using *Companion-* (Algorithm 6.8) (Toyoda and Kitsuregawa 2003). Second, in line 7, it further extracts a symmetric derivation graph, $G_s(V_s, E_s)$, from the authority derivation graph, $G_a(V_a, E_a)$, found in the first step, using a procedure called *Extract-Symmetric*. Third, it calls a procedure *Extract-core* to extract a set of communities, V. Note: a node in V is a Web community. Finally, it constructs a graph $G(V, E)$ by adding edges between communities. We discuss them below in detail.

Algorithm 6.7 *Build-Chart* (V_a, G_w, N)

Input: a set of seed nodes, V_a, a graph $G_w(V_w, E_w)$ where $V_a \subseteq V_w$, and a positive number N.
Output: a Web community chart $G(V, E)$.
1. $E_a \leftarrow \varnothing$;
2. **for** each $v_i \in V_a$ **do**
3. $V' \leftarrow$ Companion-(v_i, G_w, N);
4. $E' \leftarrow \{(v_i, v_j) \mid v_j \in (V' \cap V_a)\}$;
5. $E_a \leftarrow E_a \cup E'$;
6. Let $G_a(V_a, E_a)$ be a graph;
7. $G_s(V_s, E_s) \leftarrow$ Extract-Symmetric(G_a);
8. $V \leftarrow$ Extract-core(G_s);
9. Construct a Web community chart $G(V, E)$;
10. **return** $G(V, E)$;

The algorithm *Companion-* plays an important role in constructing the authority derivation graph, which is outlined in Algorithm 6.8. For each seed v_s, *Companion-* first, in line 2, identifies a set of nodes V_p pointing to v_s, which is a subset of nodes in the Web graph $G_w(V_w, E_w)$ such as $V_p \subseteq V_w$. Second, in line 2, it further identifies a set of nodes that are pointed to by at least one node in V_p. Third, it identifies a set of nodes, V, including the seed, and V_p and V_c (line 3). Fourth, in line 4, it constructs a vicinity graph $G(V, E)$ where $E = \{(v_i, v_j) \mid v_i \in V \wedge v_j \in V \wedge (v_i, v_j) \in E_w\}$. The set of edges includes all edges that connect two nodes in V. Fifth, line 5-6, it assigns authority weight, $aw(v_i, v_j)$, and hub weight, $hw(a_i, v_j)$, for an edge (v_i, v_j) in the set of edges, E, of the vicinity graph. *Companion-* uses the similar ways of assigning an authority weight and a hub weight to an edge (Bharat and Henzinger 1998), as follows.

- The authority weight and hub weight of an edge, (v_i, v_j), are zero, $aw(v_i, v_j) = 0$ and $hw(v_i, v_j) = 0$, if both v_i and v_j are from the same Web server,
- The authority weigh of (v_i, v_j) is $aw(v_i, v_j) = 1/n$ if there are n edges from the same Web server to v_j and v_i is one of them.
- The hub weigh of (v_i, v_j) is $hw(v_i, v_j) = 1/m$ if there are m edges from v_i to the same Web server, and v_j is one of them.

Sixth, in line 7-9, it computes authority score, $a(v_i)$, and hub score, $h(v_i)$ for any node in V, until all the authority scores and the hub scores are converged. We show them below.

- Initialize $a(v_i)$ and $h(v_i)$, for all $v_i \in V$.
- $h(v_i) = \sum_{(v_i, v_j) \in E} a(v_j) \cdot hw(v_i, v_j)$.
- $a(v_i) = \sum_{(v_j, v_i) \in E} h(v_j) \cdot aw(v_j, v_i)$.

Finally, it returns the top N highest authority nodes from V. Note, each $a(v_i)$ is normalized in a way that the sum of the squares is 1, and each $h(v_i)$ is normalized in a way that the sum of the squares is 1.

Algorithm 6.8 *Companion* (v_s, G_w, N)

Input: a seed node, v_s, a graph $G_w(V_w, E_w)$ where $v_s \in V_w$, and a positive number N.
Output: a set of authorities.
1. $V_p \leftarrow parent(v_s)$;
2. $V_c \leftarrow \cup_{v \in V_p} child(v)$;
3. $V \leftarrow \{v_s\} \cup V_p \cup V_c$;
4. let $G(V, E)$ be a vicinity graph where E is a set of edges $E \subseteq E_w$;
5. **for** each $(v_i, v_j) \in E$ **do**
6. assign an authority weight, $aw(v_i, v_j)$, and a hub weight $ah(v_i, v_j)$;
7. repeat
8. compute an authority score, $a(v_i)$, and a hub score, $h(v_i)$, for every $v_i \in V$;
9. **until** all $h(v_i)$ and $a(v_i)$ are converged
10. **return** the top N highest authority nodes;

Based on the authority derivation graph, $G_a(V_a, E_a)$, the symmetric derivation graph, $G_s(V_s, E_s)$, can be easily constructed where $V_s = V_a$ and $E_s = \{(v_i, v_j) \mid (v_i, v_j) \in E_a \wedge (v_j, v_i) \in E_a\}$.

The Web communities are extracted as follows from the symmetric derivation graph, $G_s(V_s, E_s)$. It finds a complete directed subgraph consisting of 3 nodes – every 2 nodes point to each other. The 3 node complete directed subgraph is called a 1-*connected* subgraph, which is treated as a Web community. There are a set of Web communities (1-connected subgraphs). Let G_1 and G_2 be two Web communities, $G_1 \neq G_2$, G_1 and G_2 may share some common nodes. There are some nodes in V_s that do not belong to any Web communities. Suppose v_i is such a node that does not belong to any Web communities, but is symmetric related to some nodes that are in some Web communities. Such a node will be assigned to a Web community if the Web community has the most incoming edges in the authority derivation graph $G_a(V_a, V_a)$. After the assignments, there may still exist some nodes that do not belong to any Web communities. They are formed as Web communities if they are connected. The procedure *Extract-core* will return a set of such Web communities, V.

The Web community chart, $G(V, E)$, is a weighted directed graph. Based on the authority derivation graph, $G_a(V_a, E_a)$, an edge, v_i, v_j, is in E if there are links from some nodes of the Web community v_i to some nodes in the Web community v_j. The weight is the number of such edges. Fig. 6.9 shows a Web community browser developed by the research group led by Kitsuregawa at the University of Tokyo.

6.7 From Web Community Chart to Web Community Evolution

Web community chart identifies relationships among Web communities. The following question is whether it can find how Web communities evolve, which was studied in (Toyoda and Kitsuregawa 2003).

Let a Web community chart, $G\ V, E$, and a Web community, $v \in G$, at time t, be denoted as $G(t)$ and $v(t) \in G(t)$, respectively. Consider two Web community charts, $G(t_k)$ and $G(t_l)$ where $t_k < t_l$, and t_k and t_l are close

enough to detect Web community changes. There are five cases, at time t_l, in comparison to the Web communities at time t_k $(< t_l)$ namely, emerged Web communities, dissolved Web communities, grown or shrunk communities, split communities, and merged communities (Toyoda and Kitsuregawa 2003). An emerged Web community $v(t_l)$ is a Web community in $G(t_l)$ that does not appear in $G(t_k)$. A dissolved Web community $v(t_l)$ is a Web community in $G(t_k)$ but does not appear in $G(t_l)$. A grown community $v(t_l)$ exists in $G(t_l)$ if $v(t_l)$ is a superset of the corresponding $v(t_k)$ that exists in $G(t_k)$. A shrunk community $v(t_l)$ exists in $G(t_l)$ if it is a subset of the corresponding $v(t_k)$ that exists in $G(t_k)$. A Web community $v(t_k)$ in $G(t_k)$ may be split into two smaller Web communities $v(t_l)$ and $v'(t_l)$. Two smaller Web communities $v(t_k)$ and $v'(t_k)$ in $G(t_k)$ may be merged into a Web community $v(t_l)$ in $G(t_l)$.

Because Web community charts at two different times, t_k and t_l, are extracted from two different Web archives, one of the important issues is to find a mechanism to determine a Web community, $v(t_k)$, that corresponds to a Web community $v(t_l)$, or a Web community, $v(t_l)$, that corresponds to a Web community $v(t_k)$. In addition, a Web community, v, at time t_k, may share URLs with many different Web communities at time t_l. In (Toyoda and Kitsuregawa 2003), the corresponding Web community v, at time t_l (t_k), to the Web community at time t_k (t_l) is the community that shares the most URLs with v at time t_l (t_k).

The evolution metrics are introduced in (Toyoda and Kitsuregawa 2003). Several basic measures are given below.

- $N(v(t_k))$: the number of URLs in the Web community v, at the time t_k.
- $N_{sh}(v(t_k), v(t_l))$: the number of URLs that are shared by the same Web community, v, at times, t_k and t_l.
- $N_{dis}(v(t_k), v(t_l))$: the number of URLs of the same Web community v, that exist at time t_k but disappear at time t_l.

- $N_{sp}(v(t_k), v(t_l))$: the number of URLs of a Web community v that exist at time t_k , but are split to some Web communities at time t_k .
- $N_{ap}(v(t_k), v(t_l))$: the number of URLs of a Web community v that appear at time t_l but do not exist at time t_k .
- $N_{mg}(v(t_k), v(t_l))$: the number of URLs of a Web community v that are merged into v at time t_l from Web communities at t_k .

Some evolution metrics are given below, for $t_j < t_k < t_l$ including the growth rate, R_g , (Eq. (6.5)), the stability, R_s (Eq. (6.6)), the novelty, R_n (Eq. (6.7)), the disappearance rate, R_d (Eq. (6.8)), the merge rate, R_m (Eq. (6.9)), and the split rate, R_p (Eq. (6.10)). These evolution metrics combined can specify a wide range of sophisticated metrics for Web community evolution.

$$R_g(v(t_k), v(t_l)) = \frac{N(v(t_l) - N(v(t_k))}{t_l - t_k} \tag{6.5}$$

()

$$R_s(v(t_k), v(t_l)) = \frac{N(v(t_k)) + N(v(t_l)) - 2N_{sh}(v(t_k), v(t_l))}{t_l - t_k} \tag{6.6}$$

$$R_n(v(t_k), v(t_l)) = \frac{N_{ap}(v(v_k), v(t_l))}{t_l - t_k} \tag{6.7}$$

$$R_d(v(t_k), v(t_l)) = \frac{N_{dis}(v(v_k), v(t_l))}{t_l - t_k} \tag{6.8}$$

$$R_m(v(t_k), v(t_l)) = \frac{N_{mg}(v(v_k), v(t_l))}{t_l - t_k} \tag{6.9}$$

$$R_p(v(t_k), v(t_l)) = \frac{N_{sp}(v(v_k), v(t_l))}{t_l - t_k} \tag{6.10}$$

6.8 Uniqueness of a Web Community

Michael Brinkmeier in (Brinkmeier 2002) investigated Web communities and focused on the *uniqueness* of Web communities using a graph-theoretical approach. Given a subgraph G_h of a graph G, Brinkmeier considers a Web community, for G_h, is the largest subgraph with maximal edge-connectivity of all subgraphs that contains G_h. The hypothesis based on is that nodes of a Web community are supposed to be *stronger* connected to each other than to an arbitrary node outside of the Web community. As notice, different from the previous approaches which find a Web community as a set of Web pages starting from seed pages, Brinkmeier focuses on finding a Web community for a given subgraph.

Several notations are given below before introducing the definition of a Web community. Consider Web as a undirected graph $G(V, E)$ where an edge is associated with a positive weight. For simplicity, the graph does not have self-loops and multiple edges between two nodes. Note: self-loops is an edge from a node to itself. Multiple edges between two nodes can be managed as a single edge with a weight which is the sum of the weights of the multiple edges. A cut E_c of a graph $G(V, E)$ is a subset of edges, $E_c \subseteq E$, such as the resulting graph of G, $C = (V, E, E_c)$ is disconnected. The minimum cut of G is a cut with minimal sum of the weights of the edges in the cut. A minimal cut divides a connected graph into two connected subgraphs. The *edge-connectivity* of G, denoted $conn(G)$, is the weight of a minimal cut of G. Note: different from the minimum cut for flow networks discussed in Sect. 6.5.1, the minimal cut does not consider source and terminal nodes.

