EXPLORING SOCIAL NETWORKS IN COMPUTER SYSTEMS

YILEI SHAO

A DISSERTATION

PRESENTED TO THE FACULTY

OF PRINCETON UNIVERSITY

IN CANDIDACY FOR THE DEGREE

OF DOCTOR OF PHILOSOPHY

RECOMMENDED FOR ACCEPTANCE

BY THE DEPARTMENT OF

COMPUTER SCIENCE

June 2007

Abstract

Since the first appearance of computers half a century ago, computing technology has evolved rapidly, and it continues to do so. Having the most powerful and fastest computers is no longer the only goal for a successful computer system. A common characteristic of successful computer systems in the modern era is the ability to utilize vast amounts of information efficiently.

We introduce two important concepts from sociology into computer science: human capital and social capital. In a society, people who possess high human capital are individuals with more ability. They are more intelligent, more attractive, more articulate, and more skilled. In the human capital metaphor, these people are more likely to do better than others. Social capital is the contextual complement to human capital. In the social capital metaphor, people who do better are somehow better connected. The asset they possess is an advantageous location in the society. Today's computer systems exhibit a strong resemblance to human society. The linkage relationships among nodes in a system affect the performance of the system to a large extent. A well-connected system can outperform a collection of powerful nodes.

In this dissertation, we present results of applying the social capital metaphor to two kinds of computer systems: focused web crawling and peer-to-peer systems. We have designed and implemented a novel focused crawler that captures the topical linkage in the web. We have introduced the notion of “topical link”. Topical links connect pages with similar content. By combining topical links with hyperlinks, the connectivity of the web graph is greatly enhanced. In our study of peer-to-peer systems, we have introduced the concept of “buddy net”. Buddy net connects peers with similar interests in a peer-to-peer system. It is an interest-based overlay on top of the physical overlay of a peer-to-peer system. With the help of buddy net, the efficiency and robustness of a peer-to-peer system is greatly improved.

Acknowledgement

Six years in my life were spent in Princeton. There were good times and bad times. I loved spring on campus. When driving down Washington road, trees turned into the freshest green and formed a natural arc. I wished the road never ended. Seeing the undergraduates wearing shorts and skirts in the spring zephyr, I felt the heartbeat of the school. All in a sudden, sunshine was everywhere, white tents were scattered on the greens. Summer came. Stepping into the summer heat after spending an hour in the pool was the most enjoyable moment. Fall was the shortest season on campus. It seems only last for two weeks. It flew away as more and more bright yellow and red maple leaves covered the ground. Then it came the long winter. In those gloomy winter days I questioned what the hack was I doing here.

I was fortunate enough to end up in Princeton. Princeton has changed me in many aspects. I have enjoyed classes from English department, History department, Eastern Asia department, Economics department, Philosophy department and Comparative Literature department. I felt like a sponge absorbing as much as I can in this amazing knowledge sea. Professors from all the classes I have attended offered top-notch lectures. They did not brain wash me; they encouraged individual thinking at every possibility. It was a perfect balance between Science and Humanities. Being a graduate student in Princeton has opened my eyes and my mind. It gave me the opportunity to meet friends from all over the world, listen to opinions from different viewpoints. It changed me to a more open-minded person.

I thank my friends the most. I had a gang just as tight as “Friends”. I can count on them in every situation. They will always be there for me. No need to say more. Here is my gang: Xiaomin Chen, Ruoming Pang, Lexing Ying, Guoping Huang, Yi Gu and Jiwei Lu.

I thank my colleagues in Princeton to brainstorm with me about new research ideas, to debate about the design choices and to survive sleepless nights before paper deadline. They are: Fengzhou Zheng, Chi Zhang, Ming Zhang, Junwen Lai and Nitin Garg.

I thank Professor Andrea LaPaugh for being my advisor. She has given me a lot of guidance and help in my work. I thank Professor Jaswinder P. Singh and Moses Charikar for being my thesis reader. They gave many valuable suggestions and comments on my work. I thank Professor Kai Li and Jennifer Rexford for serving as non-readers on my committee. My work was support by NSF grant CNS-0313089.

This thesis is dedicated to my parents. It was extremely hard for them to send me away to a country on the opposite side of the sphere. I wish my PhD thesis made it up a little bit.

Contents

List of Figures

List of Table

Chapter 1

Introduction

Since the first appearance of computers half a century ago, computing technology has evolved rapidly, and it continues to do so. The invention of the Internet has altered the landscape of computer science in recent years. Information has become widely available via the Internet. Having the most powerful and fastest computers is no longer the only goal for a successful computer system. A common characteristic of successful computer systems in the modern era is the ability to utilize vast information efficiently. The best systems in the peer-to-peer world and the World Wide Web all have the same merit. First, let us take a close look at how today's computer systems are evolving to a different breed.