Definition 6.16. *Given a subgraph G_h in a graph G, a Web community of G_h is the largest subgraph of the maximal edge-connectivity containing G_h, denoted G_c $(\subseteq G)$, such as*

- $G_h \subseteq G_c$,
- $conn(G_c) \geq conn(G_d)$ if $G_h \subseteq G_d \subseteq G$, and
- $G_d \subseteq G_c$ if $G_h \subseteq G_d \subseteq G$ and $conn(G_d) = conn(G_c)$.

Let $Comm_G(G_h)$ denote the Web community of G_h in a graph G. There exists a unique community $Comm_G(G_h)$ where $Comm_G(G_h)$ is an induced

subgraph of G.[1] The uniqueness can be shown below. Suppose that there are two Web communities for G_h, denoted C_i and C_j. Based on Definition 6.16, $C_i \subseteq C_j$ and $C_j \subseteq C_i$. Therefore, $Comm_G(G_h)$ is unique.

Brinkmeier showed the possible relationships between two Web communities, in a Theorem in (Brinkmeier 2002), such as exact one of the following statements is true for two subgraphs G_{h_1} and G_{h_2} in a graph G.

- $Comm_G(G_{h_1}) \cap Comm_G(G_{h_2}) = \varnothing$.
- $G\ h_1 \subset G\ h_2$.
- $Comm\ G_2 \subset Comm\ G_1$.
- $G(\ h_1) = G(\ h_2)$.

Obviously, the first case is true in some cases. For the remaining, $Comm_G(G_{h_i}) \cap Comm_G(G_{h_j}) \neq \varnothing$. Here, $Comm_G(G_{h_i}) \subseteq Comm_G(G_{h_j})$ if $conn(Comm_G(G_{h_j})) \geq conn(Comm_G(G_{h_i}))$. The second and third cases are when $conn(Comm_G(G_{h_i})) \neq conn(Comm_G(G_{h_j}))$, whereas the fourth case is when $conn(Comm_G(G_{h_i})) = conn(Comm_G(G_{h_j}))$. Based on the above finding, it is shown that communities are completely determined by their nodes and edges in (Brinkmeier 2002).

[1] Given two graphs, $G\ (V_h, E_h)$ and $G(V, E)$. G_h is a subgraph of G if $V_h \subseteq V$ and $E_h \subseteq E$. An induced graph of G_h is a subgraph of G that contains all nodes in V_h and all edges between any two nodes in V_h that exist in E.

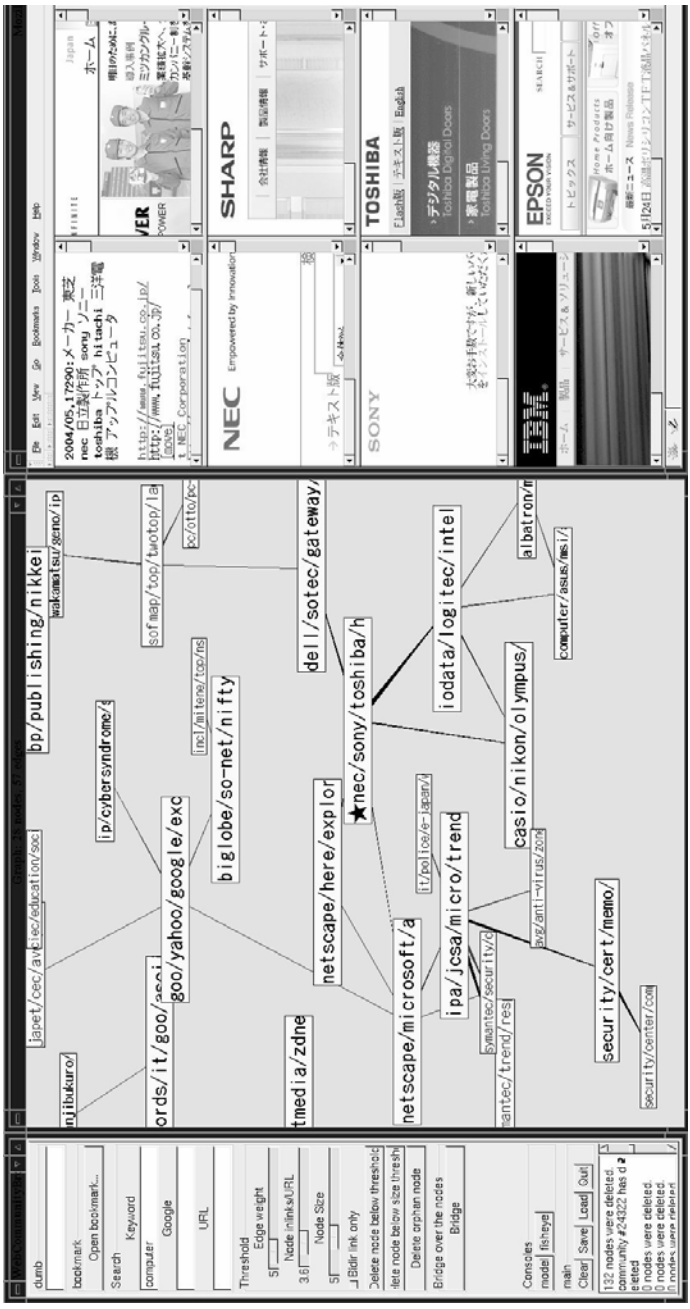


Fig. 6.9. The web community browser developed by Kitsuregawa's group at the University of Tokyo

7 Web Community Related Techniques

In this Chapter, we briefly introduce several techniques that can be used together to either find Web communities or analyze Web communities. Sect. 7.1 introduces a technique that explores Web communities based on user access patterns from Web logs. Sect. 7.2 discusses how to use co-occurrences to enlarge Web communities. Sect. 7.3 introduces Web communities from a high-level Web graph where a node is a Web site rather than a Web page. Sect. 7.4 shows that formal concept analysis can also help to find Web communities. Sect. 7.5 and Sect. 7.6 introduce two approaches to model Web communities. In Sect. 7.5, we introduce an approach to build a model that can capture both the global structure of Web properties as the entire graph and the local Web communities as subgraphs. Sect. 7.6 outlines an idea to model Web communities with attempts to generate Web communities based on some rules. Next two sections, we introduce how to estimate the geographical scope of a Web site (Sect. 7.7), and how to discover the unexpected information from other authoritative Web sites as centers in Web communities (Sect. 7.8). Sect. 7.9 presents an approach based on probabilistic latent semantic analysis, to discover usage-based Web page categories.

7.1 Web Community and Web Usage Mining

Pierrakos et al. presented a way of constructing Web communities using Web logs archived in a proxy server (Pierrakos et al. 2003; Pierrakos et al. 2003). The overview is given below. First, when users access the WWW, their Web access requests go through a proxy server where the Web logs are archived. The Web logs contain the basic information, such as which Web pages are accessed from which IP addresses. Second, based on the archived Web logs, the Web pages being accessed are recorded. A Web page can be abstracted as a feature vector where a feature indicates whether the corresponding word appears in the Web page. The page categories can then be obtained using a hierarchical clustering to cluster Web pages being accessed, where the similarity is based on the words that frequently appear in Web pages. The clustering results in a tree structure where a leaf node represents a cluster of Web

pages which share high similarity. The internal node represents a larger cluster which includes smaller clusters as represented by the child nodes of the internal node. The root node represents all Web pages as a top largest cluster. A page category is a node in the tree. There exist parent/child relationships among page categories. A page category, p_i, is the parent of another page category, p_j, if p_i is the parent node of p_j in the resulting tree. Also, there exist ancestor/descent relationships among page categories. A page category, p_i, is an ancestor of another page category, p_j, if there is a path from p_i to p_j in the resulting tree. A page category is represented as a feature vector which consists of the most important descriptive words of the page category. Third, also based on Web logs, user access sessions can be obtained using an existing Web log mining approach. A user access session, as a feature vector, represents the page categories a user accesses in a time window. Note, a Web page being accessed can be mapped onto the proper page categories. Finally, based on the page categories and access sessions, a Web community can be identified as a maximal clique using an algorithm called *Community Directory Miner* (*CDM*).

The details of the algorithm *CDM* is given below. Consider an edge/node weighted undirected graph $G(V, E)$. Here, V is a set of nodes representing page categories. There exists an edge from node v_i to node v_j if there exists a user access session in which a user accesses both v_i and v_j. A node weight is counted on feature occurrences, and an edge weight is counted on feature co-occurrences. Recall a Web page in a page category (node) is treated as a vector. Let a_{ij} be the value of a feature i in a node v_j, and a^i be the value of a feature i that appears in both nodes v_j and v_k. The node weights and edge weights are computed as follows.

- **Basic Node Weights**: Suppose that there exist n Web pages in a page category. The weight of a feature i is denoted w_i for the page category, and is computed as

$$w_i = \frac{\sum_{j=1}^{n} a_{ij}}{n}.$$

- **Basic Edge Weights**: Suppose that there exist n user access sessions that access both page categories, v_i and v_k. The weight of the edge between v_i and v_k is denoted w_{ik}, and is computed as

$$w_{ik} = \frac{\sum_{j=1}^{n} a_{ik}^{j}}{n}.$$

- **Accumulated Node/Edge Weights**: A weight of a feature in a page category, w_i, is the weight of itself plus all the weights of its child page categories. In a similar fashion, the weight of an edge between page categories, p_i and p_j, is accumulated by the weight of the edge between p_i and p_j as well as all the weights of the edges between p_m and p_n if p_m is a child of p_i and p_m is a child of p_j. As discussed in (Pierrakos et al. 2003; Pierrakos et al. 2003), the accumulated node/edge weights help to include more page categories as a part of Web communities. In other words, a weight of page category or a weight between a pair of page categories may be low, but they can help their ancestors to be a part of Web community.

The above shows how to construct a weighted undirected graph. The resulting weighted undirected graph may have high connectivities. Some edges of the graph are pruned using a threshold. That is, if a weight of an edge is below a given threshold, the edge will be deleted. After pruning edges, the *CDM* algorithm converts the weighted undirected graph into an undirected graph, and then finds maximal cliques using the approach given in (Bron and Kerbosch 1973). The found Web communities can be used to build a Web community directory (Pierrakos et al. 2003; Pierrakos et al. 2003). A similar Weblog based system is presented in (Otsuka et al. 2004).

7.2 Discovering Web Communities Using Co-occurrence

Murata proposed a way to create a Web community using *HITS* algorithm and enlarge the Web community using co-occurrence. Murata's algorithm starts with an initial set of Web pages, denoted C_I, which are taken as the initial centers (Murata 2000; Murata 2001), and enlarges C_I in the following two steps. First, the fans, C_F, are determined if they co-refer to all of the centers C_I. Here, the meaning that a Web page, p, co-refers to all the pages in C_I is that there are links from the page p to all the pages C_I. If the Web pages in C_I are important Web pages, there exists a large number of Web pages pointing to all the pages in C_I. Therefore, the set of C_F can be too large. Murata suggested to select the Web pages in C_F using an existing ranking.

The remaining task is to enlarge C_I as to find a larger set of centers, which is the goal of the proposed approaches. Second, in order to enlarge C_I, it extracts all the links (URLs) from all the fans – all the Web pages in C_F. Note: a link may appear many times in a single Web page and in all Web pages contained in C_F. All those links are sorted based on the frequency. In other words, the links that appear more will be ranked higher. The link ranked highest is considered as to be most close to the existing centers. Therefore, the top-ranked link will be added into C_I as an additional center. The above two steps will repeat to adding more centers into C_I until the number of fans that co-refer to all the enlarged C_I becomes small. We call this algorithm *Comm-Enlarge* which is outlined in Algorithm 7.1.

Algorithm 7.1 *Comm-Enlarge* (C_I, τ)

Input: a set of initial centers C_I, a threshold τ;
Output: a set of centers including C_I
1. Let C_F be the set of all Web pages that co-refer to all the pages in C_I;
2. **While** $|C_F| \geq \tau$
3. let L be a set of Web pages linked by all Web pages in C_F;
4. sort L based on the co-occurrence;
5. select the top-ranked Web page in L, and add it into C_I;
6. C_F be the set of all Web pages that co-refer to all the pages in C_I;
7. **return** C_I;

One question addressed further in (Murata 2000) is whether the newly added centers really contain the similar contents like those that are already in C_I. In other words, the question is how to ensure that the newly added link can be considered as a new center. In order to ensure the quality of centers, they take a pair-wise approach. For a given set of initial Web pages, C_I, of size n. It generates a set of initial inputs, $C_{I_1}, C_{I_2}, \cdots$, where each C_{I_j} contains two Web pages. In other words, for a size of n initial C_I, there are ${}_nC_2$ pairs. Each C_{I_k} can be enlarged individually by calling *Comm-Enlarge*(C_{I_k}, τ). Let C'_{I_k} be the enlarged set returned by *Comm-Enlarge*(C_{I_k}, τ). Murata proposed to rank those newly found centers by the number of occurrences. The top-ranked ones will be closed to most of centers in the initial C_I and will be regarded as new centers.

The above question implicitly asks whether there is a boundary between the related centers and unrelated centers. (Murata 2003) suggests using two sets of centers, a set of positive centers C_I, and a set of negative centers C_N. The positive centers are the centers a user wants to enlarge, whereas the negative centers are used as hints to assist finding the boundary. Intuitively, it hopes that enlarging C_I and C_N respectively will reach a point that both enlarged C_I and C_N share the common centers where the boundary exists.

The above approach is to explore Web communities based on link co-occurrence. Ohsawa et al.. studied Web communities using word co-occurrence structure in textual messages (Ohsawa et al. 2002).

7.3 Finding High-Level Web Communities

Most algorithms find Web communities from a Web graph where a Web page is a node and an edge is a link between two Web pages. However, a Web page may be considered as too primitive to represent a topic. In (Asano et al. 2003), Asano et al. studied how to find Web communities based on Web sites rather than Web pages, because Web sites can be more representative as a whole on certain topics. As a result, the nodes in a Web graph are Web sites and the edges links Web sites. The Web communities found on such a Web graph can give a general view on Web information.

Some Web sites are easy identified, such as Amazon, *Google*, etc, because these Web sites are hosted on a single Web server identified by an URL address. It is interesting to know that it is not easy to identify all Web sites, because there are many Internet service providers at which many companies and individual users can host their own Web sites. In other words, a single Web server can host many Web sites, and the locations/boundaries of such Web sites are hard to determine.

In (Asano et al. 2003), several heuristics were presented to identify Web sites in a Web server. Some Web servers are known to host Web sites at a certain level, for example, the third level from the root of the Web server. Many Web servers use a tilde (~) as default to indicate the root of a personal Web site. Asano et al. identify the boundaries of Web sites in a Web server based on connected components/cliques.

In the following, we introduce the multi-view Web graph (Asano et al. 2003), and its construction algorithm.

We start our discussion from a Web graph $G(V, E)$ where V is a set of Web pages and E is a set of links. A multi-view graph is defined on top of

$G(V, E)$ as follows. At level-1, $G(V, E)$ shows all the detailed structural information. At level-2, the structural information inside a Web site as a subgraph of $G(V, E)$ is ignored, and every Web site is viewed as a single node. There exists a link from a Web site A to a Web site B if there is a link from a Web page in A to a Web page in B. At level-3, the structural information inside a Web server, where many Web sites are hosted, is ignored, and every Web server is viewed as a single node. There exists a link from a Web server X to a Web server Y if there is a Web page in X points to a Web page in Y.

The algorithm presented in (Asano et al. 2003) is to construct a level-2 Web subgraph from the level-1 Web graph, based on a set of URLs as user input. After the level-2 Web subgraph has been constructed, any existing algorithm to find Web communities can be applied. The overview of the algorithm is shown in Algorithm 7.2. We call it *Find-WAW*, which is short for finding Web communities at Web site level. Algorithm 7.2 takes two parameters: a set of URLs, and a parameter, n, which controls the growth of the level-2 Web graph starting from the initial Web pages. In Algorithm 7.2, the level-2 Web graph is first initialized as an empty graph (line 1). In line 2-4, it initializes several variables needed for construct the level-2 Web graph. In brief, T is used to keep a set of graphs where each graph represents a Web site. N_v is used to keep a set of URLs for any newly found root of Web sites, and N_e is used to keep pairs of URLs for Web links. In line 5, it calls *Construct-L2-Graph* to construct the initial level-2 Web graph. Note: the first variable U is call-by-value, and the following variables are call-by-reference. In line 6-11, it expands the initial level-2 Web graph found in line 5 by n times. Every time, it expands the level-2 Web graph by adding neighbor Web sites of the found Web sites in G_2. Note: the newly found Web sites are kept in the variable N_v which is returned by the previous *Construct-L2-Graph*. In line 12, it finds Web communities using any of the algorithms we discussed before. In line 13, it returns the found Web communities.

Algorithm 7.2 *Find-WAW* (U, n)

Input: a set of initial Web pages U, and a parameter, n, to control the growth of level-2 Web graph;
Output: a set of Web communities found at level-2 Web graph $G_2(V_2, E_2)$;
1. let $G_2(V_2, E_2)$ be an empty level-2 Web graph;
2. let T be an empty set of working graphs;
3. let N_v be an empty set of URLs;
4. let N_e be an empty set of URL pairs as Web links;

5. *Construct-L2-Graph*(U, G_2, T, N_v, N_e);
6. **While** $n > 0$ **do**
7. $U \leftarrow N_v$;
8. let N_v be an empty set of URLs;
9. let N_e be an empty set of URL pairs as Web links;
10. *Construct-L2-Graph*(U, G_2, T, N_v, N_e);
11. $n \leftarrow n-1$;
12. find Web communities, W, on top of G_2 using an existing algorithm;
13. **return** W;

Below we briefly introduce the construction algorithm *Construct-L2-Graph*. First, it identifies the roots of Web sites using some heuristics based on the given set of URLs, U. The roots will be the nodes, V_2, of the level-2 Web graph, G_2. Second, starting from the root of each Web site, it constructs a subgraph, to represent the Web site, by traversing the level-1 Web graph following the breadth first order up to a limit. During the breadth first traversal, if it finds an URL which is not in the Web server where the Web site is hosted, it adds it into N_v (as a new potential neighbor of the Web site). N_v will be U as the input to *Construct-L2-Graph* in the next run to enlarge the level-2 Web graph. Also, if there are links from a Web page in a Web site in V_2 to a Web page in another Web site in V_2, the corresponding edges between the two Web sites will be added into the edge set, E_2, of the level-2 Web graph, G_2. Third, for those URLs in U that the simple heuristics cannot identify any Web sites, it will use some advanced heuristics to identify Web sites using cliques.

7.4 Web Community and Formal Concept Analysis

Formal concept analysis (FCA) is a formal approach to study conceptual structures among data sets (http://www.upriss.org/uk/fca). Rome and Haralick presented an approach to explore Web communities in (Rome and Haralick 2005), as the maximally complete directed bipartite subgraph, based on FCA, which we will discuss in this section. In the following, we brief introduce FCA, and then discuss Web community exploration based on FCA.

7.4.1 Formal Concept Analysis

In (Banter and Wille 1999), Ganter and Wille discussed the mathematical foundations of FCA. A data set is a set of objects with attributes that describe the properties of the data elements in the set. A *context* is a triple (G, M, I) where G is the domain of objects (the extent), and M is the domain of attributes (the intent), I is a binary relation such as $I \subseteq G \times M$. A *concept* in the context is a pair (C_G, C_M), where C_G $(\subseteq G)$ is a set of objects that have all the attributes in C_M $(\subseteq M)$. The binary relation I can represent the sub-concept/super-concept relationship existing among concepts. Let $C_i = (C_{G_i}, C_{M_i})$ and $C_j = (C_{G_j}, C_{M_j})$ be two concepts. C_i is a sub-concept of C_j (at the same time C_j is a super-concept of C_i), denoted $_i \leq \smile_j$, if $_{G_i} \subseteq \smile_{G_j}$ or equivalently, $C_{M_j} \subseteq C_{M_i}$. The ordered set of all concepts of the context (G, M, I) forms a *concept lattice*, denoted by $\underline{B}(G, M, I)$. The details of the mathematical foundations can be found in (Banter and Wille 1999).

7.4.2 From Concepts to Web Communities

Rome and Haralick in (Rome and Haralick 2005) considered a context (G, M, I) as a Web graph. Here, both G and M are Web pages, and I is a set of links that connect Web pages. Note: different from the original definition of the context, here, M is not a set of attributes but a set of objects like G. M and G may overlap. A concept $C = (C_G, C_M)$ is then considered as a complete directed bipartite subgraph where C_G is a set of fans and C_M is a set of centers. Therefore, a concept is considered as a Web community. There are many rather small Web communities, and Rome and Haralick in (Rome and Haralick 2005) considered to merge small Web communities as a large Web community by using the existing techniques to coalesce concepts (Haralick 1974; Funk et al. 1998).

Table 7.1. An Example

Object (x)	Relation (I(x))
1	2,6,9
2	10,11
3	2,6,9
4	6,9
5	2,6
6	10,12
7	2,6,9
8	10,12
9	10,11,12
10	1,3
11	1,3
12	1,3

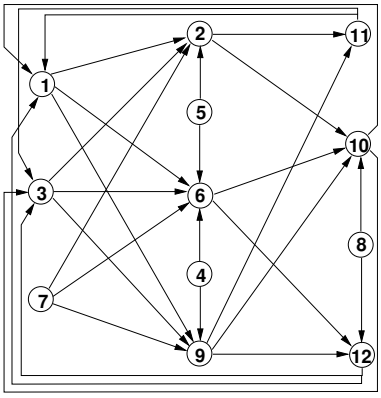


Fig. 7.1. The graph representation of Table 7.1

Table 7.2. The 9 complete directed bipartite graphs

Name	Fans (V_f)	Centers (V_c)
A	{1, 3 7}	{2, 6, 9}
B	{2, 9 }	{10, 11 }
C	{1, 3, 4, 7}	{6, 9}
D	{1, 3, 5, 7}	{2, 6}
E	{6, 8, 9}	{10, 12}
F	{9}	{10, 11, 12}
G	{10, 11, 12}	{1, 3}
H	{2, 6, 8, 9}	{10}
I	{1, 3, 4, 5, 7}	(Berry et al.)

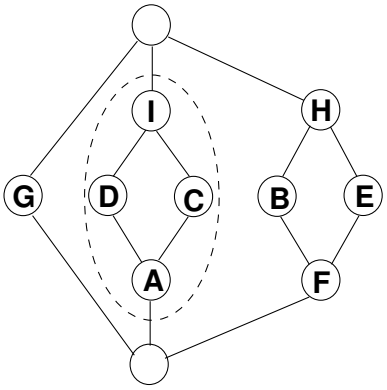


Fig. 7.2. The Concept Lattice

We use the same example in (Haralick 1974; Rome and Haralick 2005) to illustrate the main ideas. The example is shown Table 7.1 (Haralick 1974; Rome and Haralick 2005). In Table 7.1, an object, x, can be considered as a node in a graph, and the relation (x) shows that there is an edge from x to each object in the relation $I(x)$. The graph representation of Table 7.1 is shown in Fig. 7.1. Recall a complete directed bipartite graph $G_C(V, E)$ is defined in Definition 6.2. The set of nodes V consists of two disjoint sets, V_c (centers) and V_f (fans). Every node in V_f points every node in V_c. We use the (V_f, V_c) to represent a complete directed bipartite graph. There are 9 complete directed bipartite graphs in Fig. 7.1, as summarized in Table 7.2 (Haralick 1974; Rome and Haralick 2005).

The 9 complete directed bipartite graphs are 9 Web communities as subgraphs, and 9 concepts at the same time. The concept lattice on top of the 9 concepts is shown in Fig. 7.2 (Figure 4 in (Rome and Haralick 2005)). To form a large Web community from the small Web communities is to coalesce the corresponding small concepts.