1.1 Socialized Computer Systems

Through technology advancement, computer systems have changed more and more: Computer systems used to be relatively smaller scaled and centralized. They wereusually built and maintained by a sole administrator. The structure of the system was well defined and seldom changes. Data placement was strictly controlled. There are clear goals and usage scenarios. However, these characteristics do not hold in today's computer systems. The rapid growth of the Internet provides easy access to massive information and strong connectivity in cyberspace. New kinds of computer systems have emerged, such as peer-to-peer systems and the World Wide Web. They are no longer built and maintained by a single administrator. Instead, they are built by the collective efforts of a large group of individuals. They are usually self-maintained without the need of dedicated administrators. The scale of such systems is several orders of magnitude bigger than traditional computer systems. Typical peer-to-peer systems involve hundred of thousands of users, and the World Wide Web connects millions of people. These computer systems are decentralized. They do not have a central point of control. Instead, each participating node shoulders the burden of maintenance and enjoys the benefit. The structure and data placement in such systems are flexible. The usage scenarios become the collective behaviors from a large number of users. Each user has the freedom to decide how to participate and how active to be. It is not hard to see that a computer system is not only a computing framework to accomplish a pre-defined goal; it becomes more and more socialized. We call these modern computer systems "Socialized Computer Systems".

1.2 Human Capital and Social Capital in Computer Systems

In sociology, researchers view society as a market in which people exchange all variety of goods and ideas in pursuit of their interests. Certain people, or certain groups of people, do better in the sense of receiving higher returns for their efforts. Some enjoy higher incomes. Some more quickly become prominent. Some lead more important projects. The interests of some are better served than the interests of others. The human capital explanation of the inequality is that the people who do better are more able individuals; they are more intelligent, more attractive, more articulate, and more skilled.

Social capital is the contextual complement to human capital. The social capital metaphor is that the people who do better are somehow better connected. Certain people or certain groups are connected to certain others, trusting certain others, obligated to support certain others, dependent on exchange with certain others. Holding a certain position in the structure of these exchanges can be an asset in its own right. That asset is social capital, in essence, a concept of location effects in differentiated markets. [42, 43, 44, 45]

There are two network mechanisms typically discussed as social capital: closure and brokerage. In these mechanisms, the network affects the flow of information and what people do with it. They begin with the assumption that communication takes time, so prior relationships affect who knows what early. Information can be expected to spread across the people in a market, but it will circulate within groups before it circulates between groups. A generic research finding in sociology is that information circulates more within than between groups. Examples are that information circulates more within a work group than between groups, more within a division than cross divisions, more within an industry than cross industries [58, 59]. The result is that people are not simultaneously aware of opportunities in all groups. Even if information is of high quality, and eventually reaches everyone, that fact that diffusion requires an interval of time means that individuals informed early or more broadly have an advantage [42, 46, 47, 50].

Today's computer systems exhibit strong resemblances to the human society. Computer systems are built and maintained by a large number of participating peers. The structure of the systems is decentralized. Peers independently decide how they want to participate and behave in selfish ways. We could easily draw an analogy between individuals in the society with peers in these computer systems. The concept of human capital is applicable in such systems. Some peers have more content to offer. Some enjoy high communication bandwidth. Some possess strong computation power. The concept of social capital is also applicable to such systems. Some peers are better connected throughout the system and their needs are better served. Some peers possess strategic positions in the system and see information flows from different groups.

In human society, improving human capital and improving social capital for individuals are the main approaches to increase the efficiency of the whole system and the happiness of every individual. From our analogy between human society and computer systems, we can see that improving human capital and social capital for peers in computer systems are the ways to improve the performance for the whole system.

1.3 Improving Human Capital or Social Capital?

For a long time, researchers and computer scientists have tried very hard to improve the performance of computer systems. Whenever a new computer system was built, let it be a large-scale program to solve complex calculation or distributed systems to conquer a distributed task, they looked into their tool bag and picked the most familiar tools. These tools include using new hardware, improving the communication latency, caching, pre-fetching, etc. Computer scientists have successfully utilized these tools on traditional computer systems. However, scalability has come into play. The scale of today's computer system becomes much larger, easily reaching a million nodes. Capturing the hardware advantage is not only expensive but also inefficient. Applying old tricks such as caching and pre-fetching does not always yield favorable results.

Classic methods of improving performance concentrate on improving the human capital of the nodes. The goal is to make every node a better node. It was easy to see significant improvement when most of the nodes were simple and inferior. However, after applying these methods over and over again, the room for improvement has become very limited. After most nodes are equipped with these tools, even a very complex and fine-tuned method cannot achieve significant advancement any more.

As we have discussed in previous sections, the value of the individual comes not only in the form of human capital, but also in social capital. The combination of human capital and social capital accurately describe the value of the individual in the society. We believe that improving social capital is as important as improving human capital. Instead of trying to make every peer a better node, new approaches should be devised to make the system better connected. Better connected could mean having a dense connection graph or a meaningful organization of the peers throughout the system. Since there is very limited room for improvement by concentrating on human capital, improving the social capital becomes a very promising road to take.