The coalescing concept is to add information into the binary relation I to form a large concept. In terms of Web communities, it implies to add proper edges to make several small complete directed bipartite graphs as a large complete directed bipartite graph. As shown in (Haralick 1974; Rome and Haralick 2005), by adding 2 and 9 into the relation $I(4)$ and $I(5)$ in Table 7.1, or by adding edges $(4,2)$ and $(5,9)$ into the corresponding graph in Fig. 7.1, respectively, a large concept, or a large Web community, can be identified as shown in the dashed circle containing the four concepts I, D, C and

A in Fig. 7.2. The large Web community has a set of fans, $V_f = \{1,3,4,5,7\}$, and a set of centers, $V_c = \{2,6,9\}$. Why can we add more edges into the Web graph? Rome and Haralick justified the reason in (Rome and Haralick 2005): *I is prone to error, and this is especially true when dealing with the Web.* Rome and Haralick used a horizontal decomposition technique (Haralick 1974) to coalesce concepts, and reported their findings in (Rome and Haralick 2005).

7.5 Generating Web Graphs with Embedded Web Communities

In the previous sections, we introduced several algorithmic approaches for finding Web communities. In (Tawde et al. 2004), Tawde et al. indicated a great concern: how to build a model that can capture both the global structure of Web properties as the entire graph and the local Web communities as subgraphs. They presented a three step approach on generating Web graphs with embedded communities. In the first step, they construct Web communities using the preferential and random link distribution (Pennock et al. 2002). In the second step, they construct combined Web communities by taking the interaction among the found communities into consideration (Chakrabarti et al. 2002). In the last step, they reexamine the found communities in the Web data. In the following, we introduce the first step in generating a Web community.

The Web community is considered as in Definition 7.1 (Tawde et al. 2004). Unlike the others, it attempts to describe a Web community using link distribution, rather than as bipartite graphs or network flows.

Definition 7.1. *A Web community* is a set of n nodes whose link distributions can be described by two parameters, α_i and α_o, for inlink and outlink distributions.

The idea of using link distribution comes from the network growth model (Pennock et al. 2002), in which, Pennock observed that there exist qualitatively and considerably less biased link distributions. The network growth model consists of two main components, the preferential attachment and the uniform random attachment. The preferential attachment explains the network growth by the observation that a popular Website becomes much more popular, which leads to power law distribution (Mitzenmacher 2004). The uniform random attachment considers an unpopular Website that can be linked by us-

ers due to the users personal interest and has the possibility to grow. Consider a Web community, and let k_i be the current number of edges incident on node v_i. In (Pennock et al. 2002), the probability that an edge connects to node v_i is specified as $\Pi(k_i)$.

$$\Pi(k_i) = \alpha \cdot \frac{k_i}{2mt} + (1-\alpha) \cdot \frac{1}{n_0 + t} \tag{7.1}$$

Here, t is the number of nodes added to the initial number of nodes in the Web community, and $n_0 + t$ is the total number of nodes in question. Let $2m$ be the average number of edges per node, $2mt$ shows a popular case such as a node is added with $2m$ edges. Eq. (7.1) uses α for preferential attachment, and $1-\alpha$ for uniform random attachment.

Tawde's algorithm for generating link distribution is given below. It starts from an empty Web community V_L. At every t step, a node together with p outlinks and q inlinks is added into G_L. This step will repeat n times. At the end of the n steps, V_L consists of two disjoint and equal-size subsets V_{L_o} and V_{L_i} such as $|V_{L_o}| = |V_{L_i}| = n$. Every node in V_{L_o} has many outlinks but the destinations of the edges have not been decided. Every node in V_{L_i} has many inlinks but the sources of the edges have not been decided. The probability for a node, v_i, to be selected to connect an edge is based on Eq. (7.1). Specifically, when an inlink is connected to the selected node, v_i, a source node needs to be selected. When an outlink is connected from the selected node, v_i, a destination node needs to be selected. The probabilities of selecting a source and a destination node are specified by Eq. (7.2) and Eq. (7.3), respectively.

$$\Pi(p_i) = \alpha \cdot \frac{p_i}{pt} + (1-\alpha_o) \cdot \frac{1}{n_0 + t} \tag{7.2}$$

$$\Pi(q_i) = \alpha \cdot \frac{q_i}{} + (1-\alpha_i) \cdot \frac{1}{_0 +} \tag{7.3}$$

Here, α_i and α_o control preferential attachment for inlinks and outlinks.

The inlinks and outlinks are generated independently. To connect nodes in V_{L_o} and V_{L_i}, Tawde et al. use an $n \times n$ matrix, denoted M. The matrix $M[i, j]$ can be built based on page connections from the component i to the component j (Tawde et al. 2004).

7.6 Modeling Web Communities Using Graph Grammars

In this section, we introduce another approach that attempts to model Web communities based on the Web topological, rather than to find Web communities among Web pages (Frivolt and Bielikova 2005). Arivolt and Bieliková explored the issue that, if Web communities exist, can we find a way to generate (model) all such possible Web communities? They used a term rewriting system, $Gr(R, \sigma)$, that can generate graphs based on an initial set of graphs, σ, and a set of rewriting rules (or sometimes called production rules), R. A rewriting rule, $L \rightarrow R$ specifies how a node on the left, L, to be replaced by nodes in the right site, R. The definition of such a rewriting rule is given below.

Definition 7.2. (Definition 2 in (Frivolt and Bielikova 2005)) A *rewriting rule* is in the form as follows:

$$v \rightarrow \{(v_1, \mu_1, p_1), (v_2, \mu_2, p_2), \cdots\}, E_\eta .$$

Here, $p_i \in [0,1]$ *is the probability of mapping the node* v *on the left to a node* v_i *on the right.* $\mu_i \in [0,1]$ *is the probability of overtaking an incident edge to the node* v *and* v_i. *And,* E_η *is a subset of edges such as*

$$E_\eta \subset \{(v_o, v_i, p_{oi}) \mid v_o, v_i \in \{v_1, v_2, \cdots\}\},$$

where p_{oi} *is the probability of generating an edge from* v_o *to* v_i.

The idea behind Definition 7.2 is to expand a node on the left side into a graph on the right side. The graph on the right consists of a set of nodes, v_i. E_η shows the probability of connections among the nodes in the graph to be connected. The two parameter, $p_i \in [0,1]$ and $\mu_i \in [0,1]$, control how to map the node *v* onto the node(s) on the right and how to connect the links. Arivolt and Bieliková showed three rewriting rules to generate hierarchies, bipartite graphs and cliques, and used a graph generating L-systems to expand graphs in (Frivolt and Bielikova 2005).

The expanding process is discussed below in brief. Based on $Gr(R,\sigma)$, a graph $G(V,E)$, will be expanded by applying a rewriting rule, r, in R, and become another graph $G'(V',E')$. In the new graph, the set of nodes, V', and set of edges, E', are formed as follows. V' will exclude the node v, which appears at the left side of the rewriting rule, r, from the set of nodes V, but include nodes, $v_1, v_2, \cdots$, based on the corresponding probability, p_1 p_2 $\cdots$, associated with them, respectively. E' will include all the edges in E that are not incident to v, which appear at the left side of the rewriting rule, r. In E', the edges from the nodes $V - v$ to the newly added nodes are included, based on the edges in E that are incident to the node v, and the probability of μ_i. Also, E' will also include new edges among the newly added nodes based on E_η. The process will repeat to expand graphs.

7.7 Geographical Scopes of Web Resources

In this section, we introduce a novel approach to identify the geographical scopes of Web resources (Ding et al. 2000). The approach proposed in (Ding et al. 2000) is a general approach. This approach can help us to identify the geographical scope of a center in a Web community. Intuitively, there exists a geographical scope for a Web resource. For example, consider a newspaper. There are national newspapers, local newspapers, and newspapers for special interest groups. Suppose that there are Web communities on restaurants. When users want to find a Web site for restaurants in Hong Kong, they may consider which Web site they should visit first. All the existing algorithms like *HITS* or *PageRank* algorithms are not designed to answer the question on the geographical scopes of Web resources. Ding et al. defined a geographical scope of a Web resource as follows.

Definition 7.3. (Definition 1 of (Ding et al. 2000)) The geographical scope of a Web resource w is the geographical area that the creator of w intends to reach.

As can be seen from Definition 7.3, the geographical scope is a subjective matter. Ding et al. assume that there is a predefined hierarchy, L for geographical scopes. In the hierarchy, a node $l \in L$ represents a geographical scope. The root represents the whole geographical scope. A node in the bottom level of the hierarchy represents a primitive geographical scope. In a certain level i, a node represents a geographical scope which can be divided into

several disjoint geographical sub-scopes represented by the nodes in the next level $i+1$, given the geographical scope. The problem of finding the geographical scope becomes a problem of determining a proper node for a Web resource in the predefined geographical hierarchy. The determination of a geographical scope $l \in L$ for a given Web resource w is based on two conditions, *fraction* and *uniformity*. The condition of fraction states that there are significant Web pages in a geographical scope l containing links to w. The condition of uniformity states that the links to w are distributed *smoothly* across the geographical scope l. Several methods were studied in (Ding et al. 2000) to estimate the geographical scope of a Web resource.

7.7.1 Two Conditions: Fraction and Uniformity

The fraction and uniformity of a Web resource w in a geographical scope l are computed as follows.

First, if a Web resource, w, belongs to a graphical scope l, then a significant number of Web pages in l shall point to w. It is computed as a function called $Power$.

$$Power(w,l) = \frac{Links(w,l)}{Pages(l)} \tag{7.4}$$

Here, the function $Links(w,l)$ returns the number of Web pages in the geographical scope l that link to w, and the function $Pages(l)$ returns the total number of Web pages in the geographical scope l. As seen above, the fraction is dependent on the Web pages in the geographical scope, l, including any page exists in the scope l or any of its sub-scopes. Because it is difficult to obtain all Web pages in any certain geographical scope, it requests to select Web pages based on domain knowledge or some heuristics.

When the function $Power$ returns a high rate for a Web resource w in a geographical scope l, it does not necessarily mean that w should belong to l. Consider that the geographical scope l has k sub-scopes, $l_1, l_2, \cdots, l_k$. It can be the case that all Web pages linking to w reside in one sub-scope l_i. Therefore, the geographical scope of w should be l_i or one of l_i's sub-scope. In order to ensure the proper geographical scope for w, uniformity is introduced to measure the notion of smooth across all sub-scopes, $l_1, l_2, \cdots, l_k$ for a Web resource w. In other words, for a Web resource w to be in a geo-

graphical scope l, (, $_i$) must be all high in all sub-scopes l_i, for $i = 1$ to k. Ding et al. proposed a function $Spread(w,l)$ to measure the uniformity, and three methods to compute it.

Let the scope l have k sub-scopes, $l_1, l_2, \cdots, l_k$. All l_i are child nodes of l in the geographical hierarchy L. Let $p_i = Pages(l_i)$, $q_i = Links\ w, l_i$, $r_i = Power(w,l_i)$. We show the three methods to compute $Spread$, namely, a method based on information retrieval (Eq. (7.5)), a method based on information theory (Eq. (7.6)), and a method based on relative-error (Eq. (7.7)), as given in (Ding et al. 2000).

$$Spread(w,l) = \frac{\sum_{i=1}^{k} p_i \times q_i}{\sqrt{\sum_{i=1}^{k} p_i^2} \times \sqrt{\sum_{i=1}^{k} q_i^2}} \tag{7.5}$$

$$Spread(w,l) = \frac{-\sum_{i=1}^{k} \frac{r_i}{\sum_{j=1}^{k} r_j} \cdot \log(\frac{r_i}{\sum_{j=1}^{k} r_j})}{\log k} \tag{7.6}$$

$$Spread(w,l) = \frac{1}{1 + \frac{1}{\sum_{i=1}^{k} p_i} \sum_{i=1}^{k} p_i \cdot \frac{R - r_i}{R}} \tag{7.7}$$

where $R = (\sum_{i=1}^{k} q_i)/(\sum_{i=1}^{k} p_i)$.

Eq. (7.5) computes uniformity using the similarity as to compute the cosine angle between two vectors, a $Pages$-vector containing all p_i and a $Links$-vector containing all q_i. Eq. (7.6) computes the uniformity using entropy. The idea behind Eq. (7.6) is the maximum entropy can be obtained while r_i ($Power(w,l_i)$) is uniform across all l_i sub-scopes. Eq. (7.7) computes uniformity by measuring the relative error with a target value R.

Algorithm 7.3 *Estimate-Geographical-Scope*(L, w, τ_c, τ_e)

Input: a geographical hierarchy L, a Web resource w, and two thresholds τ_c and τ_e;
Output: a set of geographical scopes;
1. $L_C \leftarrow \varnothing$;
2. $L_w \leftarrow \varnothing$;

3. add $l \in L$ into L_C following a top-down traversal order over L if $Spread(w,l) \geq \tau_c$ and $Spread(w,l') < \tau_c$ for any ancestors of l;
4. **for** $l \in L_C$ **do**
5. add l into L_w if $Power(w,l) \geq \tau_e$
6. **return** L_w.

7.7.2 Geographical Scope Estimation

The algorithm for estimating geographical scope for a Web resource w is outlined in Algorithm 7.3. We call it the *Estimate-Geographical-Scope*, which takes four parameters, the predefined geographical hierarchy L over which the scopes of the Web resource w in question will be determined, and two thresholds τ_c and τ_e. In line (1-2), it initializes two sets, L_C and L_E. The former is used to keep a set of scope candidates, and the latter is used to keep a set of identified scopes for w to be returned. In line 3, it traverses the geographical hierarchy L in a top-down fashion. If it finds a scope l such as $Spread(w,l) \geq \tau_c$, it will add the scope l into L_C, and will not continue to traverse any descendents of l. It will visit l's child nodes $l_1, \cdots$, if $Spread(w,l) < \tau_c$. Note, if $Power(w,l) = 0$, there is no need to visit l's child nodes. In line 4-5, it will further prune scopes from the set of candidates, based on $Power(w,l)$. Other forms of pruning strategies were also given in (Ding et al. 2000), for example, to find top-k scopes following the $Power(w,l)$.

7.8 Discovering Unexpected Information from Competitors

Web communities include fans and centers. Centers are authoritative on certain topics, and are seen as competitors sometimes. In (Liu et al. 2001), Liu, Ma and Yu proposed approaches to discover unexpected information from the competitor's Web sites. The proposed approaches help us to further analyze the differences between centers in a Web community. In the general, the problem setting is as follows. Given two Web sites, W_b and W_c, where W_c is the Web site for the competitor, and W_b is the Web site, as the basis, to find the unexpected information from W_c. The unexpected information in W_c is

the information which is relevant to W_b but is unknown. Five methods were proposed to compare W_b with W_c in (Liu et al. 2001).

We introduce the five methods below. Suppose that the Web site W_b and W_c consist of sets of Web pages, $V_b = \{b_1, b_2, \cdots b_n\}$ and $V_c = \{c_1, c_2 \cdots, c_m\}$, respectively. Consider the running example in (Liu et al. 2001) below.

Example 7.4. Suppose there are four Web pages at the Web site W_b, $V_b = \{b_1, b_2, b_3, b_4\}$ *and there are three Web pages at the Web site* W_c, $V_c = \{c_1, c_2, c_3\}$. *The terms and their frequencies in the Web pages are shown as pairs below.*

$b_1 = \{(data,1),(predict,1)\}$,
$b_2 = \{(information,2),(extraction,1),(data,2)\}$,
$b_3 = \{(classify,2),(probability,2)\}$,
$b_4 = \{(cluster,2),(segment,1)\}$,
$c_1 = \{(data,2),(predict,2),(classify,3)\}$,
$c_2 = \{(association,3),(mine,2),(rule,1)\}$, and
$c_3 = \{(cluster,3),(segment,2),(data,2)\}$.

The first method is to find the correspondences between Web pages V_c and V_b, based on a similarity measure. Consider Example 7.4. The Web page b_1 has two corresponding Web pages, c_1 and c_3. The Web page c_1 is ranked top, because the similarity of b_1 to c_1 is higher than any other c_i. The similarity between b_1 and c_2 is zero, because there are no common terms used in both Web pages.

The second method is to find unexpected terms in a Web page, c_i, to a Web page, b_j, provided that c_i and b_j are the corresponding Web pages with high similarity. The unexpectedness is computed using a function $unexpT$ for a term k.

$$unexpT(k, c_i, b_j) = \begin{cases} 1 - tf_{k,b_j}/tf_{k,c_i} & \text{if } tf_{k,b_j}/tf_{k,c_i} \le 1 \\ \end{cases}, \tag{7.8}$$

where tf_{k,d_j} is the term frequency of the term k in the document (Web page) d_j. The term k is unexpected in c_i, from b_j's viewpoint, if $tf_{k,b_j}/tf_{k,c_i}$ is small. Consider b_1 and c_1, which are similar because they share two terms, `data` and `predict`. The term `classify` is in c_1 but not in b_1. The unexpectedness of the term of classify using the function $unexpT$ is 1.

The above two methods are used to find unexpected information based on terms. The third method is to find if a Web page at the Web site W_c is unexpected. In doing so, the authors in (Liu et al. 2001) consider all the Web pages in a Web site as a single document, and compute the unexpectedness of a term k on the Web site basis, other than the Web page basis. Let D_b and D_c be two sets of terms containing all the terms at the Web sites W_b and W_c, respectively. The unexpectedness of the term k, on the Web site basis, is $unexpT(k, D_c, D_b)$. Suppose c_i contains K terms $\{t_1, t_2, \cdots, t_K\}$. The unexpectedness of a Web page c_i at the Web site W_c is computed as follows:

$$unexpP(c_i) = \frac{\sum_{i=1}^{K} unexpT(k_i, D_c, D_u)}{K}. \tag{7.9}$$

In Example 7.4, c_2 is ranked as the top unexpected Web page because none of the terms in c_2 are expected from the Web site W_b.

In the third method, the unexpectedness of a Web page c_i is computed based on individual terms appearing in the Web page. The fourth method is to find unexpected Web pages based on sets of terms. Liu et al. used association rule mining techniques to identify sets of terms, which appear frequently in a page, and take sets of terms instead terms to compute the unexpectedness of a Web page. In doing so, they consider a sentence as a set of terms, and a document as a set of such sentences. A set of terms is frequent if they together appear in sentences more than a given threshold in a document. After finding frequent sets of terms in individual Web pages, the unions of all such frequent sets of terms, associated with total frequencies, at Web site W_b and W_c can be obtained, denoted S_b and S_c, respectively. Both S_b and S_c serve the same role as D_b and D_c used in the third method. Suppose a Web page c_i

contains L frequent sets of terms, $\{ _1, _2, \cdots, _L\}$. The unexpectedness of the Web page c_i in terms of frequent sets of terms is computed as follows:

$$unexpS(c_i) = \frac{\sum_{i=1}^{L} unexpS(l_i, S_c, S_u)}{L}, \tag{7.10}$$

where $unexpS$ can be computed in a similar way like $unexpT$ by treating a set of terms as a virtual term.

As the last of the five methods, in (Liu et al. 2001), the authors indicated to use out-going links from both Web sites, W_b and W_c to find unexpected information. With a similar mechanism like crawlers, the outgoing links from W_b and W_c can be determined up to a certain level. The differences among the link structures can give an indicator about the unexpectedness of links.

7.9 Probabilistic Latent Semantic Analysis Approach

Recently, Web usage mining is addressed for Web community analysis. Basically, there are two kinds of clustering methods in the context of Web usage mining, which are associated with the objects of performing: user session clustering and Web page clustering (Mobasher 2004). One successful application of Web page clustering is adaptive Web site. For example, an algorithm called *PageGather* (Perkowitz and Etzioni 1998) is proposed to synthesize index pages that are not existing initially, based on finding Web page segment sharing common semantic similarities. The generated index pages are conceptually representing the various access tasks of Web users. Mobasher et al. (Mobasher et al. 2002) utilize Web user session and page clustering techniques to characterize user access pattern for Web personalization based on Web usage mining. Generally, Web page clusters can be resulted from applying clustering process on the transpose of the session-page matrix. However, the conventional clustering techniques such as distance-based similarity methods are not capable of tackling this type high-dimensional matrix. This is mainly because that there is usually tens to hundreds of thousands sessions in Web log files. Consequently, the high computational difficulty will be incurred when we utilize sessions as dimensions rather than pages, on which we will employ clustering technique. An alternative approach for Web page clustering is proposed to overcome this type clustering by (Han et al. 1998). This so-called *Association Rule Hypergraph Partitioning* (ARHP) exploits associate rule mining and graph-based technique to classify Web pages into a set of clusters efficiently.

7.9.1 Usage Data and the PLSA Model

In general, the user access interests exhibited may be reflected by the varying degree of visits in different Web pages during one session. Thus, we can represent a user session as a weighted page vector visited by users during a period. After data preprocessing, we can built up a page set of size n as $P=\{p_1,p_2,\ldots,p_n\}$ and user session set of size m as $S=\{s_1,s_2,\ldots,s_m\}$. The whole procedures are called page identification and user sessionization respectively. By simplifying user session in the form of page vector, each session can be considered as an n-dimensional page vector $s_i=\{a_{i1},a_{i2}\ldots,a_{in}\}$, where a_{ij} denotes the weight for page p_j in s_i user session.
As a result, the user session data can be generated to form Web usage data represented by a session-page matrix $SP_{m\times n}=\{a_{ij}\}$. The entry in the session-page matrix, a_{ij}, is the weight associated with the page p_j in the user session s_i, which is usually determined by the number of hit or the amount time spent on the specific page.

The PLSA model is based on a statistic model called the aspect model, which can be utilized to identify the hidden semantic relationships among general co-occurrence activities. Similarly, we can conceptually view the user sessions over Web pages space as co-occurrence activities in the context of Web usage mining to discover the latent usage pattern. For the given aspect model, suppose that there is a latent factor space $Z=\{z_1,z_2,\ldots,z_l\}$ and each co-occurrence observation data (s_i, p_j) is associated with the factor $z_k \in Z$ by varying degree to z_k.

According to the viewpoint of aspect model, thus, it can be inferred that there are existing different relationships among Web users or pages related to different factors, and the factors can be considered to represent the user access pattern. For example, for an academic Website, we can predefine that there exist k latent factors associated with k navigational behavior patterns, such as z_1 standing for admission applying of international students, z_2 for particular interests on postgraduate programs, and z_3, z_4 … etc,. In this manner, each usage data (s_i, p_j) can convey the user navigational interests by mapping the observation data into the k-dimensional latent factor space. The degrees, to which such relationships are "explained" by each factor, are derived from the factor-conditional probabilities. In (Xu et al. 2005), authors adopt PLSA model to model the relationships among Web pages and reveal latent semantic factors as well.

Firstly, let us introduce the following probability definitions:

- $P(s_i)$ denotes the probability that a particular user session s_i will be observed in the occurrences data,

- $P(z_k|s_i)$ denotes a *user* session-specific probability distribution on the unobserved class factor z_k explained above,
- $P(p_j|z_k)$ denotes the class-conditional probability distribution of pages over the latent variable z_k.

By combining probability definition and Bayesian formula, we can model the probability of an observation data (s_i, p_j) by adopting the latent factor variable z_k as:

$$P(s_i, p_j) = \sum_{z_k \in Z} P(z_k) \bullet P(s_i \mid z_k) \bullet P(P_j \mid z_k) \tag{7.11}$$

.

Furthermore, the total likelihood of the observation is determined as:

$$L_i = \sum_{s_i \in S, p_j \in P} m(s_i, p_j) \bullet \log P(s_i, p_j) \tag{7.12}$$

,

where $m(s_i,p_j)$ is the element of the session-page matrix corresponding to session s_i and page p_j.

In order to maximize the total likelihood, we make use of *Expectation Maximization* (*EM*) algorithm to perform maximum likelihood estimation in latent variable model (Dempster et al. 1977). Generally, two steps are needed to implement in this algorithm alternately: the (1) Expectation (E) step, where posterior probabilities are calculated for the latent factors based on the current estimates of conditional probability; and the (2) Maximization (M) step, where the estimated conditional probabilities are updated and used to maximize the likelihood based on the posterior probabilities computed in the previous E-step.