1.4 Overview of the Dissertation

In this dissertation, we illustrate the idea of improving performance in socialized computer systems by improving the social capital for peers in these systems. We first introduce two socialized computer systems of interest, peer-to-peer systems and the World Wide Web. We investigate their organization, graph structure and usage patterns. In chapter 2, we discuss the motivation for our work and lay out the goals we would like to achieve. In chapter 3, we explain recent research advancements in these two areas. We describe the advantage and disadvantage of related work. We identify the emerging trends in both areas. In chapter 4, we present the first part of the thesis: “Topical Crawler: Focused web crawling through topical linkage”. We focus our research on answering the following three questions: 1) Are there useful structures in the web besides hyperlinks? 2) Can we capture the structural information and use it to benefit the information-hunger users? 3) Can such captured structural information have a lasting effect over the web’s evolution? We designed and implemented a novel focused crawler that captured the topical linkage on the web. We introduce the notion of “topical link”. Topical links connect pages with similar content. Combining topical links with hyperlinks, the connectivity of the web graph is greatly enhanced. We show results from different experiments. In chapter 5, we present the second part of the thesis: “BuddyNet: History-based Peer-to-peer Search”. In this part, we draw a clear analogy between individuals in a human society and peers in a p2p system. We apply the social capital concept to peers in the system. We introduce the concept of “buddy net”. Buddy net connects peers with similar interests in a peer-to-peer system. It is an interest overlay on top of the physical overlay in peer-to-peer systems. With the help of buddy net, the efficiency and robustness of the system is great improved. Finally, we summarize the finding from these two systems and discuss future works in chapter 6.

Chapter 2

Motivations and Goals

The thriving of the Internet changes the landscape of the technology world. Two types of computer systems caught our attention: the web crawler and the peer-to-peer system.

Web crawling is a technique widely used by Internet search engines. In order to perform keyword search, a search engine needs to build its repository of web pages first. Web crawling is the process of going out to the web and collecting web pages. The coverage and the quality of the web pages collected by the web crawler directly affect the results of searches using the search engine. We are interested in the web crawler because of the important role it plays in the web search arena and the way it interacts with the web. The web is a free place for any individual to present information. There is almost no restriction on how web pages should be presented and how pages are linked together. The lack of authority and the way the web is maintained by the collective effort of a large number of users match the definition of a socialized computer system very well. We can easily draw an analogy between web pages on the web and individuals in society. The web crawler is not only a very useful tool to study the relationship among web pages but also a useful application to utilize the discovered relationships.

The class of peer-to-peer systems is one of the fastest growing applications on the Internet. A peer-to-peer system is a decentralized file sharing system: every peer participates in the system by sharing some content on its hard disk. By participating in the system, it can ask for contents from other peers. Peers communicate either directly or by relaying messages through other peers. Systems like Gnutella, Kazza and Edonkey are typical examples of peer-to-peer systems. The distributed form and anonymity of peer-to-peer systems attract the mass public. The lack of central control and the freedom to join or quit at will add even more appeal. From a research point of view, the peer-to-peer system is a mirror image of the human society in cyber space. Every peer in the system resembles an individual in the society. However, the anonymous feature makes peer-to-peer systems even more interesting to study. Unlike human society, peers in peer-to-peer systems join or quit the system with high frequency because there is almost no penalty for such behavior. Studying the relationship between peers in such dynamic systems is an interesting topic by itself. Being able to find stable links in the system and utilize such links to achieve efficiency is very desirable.

In this chapter, we explain the motivations of our work in focused web crawlers and peer-to-peer search systems and the goals we would like to achieve.

2.1 Motivation in Focused Web Crawling

The relationship between the web pages in the World Wide Web is a very interesting topic to study by itself. How are the pages linked on the web when there is no restriction imposed? How does a person decide what pages to link to when she first creates a web page? How do the links change over time? How do other pages find this page and link to it? Does the relationship between the pages change over time?

One way to investigate the above questions is to take a look at the web. Today, the web has grown to an enormous size containing billions of web pages. Taking a complete snapshot of the web and analyzing its pattern are almost impossible because of the time and storage requirements. However, if we concentrated on a particular part of the web, such analysis can be performed. Focused web crawling serves the purpose. A focused web crawler is a robot that goes to the web and collects only the web pages related to a certain topic. Since the web consists of web pages covering thousands of topics, a focused crawler needs to collect a much smaller sample of the web. It can return a reasonable result set in hours instead of days or weeks. Focused web crawlers have not only research use but also practical applications. One of the most popular uses of focused web crawlers is to find resumes online for recruiters. Being able to find just the resumes that fit the job description is invaluable to recruiters. Being able to find the most up-to-date resumes in several hours is another huge plus. A general keyword-based search by a web search engine does not always give the best result set. For search criteria that are hard to describe in keywords, the result set might contain very little useful information. For example, what would be the right keyword combination to find candidates for McKinsey. & Co? In other cases, the result set could miss a huge set of potential candidates. For example, when trying to find candidates with the ability to solve differential equations and with interest in finance, the keywords search of "differential equation, finance, and resume" returns resumes from candidates with financial engineering degrees. A lot of candidates who have a degree in physics and mathematics can do the job just as well as those from financial engineering. Since very few of them put financial terms in their resume, keyword-based search engines would most likely exclude their resumes from the search results. Focused crawling can help in this case. By analyzing the relationship and similarity between pages, a focused crawler can concentrate in a small portion of the web and return targeted pages with higher precision. For example, when a focused crawler finds a matching resume page in the physics department of one university, it would utilize links around this page to find other potentially matching pages. Ideally, when it finds a rich region on the web, it could spend more time in following the link structure to explore more extensively in this region. In contrast, when it reaches a region with limited content, it would quickly redirect itself towards other areas and spend little time in such area.

By analyzing the relationship between pages, we would like to achieve several goals. First, we would like to train the crawler to recognize patterns and identify different regions on the web. By doing this, the focused crawler will return a result set containing few false positives and containing valuable content that cannot be found by simple keyword-based search. Second, we would like to add missing links between pages to make related pages closer to each other. Just like introducing people to each other, we could introduce pages to each other. Improving the linkage between web pages will enhance the performance of later focused crawls, thus creating a positive feedback loop.

2.2 Motivation in Peer-to-peer Search System

Our primary research interest is to explore the relationships of participants in modern computer systems and study the life cycle of such relationships. We view a peer-to-peer system as an extension of human society to cyber space. Every peer in the system resembles an individual in the society. How do peers interact with others? How do relationships develop among peers? Do peers form groups and clusters over the time? What is the rational behind these clusters? These questions spur great interest.

A peer-to-peer system is a distributed content sharing system. Every peer in the system provides shared content to others. As the reward, it gets access to the shared content offered by other peers. Peers form point-to-point connections to each other on top of their physical Internet connections. Management of the system is decentralized. There usually are no controlling nodes or administrators. Peers can join or leave the system at will. Peers in the system are heterogeneous. Size of shared contents, communication bandwidth and connectivity in the overlay network can be vastly different. Ipeer-to-peer systems differ from traditional computer systems in two aspects. First, the heterogeneity of the peer-to-peer system highlights the importance of the location in the system. In other words, a perfectly located peer might have its needs served better even with lower bandwidth and weaker connectivity. Second, the ad-hoc nature of the system with peers joining and leaving all the time underscores the importance of finding stable relationships and being able to adapt dynamically. In a system with high network instability, peers that utilize stable links in the system will be less affected. Peers that adapt their connections to absorb the network change enjoy the most benefit of the system.

Our study in peer-to-peer networks involves the following goals. First, we want to analyze existing peer-to-peer systems to better understand the organization of such distributed and self-organizing systems. We are interested in three aspects of such systems: connectivity, content distribution, and dynamic adaptation.

- Connectivity: What percentage of peers are well connected; what percentage of peers are poorly connected? Is there a pattern to fit the distribution of connectivity? How does a peer position itself when first joining the network; how does it change its connections to other peers after participating actively in the network? How do peers discover and rebuild their connections when other peers leave the network? Are the connections in the system rather stable or short-lived?
- Content distribution: How are contents distributed among peers? Can we categorize peers into groups by the contents they host? After exchanging contents with other

peers, how does the composition of contents change on a peer? Is the peer hosting most of the contents for its own interest or for the well functioning of the entire network? How do rare items get discovered in the system? Once discovered, are the rare items distributed widely across the network?

- Dynamic adaptation: How does the system evolve through time? If we compare the system at the beginning with the system after a couple months, what difference can we find? How do we measure the efficiency of the peer-to-peer system? Is the resulting system a better system in terms of its connectivity and content distribution? Do peers get their requests served in an efficient way if they stay with the system for a long time?

Secondly, with the results from the above analysis, we would like to explore ways to further enhance the efficiency of peer-to-peer systems. The ultimate goal for an efficient peer-to-peer system is to serve every peer's request with the lowest latency and minimal load on the system. Such a system needs to behave well in dynamic situations when peers join and leave freely. To achieve this goal, we experiment with an approach that adapts the peers' connectivity through their message exchanges. After participating in the system for a while, every peer should naturally find the right position for itself and connect to peers with high probability to serve its requests. Adapting the content distribution in the network provides another way to enhance the system performance. During the lifetime of the system, every peer exchanges messages with other peers either by requesting the content directly or by serving as an intermediate node for other peers. By controlling what content to retain during the exchange, we can alter the content distribution in the network. Rare items will be retained by intermediate nodes and become more available to most of the nodes in the network. By distributing the content smartly, the system will become more resilient to peers joining and quitting the network. Our design does not require peers to keep information that is not directly self-beneficial. We try to achieve better performance for the whole system through each peer's self-benefiting behavior.

Lastly, if achieving efficiency requires the system designer to impose hierarchy and order in peer-to-peer systems, what is the balance between keeping the distributed and self-organizing features of the original system and reaching for better efficiency?

Chapter 3

Related Work

In this chapter, we discuss the related works in web crawling and peer-to-peer systems and compare our approach with prior work.