We describe the whole procedure in detail:

1. Firstly, given the randomized initial values of $P(z_k)$, $P(s_i|z_k)$, $P(p_j|z_k)$
2. Then, in the E-step, we can simply apply Bayesian formula to generate following variable based on usage observation:

$$P(z_k \mid s_i, p_j) = \frac{P(z_k) \bullet P(s_i \mid z_k) \bullet P(p_j \mid z_k)}{\sum_{z_k \in Z} P(z_k) \bullet P(s_i \mid z_k) \bullet P(p_j \mid z_k)} \tag{7.13}$$

..

(1) Furthermore, in M-step, we can compute:

$$P(p_j \mid z_k) = \frac{\sum_{s_i \in S} m(s_i, p_j) \bullet P(z_k \mid s_i, p_j)}{\sum_{s_i \in S, p_j \in P} m(s_i, p_j^{'}) \bullet P(z_k \mid s_i, p_j^{'})}, \tag{7.14}$$

$$P(s_i \mid z_k) = \frac{\sum_{p_{j\in} \in P} m(s_i, p_j) \bullet P(z_k \mid s_i, p_j)}{\sum_{s_i^{'} \in S, p_j \in P} m(s_i^{'}, p_j) \bullet P(z_k \mid s_i^{'}, p_j)}, \tag{7.15}$$

$$P(z_k) = \frac{1}{R} \sum_{s_i \in S, p_j \in P} m(s_i, p_j) \bullet P(z_k \mid s_i, p_j), \tag{7.16}$$

where

$$R = \sum_{s_i \in S, p_j \in P} m(s_{i,}, p_j) \tag{7.17}$$

Basically, substituting equation (7.14)-(7.16) into (7.11) and (7.12) will result in the monotonically increasing of total likelihood L_i of the observation data. The executing of E-step and M-step is repeating until L_i is converging to a local optimal limit, which means the estimated results can represent the final probabilities of observation data.

It is easily found that the computational complexity of this algorithm is *O(mnk)*, where *m* is the number of user session, *n* is the number of page, and *k* is the number of factors.

7.9.2 Discovering Usage-Based Web Page Categories

As discussed in last section, note that each latent factor z_k do really represent specific aspect associated with co-occurrence in nature. In other words, for each factor, the degrees related to the co-occurrence are expressed by the factor-based probability estimates. From this viewing point, we, thus, can utilize the class-conditional probability estimates generated by the PLSA model and clustering algorithm to partition Web pages into various usage-based groups.

Note that the set of $P(z_k|p_j)$ is conceptually representing the probability distribution over the latent factor space for a specific Web page p_j, we, thus, construct the page-factor matrix based on the calculated probability estimates, to reflect the relationship between Web pages and latent factors, which is expressed as follows:

$$vp_j = (c_{j,1}, c_{j,2}, ..., c_{j,k}) \tag{7.18}$$

Where $c_{j,s}$ is the occurrence probability of page p_j on factor z_s. In this way, the distance between two page vectors may reflect the functionality similarity exhibited by them. We, therefore, define their similarity by applying well-known cosine similarity as:

$$sim(p_i, p_j) = (vp_i, vp_j) / (\|vp_i\|_2 \bullet \|vp_j\|_2), \tag{7.19}$$

where $(vp_i, vp_j) = \sum_{m=1}^{k} c_{i,m} c_{j,m}$, $\|vp_i\|_2 = \left(\sum_{l=1}^{k} C_{i,l}^2 \right)^{1/2}$

With the page similarity measurement (7.19), we propose a modified *k*-means clustering algorithm to partition Web pages into corresponding groups. The detail of the clustering algorithm is described as follows:

Algorithm 7.4 *Web Page Grouping*

Input: the set of $P(z_k|p_j)$, predefined threshold μ
Output: A set of Web page groups $PCL = (PCL_1, PCL_2, \ldots, PCL_P)$

1. Select the first page p_1 as the initial cluster PCL_1 and the centroid of this cluster: $PCL_1=\{p_1\}$ and $Cid_1=p_1$.
2. For each page pi, measure the similarity between pi and the centroid of each existing cluster $sim(p_j, Cid_j)$
3. If $sim(p_i, Cid_t) \quad \max_j(sim(p_i, Cid_j)) \quad \mu$, then insert p_i into the cluster PCL_t and update the centroid of PCL_t as

$$Cid_t = 1/|PCL_t| \bullet \sum_{j \in PCL_t} vp_j \tag{7.20}$$

   where $|PCL_t|$ is the number of pages in the cluster; Otherwise, p_i will create a new cluster itself and is the centroid of the new cluster.
4. If there are still pages to be classified into one of existing clusters or a page that itself is a cluster, go back to step 2 iteratively until it converges (i.e. all clusters' centroid are no longer changed)
5. Output $PCL=\{PCL_P\}$

Similarly, the probabilistic distribution over the factor space of a user $P(z_k|s_i)$ can reflect the specific user's access tendency over the whole latent factor space, in turn, may be utilized to uncover usage pattern. With this method, (Xu et al. 2005) reveals the user access patterns.

8 Conclusions

8.1 Summary

The World Wide Web has become very popular recently and brought us a powerful platform to disseminate and retrieve information as well as conduct business. Nowadays the Web has been well known as a large data repository consisting of a variety of data types and users are facing the problems of information overload and drowning due to the significant and rapid growth in amount of information and the number of users. Moreover, Web users usually suffer from the difficulties of finding desirable and accurate information due to two problems: low precision and low recall caused by above the reasons. For example, if a user wants to search desired information by utilizing a search engine such as *Google*, the search engine will provide not only Web content related to the query topic, but also a large mount of irrelevant information. It is hard for users to obtain their exactly needed information. Thus, the emerging of Web has put forward a great deal of challenges to Web researchers for Web-based information management and Web service management.

This book starts with introduction of some preliminary background knowledge for better understanding of the succeeding chapters. Some matrix concepts and theories commonly used in matrix-based analysis are presented accordingly, such as matrix eigenvalue, eigenvector; norm, singular value decomposition (SVD) of matrix as well as similarity measure of two vectors. In addition, graph theory basics and Markov chain. Then, this book presents materials to show how to find useful information from Web at two levels according to users' interest, namely the individual Web pages and the primitive structural behavior of Web pages.

The former is covered in Chap. 3 and Chap. 4, in which we present material from different viewpoints on two well-known algorithms, HITS and PageRank. Chap. 3 discusses hyperlink-based algorithm, its improvements, variations, and related issues. Some in-depth analyses of HITS are presented as well. We give the original algorithm of HITS, stability issues of HITS as well as randomized, subspace and weighted HITS. Especially other HITS-related algorithms and in-depth analysis of HITS are discussed to reveal some features

of HITS. Chap. 4 gives the original PageRank algorithm based on hyperlink information. Moreover, the algorithms, such as that probabilistically combines hyperlink and content information for page ranking, and accelerate PageRank computation, are also proposed. Such materials aim to provide in-depth understanding with regarding to PageRank algorithm and its variants.

The latter is covered in Chap. 5, Chap. 6 and Chap. 7. In Chap. 5, we present materials on some Web community analysis approaches, particularly, we introduce affinity and co-citation such kinds of matrix-based methods to cluster Web pages. In this chapter, the concept of similarity between two pages is proposed to measure the relevance distance of them, meanwhile, several similarity-based algorithms are presented to discover the mutual relationships amongst the Web pages. In Chap. 6 and Chapter 7, we present materials on constructing and analyzing various Web communities. In Chap. 6, we introduce Web communities as complete directed bipartite graphs the notion of small world, which indicates that such subgraphs do exist on Web. Algorithms for finding such complete directed bipartite graphs and finding Web communities in arbitrary shapes are discussed sequentially. Besides, we also discuss the algorithms on finding connections among Web communities and exploring the Web community evolution patterns. Especially, a graph-theoretical approach is introduced, which tries to answer the question when a found Web community can be determined as unique.

Chap. 7 introduces techniques that explore Web communities based on user access patterns from Web logs with Web usage mining techniques. We discuss how to use co-occurrences to enlarge Web communities and introduce Web communities from a high-level Web graph where a node is a Web site rather than a Web page. Three approaches to model Web communities are presented. Finally, we introduce how to estimate the geographical scope of a Web site and how to discover the unexpected information from other authoritative Web sites as centers in Web communities.

Informally, both Web clustering and Web communities finding are developed to find information at a level, which is higher than the level of individual Web pages. As indicated in Chap. 5, Web clustering, as one approach, can assist users in finding Web communities, because the Web pages found in a single cluster share a high similarity. Web communities discussed in Chap. 6 and Chap. 7, on the other hand, take different approaches. We discuss the differences between the two main approaches. First, Web clustering is developed on similarity measures. The Web clustering is able to distinguish hubs and authorities, and to distinguish fans and centers. On the other hand, such fans and centers are the two most important issues in Web communities to be distinguished. Second, Web clustering is a computationally intensive task. The huge amount of Web pages and the speed of information growth on the Web make it difficult to find all Web clusters at different granularities and to

evaluate the Web clusters found, even though it uses heuristics to achieve sub-optimal solutions. The algorithms developed for finding Web communities show the attempts to find some primitive structural behaviors in Web, such as complete directed bipartite graphs or dense directed bipartite graphs. Finding such Web communities is still computational intensive, but is considered with less overhead. Next, we will outline some future research directions in the area of Web communities, because it is more important to find structural behavior of Web pages.

8.2 Future Directions

The first issue is related to what we can additionally provide to users on top of the Web communities and Web clusters found. We believe that one of the most important areas is the applications for recommendation and personalization design. In order to make it effective for recommendation and/or personalization, in addition to the hubs (fans) and authorities (centers) found in Web communities, users views or users access patterns need to be taken into consideration entirely, where a user can be a Web designer or an end Web user. The second issue is how we can build up recommendation systems for Web users based on all we can possibly obtain including Web clusters, Web communities, as well as users viewpoints. In the current Web recommendation and Web mining areas, there are two intrinsic problems. One is the lack of effective approaches to incorporate both the designer's view and user's view regarding the Web documents/information. The other is the lack of systematic or automatic ways to construct user profiles and to consistently build up relationships among users and documents. Technical innovation and originality lies in the development of new data structures and data model to incorporate both designers and users' views on Web information and services, and then develop an effective approach to clustering Web information and user sessions for recommendation and personalization design.

The following research questions need to be thoroughly studied and addressed in the future:

- Q1. What model is suitable for capturing the relationships between Web documents and Web usage information?
- Q2. How do we discover Web communities and user communities based on such a model?
- Q3. How do we make recommendations based on the Web communities and user communities being found?
- Q4. How do we automatically build user profiles for a large number of users?

– Q5. How do we make online recommendations?

The key issue behind the above questions is to find relationships between the users and Web communities. There are several components, which make it challenging. First, user access patterns are different and can dynamically change from time to time. In information retrieval or database queries, there are 80-20 rules, which mean 80% users access 20% of information. As people found in user access patterns in *Google*, such rules may not exist due to the scale of information available and the large number of users from all different backgrounds globally. Second, Web communities are still the primitive structures to be studied. Currently, the algorithms found are more or less interested in the primitive structural behavior of Web pages rather than the ways users use them. There are a huge number of Web communities. The relationships among Web communities and the ways of the links across Web communities need more investigations. Third, the bridge between the two gulfs, users and Web communities, needs to be identified. One such a common basis can be the content topics, because Web communities are built for certain topics and users are interested in certain topics. Some existing work attempted to use reinforcement to cluster users based on their access patterns to Web pages via pre-given topics, and to cluster Web pages based on the access patterns users access them via pre-given topics simultaneously. Due to the diversity of Web communities and user access patterns, we believe that the future work along this direction will lead better understanding of the relationships between users and Web communities. Fourth, time plays an important role. Web communities may emerge, merge, split and disappear and user interest may change from time to time and, thus, the evolving patterns and the ways the Web communities reconstruct themselves are unclear.

The exploration of research experience and techniques obtained from various disciplines, such as database, information retrieval, Web community and data mining, Web mining especially Web usage mining, is needed to discover the underlying relationships among Web communities or Web users for Web recommendation and Web personalization.

References

The Open Directory Project: Web directory for over 2.5 million URLs, http://www.dmoz.org.

Abiteboul, S. (1997). Querying Semi-Structured Data. *Proceedings of ICDT*, Delphi, Greece.