The World Wide Web is one of the most important information sources in today's world. Improving the performance and accuracy of web search has attracted the brightest minds all over the world. Nevertheless, the grand scale of the Web and its intrinsic chaotic nature make it very hard to design and implement the best algorithms. We introduce recent works in the web crawling and discuss the pros and cons of each of them. We show that earlier methods paid attention to improving the efficiency of the crawling process. They tried hard to validate that every page the crawler found is indeed a good page. Recently proposed methods aim more on augmenting the web graph to influence the crawling path in order to achieve higher precision and recall. We believe that improving the web graph connectivity is the future direction in this area. We introduce the notion of "topical link". Topical links connect pages with similar content. Combining topical links with hyperlinks, the connectivity of the web graph is greatly enhanced. Tasks on the web, such as crawling and searching can be performed much more efficiently.

Peer-to-peer system has become "the fastest growing Internet application" [34], there has been numerous research groups working in this area in the past several years. We highlight the most recent works and compare the advantages and disadvantages among them. At the end of our comparisons, we conclude that methods targeting at improving the ability of each node in the system are less successful, methods aiming at improving the connectivity of the system are more popular and they achieve better results.

3.1 State-of-the-Art in Focused Web Crawling

A typical focused crawler consists of a topic classifier controlling the priority of unvisited pages in the task queue and a page fetcher actively fetching pages ordered by their priorities. A focused crawler starts from a set of seed URLs and selectively expands the crawl map based on the result of the topic classification. The goal is to harvest most pages related to a specific topic while avoid downloading irrelevant pages.

How to expand the crawl map is the central question determining the performance of the focused crawler. Choosing wisely about which directions to pursue and which to avoid is very important in focused crawling. Traditionally, researchers expand the crawl map by following the outward hyperlinks of visited pages. (An outward link is a hyperlink going out from the page.)

A great deal of effort has been invested in refining the priority of unvisited pages. In an early paper by Cho *et al.* [3], the author experimented with various strategies of URL ordering in web crawling. Although their study was not focused on crawling for a specific topic, their results cast light on the importance of the ordering of unvisited pages in web crawling. In a later survey conducted by Menczer *et al.* [7], they showed that BestFirst algorithm outperformed PageRank based algorithm in topic-driven web crawling. BestFirst algorithm performs a depth-first traversal during web crawling. On the contrary, PageRank algorithm performs a breadth-first traversal and evaluates the results afterwards. Their results underscored the importance of choosing the right path during the crawling process.

Rennie *et al.* [1] and Diligenti *et al.* [2] each proposed an algorithm to utilize the context information around target web pages. Rennie and McCallum integrated reinforcement learning in the crawling process. Before the crawling, the crawler was trained to learn a mapping from the text in the neighborhood of a hyperlink to the expected number of relevant pages that can be found as a result of following that link. During the crawling, the focused crawler directs itself to links that are more likely to point to a target page using its knowledge learned from the training process. Diligenti *et al.* analyzed the surroundings for on-target pages and built a context graph that captured the typical hyperlink hierarchies within which valuable pages occur. They then used this context model as classifier to identify pages that could lead to potential on-target documents. Both algorithms showed the importance of the context information around the target pages in focused crawling. Such context information includes the hierarchy of the web site, the linkage information between sibling pages, the text and order of sibling page on the parent page. In their paper, they showed good improvement for the harvest rate of their crawlers. (Harvest rate is a ratio between the number of on target page found by the crawler and the number of total crawled pages) Building a rather complete context graph for a local web site tends to be easy. The difficulty for both algorithms lies in building an accurate context mapping or context graph for a large collection of unrelated web pages.

As another attempt to better prioritize unvisited pages, Chakrabarti *et al*. [5] used two classifiers to assign priorities to unvisited frontier pages. Their basic classifier collected information from crawled pages. Their apprentice classifier utilized the information in the Document Object Model (DOM) tag-tree structure from the HTML pages and its text to assure more accurate classification. Their approach treats the entire DOM tag-tree structure as the context information valuable to the focused crawler. Since the HTML DOM tree is not always organized in a structural way that matches the relationship of the contents on the page, it is harder to extract meaningful information from it. Chakrabarti *et al*. combined the structural information from the DOM tree and the content information from the text on the page to train their classifier.

Aggarwal *et al.* [6] proposed the intelligent crawler using the inward linking web page content, candidate URL structure or other behaviors of the inward linking web pages or siblings to estimate the probability that a candidate is useful for a given crawl. (An inward link is a hyperlink pointing to the page of interest.)

Our work differs from previous works [61]. Previous works concentrate on improving the success rate of finding the right crawl path. They tried their best to make sure that the focused crawler follows the paths that would most likely lead to on-target pages and prunes other paths that do not appear as promising. We take a different step. We believe that making sure the focused crawler jump towards the correct direction is important; we also think that providing more information at the time for the crawler to decide where to go next is another important point. We introduce the concept of topical links. Topical links are similar to hyper links in terms that they link web pages together. Hyper links are put up by arbitrary webpage creators to point to other pages. Topical links are links found by our crawler and added to pages with strong topical similarities. Besides pruning the crawl path, we expand the crawl path to include not only hyperlinks but also topical links. After the topical crawler collects all the pages, they are sent to an analyzing engine to extract the topical linkage among these pages. Topical links are added to pages with similar topical interest considering also the path length among them.