Adamic, L. A. (1999). The Small World Web. *Proc. of the 3rd European Conference on Research and Advanced Technology for Digital Libraries (ECDL'99)*.

Arasu, A., Cho J, Garcia-Molina H, Paepcke A, Raghavan S (2000). Searching the Web, Computer Science Department, Stanford University.

Asano, Y., H. Imai, M. Toyoda and M. Kitsuregawa (2003). Finding Neighbor Communities in the Web Using Inter-site Graph. *Proc. of the 14th International Conference on Database and Expert Systems Applications (DEXA'03)*.

Baeza-Yates, R. and B. Ribeiro-Neto (1999). *Modern Information Retrieval*, Addison Wesley, ACM Press.

Banter, B. and R. Wille (1999). *Formal concept Analysis: mathematical Foundations*, Springer Verlag.

Berry, M. W., S. T. Dumais and G. W. O'Brien (1995). Using Linear Algebra for Intelligent Information Retrieval. *SIAM Review* **37**(4): 573-595.

Bharat, K., A. Broder, M. Henzinger, P. Kumar and S. Venkatasubramanian (1998). The Connectivity Server: fast access to linkage information on the Web. *Proceedings of the 7th International World Wide Web Conference*.

Bharat, K. and M. GA (2001). When Experts Agree: Using Non-Affiliated Experts to Rank Popular Topics. *Proceedings of WWW10*, Hong Kong.

Bharat, K. and M. Henzinger (1998). Improved Algorithms for Topic Distillation in a Hyperlinked Environment. *Proc. the 21st International ACM Conference of Research and Development in Information Retrieval (SIGIR98)*.

Borges, J., Levene M (2000). Data Mining of User Navigation Patterns. *Web Usage Analysis and User Profiling* **1836**: 92-111.

Borodin, A., Roberts G, Rosenthal J and T. P (2001). Finding Authorities and Hubs from Hyperlink Structures on the World Wide Web. *Proceedings of the 10th International World Wide Web Conference*.

Botafogo, R. A. and B. Shneiderman (1991). Identifying Aggregates in Hypertext Structures. *Proceedings of Hypertext'91*.

Bourret, R., C. Bornhovd and A. Buchmann (2000). A Generic Load/Extract Utility for Data Transfer Between XML Documents and Relational Databases. *Proceedings of Second International Workshop on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS'00)*, Milpitas, California, USA.

Brin, S. and L. Page (1998). The Anatomy of a Large-Scale Hypertextual Web Search Engine. *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia.

Brin, S. and L. Page. (1998). "The PageRank Citation Ranking: Bringing Order to the Web." Retrieved January, 1998.

Brinkmeier, M. (2002). Communities in Graphs. *Technical Report, Technical University Ilmenau.*

Broder, A., S. Glassman, M. Manasse and G. Zweig (1997). Syntactic Clustering of the Web. *Proceedings of the 6th International WWW Conference*, Santa Clara, CA, USA.

Broder, A., R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins and J. Wener (2000). Graph Structure in the Web. *Proc. of 9th International World Wide Web Conference*.

Bron, C. and J. Kerbosch (1973). Algorithm 457 - finding all cliques of an undirected graphBron, C. *Communications of the ACM* **16** (9): 575-577.

Chakrabarti, S., M. v. d. Berg and B. Dom (1999). Focused crawling: A New Approach to Topic-Specific Web Resource Discovery. *Proceedings of the 8th International World Wide Web Conference*, Toronto, Canada.

Chakrabarti, S., Dom B, Gibson D, Kleinberg J, Raghavan P and Rajagopalan S (1998). Automatic Resource Compilation by Analyzing Hyperlink Structure and Associated Text. *Proceedings of the 7th International World Wide Web Conference*.

Chakrabarti, S., B. E. Dom, S. R. Kumar, P. Raghavan, S. Rajagopalan, A. Tomkins, D. Gibson and J. Kleinberg (1999). Mining the Web's Link Structure. *Computer*: 60-67.

Chakrabarti, S., M. M. Joshi, K. Punera and D. M. Pennock (2002). The Structure of Broad Topics on the Web. *Proc. of 11th International World Wide Web Conference*.

Chakrabarti, S., M. van den Berg and B. Dom (1999). Distributed Hypertext Resource Discovery Through Examples. *Proceedings of the 25th International Conference on Very Large Data Bases*, Edinburgh, Scotland.

Chen, C. and L. Carr (1999). Trailblazing the Literature of Hypertext: Author Co-Citation Analysis (1989-1998). *Proceedings of the 10th ACM Conference on Hypertext and Hypermedia (Hypertext99).*

Cho, H., J., Garcia-Molina and L. Page (1998). Efficient Crawling through URL Ordering. *Proceedings of the 7th International World Wide Web Conference*, Brisbane, Australia.

Cho, J. and H. G.-M. (2000). Synchronizing a Database to Improve Freshness. *Proceedings of the ACM SIGMOD International Conference on Management of Data*.

Cho, J. and H. Garcia-Molina (2000). The Evolution of the Web and Implications for an Incremental Crawler. *Proceedings of the 26th International Conference on Very Large Data Bases*, Cairo, Egypt.

Clarke, C., Cormack GV and T. EA (2000). Relevance Ranking for One to Three Term Queries. *Information Processing & Management*(36): 291-311.

Cohn, D. and C. H (2000). Learning to Probabilistically Identify Authoritative Documents. *Proceedings of the 17th International Conference on Machine Learning*, Stanford University.

Cormen, T. H., C. E. Leiserson and R. L. Rivest (1990). *Introduction to Algorithms*, The MIT Press.

Datta, B. (1995). *Numerical Linear Algebra and Application*, Brooks/Cole Publishing Company.

Dean, J. and M. Henzinger (1999). Finding Related Pages in the World Wide Web. *Proc. the 8th International World Wide Web Conference*.

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer and R. Harshman (1990). Indexing by Latent Semantic Analysis. *J. Amer. Soc. Info. Sci.* **41**(6): 391-407.

Dempster, A. P., N. M. Laird and D. B. Rubin (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal Royal Statist. Soc. B* **39**(2): 1-38.

Diligenti, M., F. Coetzee, S. Lawrence, C. Giles and M. Gori (2000). Focused Crawling Using Context Graphs. *Proceedings of the 26th International Conference on Very Large Data Bases*, Cairo, Egypt.

Diligenti, M., Gori M and M. M (2002). Web Page Scoring Systems for Horizontal and Vertical Search. *Proceedings of the WWW2002*, Honolulu, Hawaii, USA.

Dill, S., R. Kumar, K. McCurley, S. Rajagopalan and D. S. a. A. Tomkins (2001). Self-similarity in the Web. *Proc. of 27th VLDB Conference*.

Ding, C., He X, Husbands P, Zha H and S. H (2002). PageRank, HITS and a Unified Framework for Link Analysis. L. B. N. L. T. Report, University of California, Berkeley,CA.

Ding, J., L. Gravano and N. Shivakumar (2000). Computing Geographical Scopes of Web Resources. *Proc. of the 26th International Conference on Very Large Data Bases (VLDB'00)*.

Duncan, J. and S. H. Watts (1998). Navigation in a small world. *Nature* **393**(400).

Flake, G. W., S. Lawrence and C. L. Giles (2000). Efficient Identification of Web Communities. *Proc. of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

Flake, G. W., S. Lawrence and C. L. Giles (2002). Self-Organization of the Web and Identification of Communities. *IEEE Computer* **35**(3).

Flake, G. W., R. E. Tarjan and K. Tsioutsiouliklis (2004). Graph Clustering and Minimum Cut Trees. *Internet Mathematics* **1**(4).

Frivolt, G. and M. Bielikova (2005). Topology Generation for Web Communities Modeling. *Proc. of the 31st Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM 2005)*.

Funk, P., A. Lewien and G. Snelting (1998). Algorithms for Concept Lattice Decompositino and their Applications. *Technical Report, TU Braunscheweig, FE Informatik*.

Garfield, E. (1972). Science Citation Analysis as a Tool in Journal Evaluation. **178**: 471-479.

Garfield, E. (1979). *Citation Indexing: Its Theory and Application in Science, Technology, and Humanities*. New York, John Wiley & Sons.

Gibson, D., J. Kleinberg and P. Raghavan (1998). Inferring Web Communities from Link Topology. *Proc. of 9th ACM Conference on Hypertext and Hypermedia.*

Golub, G. H. and C. F. V. Loan (1993). *Matrix Computations, second edition*, The Johns Hopkins University Press.

Greco, G., S. Greco and E. Sumpano (2002). A Stochastic Approach for Modeling and Computing Web Communities. *Proc. of the 3rd International Conference on Web Information System Engineering (WISE'02).*

Han, E., G. Karypis, V. Kumar and B. Mobasher (1998). Hypergraph Based Clustering in High-Dimensional Data Sets: A Summary of Results. *IEEE Data Engineering Bulletin* **21**(1): 15-22.

Haralick, R. M. (1974). The diclique representation and decomposition of binary relations. *Journal of the ACM* **21**: 356-366.

Haveliwala, T. (2002). Topic-Sensitive PageRank. *Proceedings of WWW02*, Honolulu, Hawaii.

Hock, R. (2000). *The Extreme Searcher's Guide to Web Search Engines: A Hand Book for the Serious Searcher*, New Jersey, USA.

Hou, J. and Y. Zhang (2002). Constructing Good Quality Web Page Communities. *Proceedings of the 13th Australasian Database Conferences (ADC2002)*, Melbourne, Australia.

Hou, J. and Y. Zhang (2003). Effectively Finding Relevant Web Pages from Linkage Information. *IEEE Transactions on Knowledge & Data Engineering (TKDE)* **15**(4): 940-951.

Hou, J., Y. Zhang, J. Cao, W. Lai and D. Ross (2002). Visual Support for Text Information Retrieval Based on Linear Algebra. *Journal of Applied Systems Studies* **3**(2).

Imafuji, N. and M. Kitsuregawa (2003). Finding a Web Community by Maximum Flow Algorithm with Hits Score Based Capacity. *Proc. of the 8th International Conference on Database Systems for Advanced Applications.*

Imafuji, N. and M. Kitsuregawa (2004). Finding Web Communities by Maximum Flow Algorithm using Well-Assigned Edge Capacities. *The IEICE Transactions on Information and Systems* **E87-D**(2).

Kamvar, S., Haveliwala TH, Manning CD and G. GH (2003). Exploiting the Block Structure of the Web for Computing PageRank, Stanford University.

Kamvar, S., H. TH and G. G. Manning CD (2003). Extrapolation Methods for Accelerating PageRank Computations. *Proceedings of WWW03*, Budapest, Hungary.

Kleinberg, J. (1998). Authoritative Sources in a Hyperlinked Environment. *Proc. of 9th ACM-SIAM Symposium on Discrete Algorithms.*

Kleinberg, J. (1999). Authoritative Sources in a Hyperlinked Environment. *Journal of the ACM* **46**.

Kleinberg, J. (2000). Navigation in a small world. *Nature* **406**(845).

Kleinberg, J. (2002). The Small-World Phenomenon: An Algorithmic Perspective. *Proc. of 32nd ACM Symposium on Theory of Computing.*

Kumar, R., P. Raghavan, S. Rajagopalan and A. Tomkins (1999). Extracting large-scale knowledge bases from the Web. *Proc. of the 25th VLDB Conference.*

Kumar, R., P. Raghavan, S. Rajagopalan and A. Tomkins (1999). Trawling the Web for emerging cyber-communities. *Proc. of the 8th International World Wide Web Conference*.

Laender, A. H. F., B. Ribeiro-Neto, A. S. da Silva and E. S. Silva (2000). Representing Web Data as Complex Objects, Electronic Commerce and Web Technologies. *Proceedings of the First International Conference, EC-Web 2000*, London, UK.

Larson, R. (1996). Bibliometrics of the World Wide Web: An Exploratory Analysis of the Intellectual Structure of Cyberspace. *Ann. Meeting of the American Soc. Info. Sci.*

Lawrence, S., C. L. Giles and K. Bollacker (1999). Digital Libraries and Autonomous Citation Indexing. *IEEE Computer* **36**(6): 67-71.

Lempel, R., Moran S (2001). SALSA: The Stochastic Approach for Link-Structure Analysis. *ACM Transaction on Information Systems* **19**(2): 131-160.