Utilizing topical link structure leads to fast navigation among relevant pages. On one hand, when the focused crawler finds a rich area with a lot of on-target pages, the topical links from these on-target pages will lead the crawler to explore the area in depth. Since topical links are shortcuts among pages with similar contents, the crawler can retrieve on-target pages faster without fetching other pages in the webpage hierarchy. On the other hand, when the focused crawler reaches an area with few on-target pages, the topical links will direct the crawler to leave the area and point it to an on-target page in another area. Using that on-target page as the starting point, the focused crawler can start exploring the newly encountered area more efficiently. The success of our approach largely depends on the existence of topical locality in the web and our ability to extract it.

Various researchers have studied topical locality in the web. Davidson [4] conducted an empirical study on DiscoWeb dataset. He found that there was topical locality in the hyperlink structure of the web. For example, linked pages tend to have similar textual content comparing to unrelated pages; when the hyperlinks from the parent page are close together, the sibling pages pointed by these links tend to be similar to each other.

Study on the broad topics of the web by Chakrabarti *et al.* [11] has explored the background distribution of broad topics on the Web. In reality, different topics are not represented the same on the web. From their experiment, they found that there are deviations between the topic representation from popular online directories (such as Dmoz or Yahoo directory) and topics gathered from random walking the web. Commercial interest clearly boosted some of the topic, while other topics appeared to be under-represented. We are mostly interested in the finding of how pages relevant to different topics cite each other. After analyzing the NEC web crawling data set, they built a topic citation matrix to represent the relevance of topics on the web. It is exciting to see that in this matrix, there are topics with clearly higher relevance to each other. Such as from /Art/Music to /Shopping/Music and /Shopping/Entertainment/Recordings, or /Art/Literatures and /Art/Movies. Such finding has great implication in the focused crawler arena.

Our work is especially useful in connection with the topic citation matrix: for every topic, we can build redirection portal for several highly relevant topics. If /Sports/Basketball shows higher topical relevance to /Sports/Football in the topic citation matrix, we could instruct our focused crawler to redirect to a “Football” page when a “Basketball” page is encountered. There is no need to build redirection portal for “Sciences” pages if the topical proximity between “Basketball” and “Science” is low. By identifying the topical relevance among topics, we give the focused crawler extra guidance under situation where no explicit topical relationship can be found.

3.2 State-of-the-Art in Peer-to-peer Search Systems

Improving Gnutella-like system’s scalability has become a hot research topic because the popularity of the peer-to-peer systems. The simplest approach of communication in such system is to flood every peer with every request. Flooding worked in the initial stage of peer-to-peer system. When the system grows dramatically in size, flooding approach doesn’t scale as well. The number of messages grows so fast that the system will be quickly filled and become unresponsive. Various groups have tried to find new solutions to solve the scalability issue of Gnutella-like system. Approaches based on expanding ring and random walk [1], where queries are forwarded to a randomly chosen neighbor, are designed to limit the scope of the queries and avoid the message explosion caused by the simple flooding mechanism. Such approaches are effective when the replication factor in the system is high. (Replication factor indicates for a single file on average how many peers possess it.) In other words, such approaches are good at finding popular content. However, it does not outperform flooding scheme in finding rare items.

Kazaa [36] and Gia [22] both adopt super-node based architecture. A super-node takes the responsibility of indexing content located at other peers. When locating content, a peer contacts its super-node first. A super-node may subsequently contact other super-nodes. Super-nodes are nodes with higher computation power and communication bandwidth. The super-node approach needs accurate accounting of peers' capacities and distributes contents accordingly. In Kazaa's case, super-node may be manually specified. In Gia's case, the overlay network adapts its topology based on each node's capacity. A study [27] shows that peers tend to deliberately misreport information if there is an incentive to do so. We think that a mechanism to ensure accurate accounting of peers' capacities is the key to success in these systems. The super-node approach sacrifices the heterogeneity of the peer-to-peer system; it requires the super-node to be more powerful and more capable. Implicitly, it requires the super-node to be stable in the network since reorganization from losing a super-node is expensive. However, from behavior analysis of peer-to-peer network, above assumption is usually not true. Peers tend to leave and join the network frequently.

pSearch [8] and SETS [38] utilize techniques from information retrieval systems. SETS organizes peers into a topic-segmented topology, and data placement is strictly controlled. Queries are then matched and routed to the topically closest regions. pSearch distributes document indexes through the P2P network based on document semantics generated by Latent Semantic Indexing (LSI). Because of the controlled data placement and underlying topology, both systems can achieve low search cost. Both systems work well when the content of the system stays the same, such as a system for publication retrieval. Nevertheless, when large portion of data has changed, LSI needs to be recomputed and data needs to be redistributed. This may cause a high maintenance cost. The most active peer-to-peer network contains video and audio contents to share among peers. Because of the size of the content and the short livelihood, such contents change very frequently in the system. PSearch and SETS may apply well under certain situations, but they are not suitable for todays widely used peer-to-peer system.

Freenet [40] is a P2P system built on top of Distributed Hash Table (DHT). It utilizes query responses to dynamically adapt nodes' routing tables. Its goal is to make a node specialize in locating sets of similar keys. A node gradually accumulates more and more information about how to route the queries that it is asked for. One thing worth pointing out is: this information may not be directly self-beneficial. A node may have collected the most knowledge about how to find contents related to film star Tom Cruise by participating in the routing for a while. It may not be interested in this information at all. The philosophy of Freenet is to benefit each individual node by benefiting the system as a whole. However, a study [27] has shown that the participants in peer-to-peer systems tend to be selfish. The peers try to maximize their own benefit without considering the system as a whole. Peers try to find as much content as they need from the system and quit the system after their needs are fulfilled. The Freenet paradigm regards peers as responsible and assumes the network is rather stable. In reality, stability cannot be guaranteed in peer-to-peer system. Our approach considered the frequent changes from the participants of the network. We assume peers are selfish and they join and quit the network at their will. Our design does not require peers to keep information that is not directly self-beneficial. We try to achieve better performance for the whole system through each peer's self-benefiting behavior.

More recently, associative overlays [21], and interest-based shortcuts [24] proposed different techniques to improve Gnutella's performance based on interest-based locality. An associative overlay forms a guide-rules based overlay on top of Gnutella's network. The vast interest space is partitioned into a set of guide-rules. Every peer participates in some guide-rules based on its interest when the peer first joins the network. Search is carried out within the scope of a guide-rule group and propagates to bigger groups if the content cannot be found in the smaller group. Associative overlay is effective, especially in finding rare items. By narrowing the search scope, it decreased the search cost. However, it needs a human to identify in which guide rule to participate. The partition of the guide-rule groups directly affects the performance and the accuracy of the search. There is similarity between associative overlay and our system. We also want to narrow the scope for every search to improve the performance of the system based on peers' interest. Instead of partitioning the interest group statically at the beginning, we let the peers adapt the interest partition during the lifetime of the system. The longer a peer participates in the system, the more information it will gather on other peers that share similar interest.

The interest-based shortcuts technique keeps shortcuts to nodes that satisfied previous queries. It is similar to our buddy list. We have shown that with only this simple technique, the system does not perform well in real-world situations. Our system combines the buddy list, one hop indexing and dynamic adaptation techniques to utilize the interest-based locality and cluster peers by their mutual interests [60]. We have shown that it is the combination of these techniques that achieves the biggest performance advantage.

Chapter 4

TopicalCrawler: Focused Web Crawling through Topical Linkage

4.1 Introduction

4.1.1 Problem overview

Since the introduction of the World Wide Web in 1990, the number of web pages has grown dramatically. Google currently indexes about 4 billion web pages that account for a fraction of the total web pages. New information in every topic is getting published everyday. Finding the right information a user asks for becomes more and more challenging in the enormous information sea.

Keywords-search has been the dominant mode of information discovery in the web. Nevertheless, given the massive information available online, surfers' expectation expands from looking for specific document or finding answers to specific question to finding a set of documents relevant to the topic of interest. For example, recruiters want to find resumes for software engineers with more than 3 years of C++ experience. Researchers want to find papers published in the past two years about bioinformatics. Venture capitalists want to find homepages for all Internet service startup.

To meet users' topic specific information needs, a set of large scale and topic-driven focused crawlers have been proposed ([1, 2, 12, 16]).

A typical focused crawler consists of a topic classifier controlling the priority of unvisited pages in the task queue and a page fetcher actively fetching pages ordered by their priorities. A focused crawler usually starts from a set of seed URLs and selectively expands the crawl map based on the result of the topic classification. The goal is to harvest most pages related to a specific topic while avoid downloading irrelevant pages.

How to expand the crawl map is the central question determining the performance of the focused crawler. Choosing wisely about which directions to pursue and which to avoid is very important in focused crawling. Traditionally, researchers expand the crawl map by following the outward hyperlinks of visited pages. We believe that hyperlink relationship is not the only useful structure in focused crawling. Pages should be linked by their topical relevance as well. An enhanced focused crawler should follow topical links as well as hyperlinks at crawling time in order navigate through highly relevant pages efficiently.

4.1.2 Baseline focused-crawler

In this section, we introduce the baseline focused-crawler. The baseline crawler is a straightforward focused crawler. We later compare our enhanced focused-crawler against it. Figure 4.1 shows the structure of a baseline crawler.