Levene, M., Borges J, Loizou G (2001). Zipf's Law for Web Surfers. *Knowl-edge and Information Systems an International Journal*: 3.

Li, L., Shang Y and Z. W (2002). Improvement of HITS-based Algorithms on Web Documents. *Proceedings of WWW2002*, Honolulu, Hawaii, USA.

Lifantsev, M. (2000). Voting Model for Ranking Web Pages. *Proceedings of the International Conference on Internet Computing*, CSREA Press, Las Vegas.

Liu, B., Y. Ma and P. S. Yu (2001). Discovering unexpected information from your competitors' Web sites. *Proc. of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.

McBryan, O. A. (1994). GENVL and WWWW: Tools for Taming the Web. *Proceedings of the First International World Wide Web Conference*, CERN, Geneva, Switzerland.

McHugh, J., S. Abiteboul, R. Goldman, D. Quass and J. Widom (1997). Lore: A Database Management System for Semistructured Data. *SIGMOD Record* **26**: 54-66.

Meng, W., C. Yu and K. Liu (2002). Building Efficient and Effective Metasearch Engines. *ACM Computing Surveys*: 48-84.

Mishra, N., D. Ron and R. Swaminathan (2003). On Finding Large Conjunctive Clusters. *Proc. of the 16th Annual Conference on Computational Learning Theory (COLT'03)*.

Mitzenmacher, M. (2004). A Brief History of Generative Models for Power Law and Lognormal Distributions. *Internet Mathematics* **1**(2): 226-251.

Mobasher, B. (2004). Web Usage Mining and Personalization. *Practical Handbook of Internet Computing*. M. P. Singh, CRC Press.

Mobasher, B., H. Dai, M. Nakagawa and T. Luo (2002). Discovery and Evaluation of Aggregate Usage Profiles for Web Personalization. *Data Mining and Knowledge Discovery* **6**(1): 61-82.

Mukherjea, S. and Y. Hara (1997). Focus+Context Views of World-Wide Web Nodes. *Proceedings of the 8th ACM Conference on Hypertext (Hypertext97)*.

Murata, T. (2000). Discovery of Web Communities Based on the Co-Occurence of References. *Proc. of the 3rd International Conference on Discovery Science (DS'00)*.

Murata, T. (2001). A Method for Discovering Purified Web Communities. *Proc. of the 4th International Conference on Discovery Science (DS'01).*

Murata, T. (2003). Discovery of Web Communities from Positive and Negative Examples. *Proc. of the 6th International Conference on Discovery Science (DS'01).*

Najork, M. and J. Wiener (2001). Breadth-first search crawling yields high-quality pages. *Proceedings of the 10th International World Wide Web Conference*, Hong Kong.

Ng, A., Zheng AX and J. MI (2001). Stable Algorithms for Link Analysis. *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New Orleans, Louisiana, USA.

Ng, A., Zheng AX, Jordan MI (2001). Link Analysis, Eigenvectors and Stability. *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI 01)*, Seattle, Washington, USA.

Ohsawa, Y., H. Soma, Y. Matsuo, N. Matsumura and M. Usui (2002). Featuring Web communities based on word co-occurrence structure of communications. *Proc. of the 11th International World Wide Web Conference (WWW11).*

Otsuka, S., M. Toyoda, J. Hirai and M. Kitsuregawa (2004). Extracting User Behavior by Web Communities Technology on Global Web Logs. *Proc. of the 15th International Conference on Database and Expert Systems Applications (DEXA'04).*

Özsu, M. T. and P. Valduriez (1991). *Principle of Distributed Database Systems.* Englewood Cliffs, New Jersey, USA, Prentice Hall, Inc.

Page, L., Brin S, Motwani R, Winograd T (1998). The Pagerank Citation Ranking: Bringing Order to the Web, Computer Science Department, Stanford University.

Papadimitriou, C., P. Raghavan, H. Tamaki and S. Vempala (1997). Latent Semantic Indexing: A Probabilistic Analysis. *Proceedings of ACM Symposium on Principles of Database Systems.*

Papadimitriou, C. H. and K. Steiglitz (1998). *Combinatorial Optimization: Algorithms and Complexity*, Dover Publications, Inc.

Papakonstantinou, Y., H. Garcia-Molina and J. Widom (1995). Object Exchange Across Heterogenous Information Sources. *Proceedings of the Eleventh International Conference on Data Engineering*, Taipei, Taiwan.

Pennock, D. M., G. W. Flake, S. Lawrence, E. J. Glover and C. L. Giles (2002). Winners don't take all: Characterizing the competition for links on the Web. *Proc. of the National Academy of Sciences.*

Perkowitz, M. and O. Etzioni (1998). Adaptive Web Sites: Automatically Synthesizing Web Pages. *Proceedings of the 15th National Conference on Artificial Intelligence*, Madison, WI, AAAI.

Pierrakos, D., G. Paliouras, C. Papatheodorou, V. Karkaletsis and M. D. Dikaiakos (2003). Construction of Web Community Directories by Mining Usage Data. *Proc. of the 2nd Hellenic Data Management Symposium (HDMS'03).*

Pierrakos, D., G. Paliouras, C. Papatheodorou, V. Karkaletsis and M. D. Dikaiakos (2003). Web Community Directories: A New Approach to Web Personalization. *Proc. of the 1st European Web Mining Forum (EWMF'03).*

Pitkow, J. and P. Pirolli (1997). Life, Death, and Lawfulness on the Electronic Frontier. *Proceedings of ACM SIGCHI Conference on Human Factors in Computing (CHI 97).*

Reddy, P. K. and M. Kitsuregawa (2001). An approach to relate the Web communities through bipartite graphs. *Proc. of the 2nd International Conference on Web Information Systems Engineering (WISE'01).*

Rennie J. and A. McCallum (1999). Using Reinforcement Learning to Spider the Web Efficiently. *Proceedings of the International Conference on Machine Learning (ICML).*

Richardson, M., Domingos P (2002). The Intelligent Surfer:Probabilistic Combination of Link and Content Information in PageRank. **14**.

Rome, J. E. and R. M. Haralick (2005). Towards a Formal Concept Analysis Approach to Exploring Communities on the World Wide Web. *Proc. of the 3rd International Conference on Formal Concept Analysis (ICFCA'05).*

Salton, G. and M. MJ (1983). *Introduction to Modern Information Retrieval*. New York, NY, McGraw-Hill.

Sha, F., G. Gardarin and L. Nemirovski (2000). Managing Semistructured Data in Object-Relational DBMS. *Networking and Information Systems Journal,* **Vol. 1** (1).

Shivakumar N., H. G.-M. (1998). Finding Near-Replicas of Documents on the Web. Workshop on Web Databases. Valencia, Spain.

Sonnenreich, W. and T. Macinta (1998). *Web Developer.Com Guide to Search Engines*, ohn Wiley & Sons, Inc., USA.

Stewart, G. and S. JG (1990). *Matrix Perturbation Theory*, Academic Press.

Strang, G. (1993). *Introduction to Linear Algebra*, Wellesley-Cambridge Press.

Surjanto, B., N. Ritter and H. Loeser (2000). XML Content Management based on Object-Relational Database Technology. *Proceedings of the 1st International Conference on Web Information Systems Engineering (WISE2000)*, Hong Kong, China.

Talim, J., Z. Liu, P. Nain and E. Coffman (2001). Controlling Robots of Web Search Engines. *Proceedings of SIGMETRICS Conference*.

Tawde, V. B., T. Oates and E. Glover (2004). Generating Web Graphs with Embedded Communities. *Proc. of 3rd Workshop on Algorithms and Models for the Web-Graph (WAW 2004).*

Toyoda, M. and M. Kitsuregawa (2001). Creating a Web Community Chart for Navigating Related Communities. *Proc. of the 12th ACM Conference on Hypertext and Hypermedia (Hypertext'01).*

Toyoda, M. and M. Kitsuregawa (2003). Extracting Evolution of Web Communities from a Series of Web Archives. *Proc. of the 14th ACM Conference on Hypertext and Hypermedia (HT'03).*

Trefethen, L. and B. D (1997). *Numerical Linear Algebra*. Phila-delphia, SIAM Press.

Wang, L. (1997). On Competitive Learning. *IEEE Transaction on Neural Networks* **8**(5): 1214-1217.

Wang, M. (2002). A Significant Improvement to Clever Algorithm in Hyper-linked Environment. *Proceedings of the 11th International World Wide Web Conference (WWW2002)*, Honolulu, Hawaii, USA.

Wang, Y. and M. Kitsuregawa (2001). Use Link-based Clustering to Improve Web Search Results. *Proceedings of the 2nd International Conference on Web Information Systems Engineering (WISE2001)*, Kyoto, Japan.

Weiss, R., B. Vélez, M. A. Sheldon, C. Namprempre, P. Szilagyi, A. Duda and D.K. Gifford. (1996). HyPursuit: A Hierarchical Network Search Engine that Exploits Content-Link Hypertext Clustering. *Proceedings of the Seventh ACM Conference on Hypertext*.

Wen, C. W., H. Liu, W. X. Wen and J. Zheng (2001). A Distributed Hierarchical Clustering System for Web Mining. *Proceedings of the Second International Conference on Web-Age Information Management (WAIM2001)*, Xi'An, China.

Wilkinson, R., Zobel JJ and S.-D. RD (1995). Similarity Measures for Short Queries. *Proceedings of the Fourth Text Retrieval Conference (TREC-4)*, Gaithersburg, MD.

Xu, G., Y. Zhang, J. Ma and X. Zhou (2005). Discovering User Access Pattern Based on Probabilistic Latent Factor Model. *Proceeding of 16th Australasian Database Conference*, Newcastle, Australia, ACS Inc.

Xu, G., Y. Zhang and X. Zhou (2005). Using Probabilistic Semantic Latent Analysis for Web Page Grouping. *15th International Workshop on Research Issues on Data Engineering: Stream Data Mining and Applications (RIDE-SDMA'2005)*, Tyoko, Japan.

Yoon, J. P. and V. Raghavan (2000). Multi-Level Schema Extraction for Heterogeneous Semi-Structured Data. *Proceedings of the First International Conference on Web-Age Information Management (WAIM'00)*, Shanghai, China.

Index

About the Authors

Yanchun Zhang

Dr. Yanchun Zhang is full Professor and the Director of Internet Technologies and Applications Research Laboratory in the School of Computer Science and Mathematics at Victoria University, Australia. He obtained PhD in Computer Science from the University of Queensland in 1991. His research areas cover databases, electronic commerce, internet/Web information systems, Web data management, Web search and Web services. He has published over 100 research papers on these topics in international journals and conference proceedings, and edited over 10 books/proceedings and journal special issues. He is a co-founder and Co-Editor-In-Chief of *World Wide Web: Internet and Web Information Systems* and Chairman of International Web Information Systems Engineering (WISE) Society.

Jeffrey Xu Yu

Dr. Jeffrey Xu Yu received his B.E., M.E. and Ph.D. in computer science from the University of Tsukuba, Japan, in 1985, 1987 and 1990, respectively. Jeffrey Xu Yu was a faculty member in the Institute of Information Sciences and Electronics, University of Tsukuba, Japan, and was a Lecturer in the Department of Computer Science, The Australian National University. Currently, he is an Associate Professor in the Department of Systems Engineering and Engineering Management at the Chinese University of Hong Kong. His research areas cover XML databases, data warehouse, data mining, and data stream mining. He has published over 100 research papers on these topics in international journals and conference proceedings. Jeffrey Xu Yu is a member of ACM, and a society affiliate of IEEE Computer Society.

Jingyu Hou

Dr Jingyu Hou received his BSc in Computational Mathematics from Shanghai University of Science and Technology (1985) and his PhD in Computational Mathematics from Shanghai University (1995). He also received another PhD in Computer Science in the Department of Mathematics and Computing at The University of Southern Queensland, Australia (2004). He is now a Lecturer in the School of Information Technology at Deakin University, Australia. His research interests include Web-Based Data Management

and Information Retrieval, Web Databases, Internet Computing and Electronic Commerce, and Semi-Structured Data Models. He has published extensively in the areas of Web information retrieval and Web Communities.