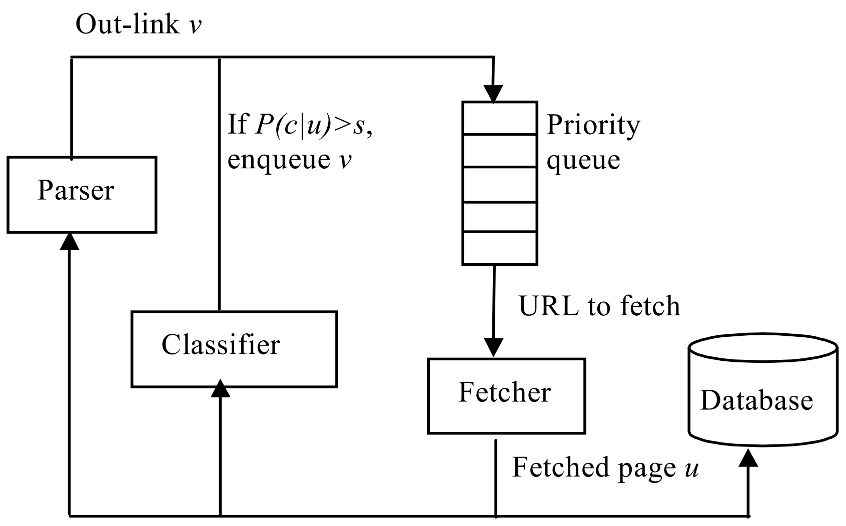


Figure 4.1. Baseline focused-crawler

Prior to performing a focused-crawl, a classifier needs to be created. Sample URLs from online directories such as Dmoz [18] and Yahoo directories are fetched and a topic-taxonomy is built on the sample pages. The Dmoz directory contains about 600,000 distinct topics. We select its first or second level topics based on the topics of interest. For each topic, we randomly select couple hundred of pages from the links provided by Dmoz under this topic and fetch these pages from the web. We store fetched pages in a repository. The classifier is trained on the pages in the repository. For each newly fetched web page, the classifier evaluates the probabilities that this page belongs to each of the topics and gives the category with the highest probability.

The crawling process starts from a set of seed URLs for a target topic *c*. The fetcher pulls URLs from the priority queue and fetches the corresponding pages from the web. A newly fetched page *u* is sent to the classifier and the parser. The parser parses page *u* and extracts all the hyperlinks going out of *u*. The classifier calculates the probability that page *u* belongs to the target topic c, *P(c|u).* The outward hyperlinks from *u* are put into the priority queue with *P(c|u).* When URL *v* comes to the head of the queue and is actually fetched, we will evaluate *P(c|v)* and verify whether our guess from *u* to *v* has paid off. The fraction of relevant pages collected is called *harvest rate*. Suppose *V* is the set of pages crawled, *harvest rate H(c,V)* equals to the average value of *P(c|v).* Alternatively, we can also measure the efficiency of the crawl using *loss rate. L(c,V)* equals to the fraction of irrelevant pages collected.

The baseline crawler develops its crawl path as it selectively follows hyperlinks going out of visited pages. We illustrate the process in figure 4.2. The tree structure shown in the graph represents the hyperlink structure. When the crawler fetches a page *u*, it first evaluates *P(c|u).* If it is a good page, all out-links from this page are followed. In the graph, solid nodes are good pages with *P(c|u)* greater than *0.5* (0.5 is the threshold we use in our focused crawler). Shaded nodes are pages with probability *P(c|u)* smaller than the threshold. Out-links from shaded nodes are not followed. Blank nodes are pages the baseline crawler did not visit.

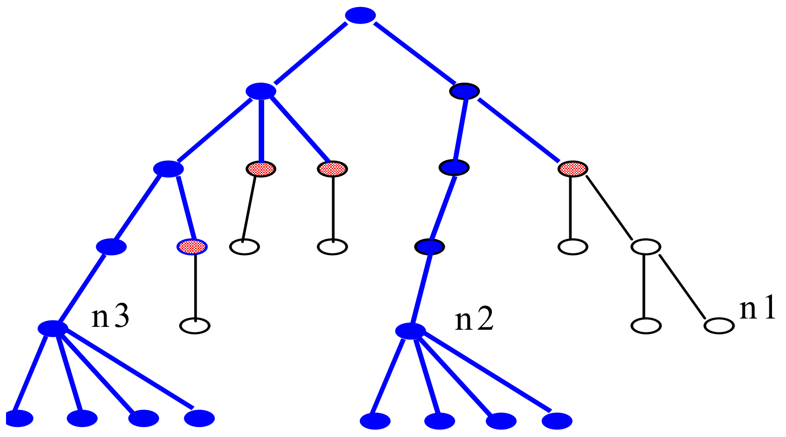


Figure 4.2. Crawl path for baseline crawler

From the crawl map, we can see two areas of potential improvement. First, nodes such as n1 may be good pages but the baseline crawler cannot reach them. This is because pages on the path to n1 are not related to the target topic. The concept of using the relevance of parent page to indicate the relevance of children pages makes pages like n1 hard to reach. Second, pages such as n2 and n3 require a long latency to locate. N2 and n3 are good entry points for a rich area. Not only is the route from starting pages to n2 or n3 long, lack of direct hyperlinks between the n2 cloud and the n3 cloud makes it hard to navigate from one to the other.

4.2 Central Questions

In this section, we address the following questions:

- Is hyperlink relationship the only structure we should follow in a focused crawl?
- Can we extract a useful topical crawl map from pages we have already seen?
- Can the topical crawl map extracted benefit future crawls of arbitrary topic?

Following hyperlink structure is the most natural way to proceed in focused crawling, but it is not always efficient. Web pages are usually created and maintained by numerous individuals. It is common that we do not see related pages pointing to each other. In some cases, web masters are reluctant to put explicit links to competitor's pages. In other cases, they simply do not realize the existence of related pages in the vast web. Therefore, traversing from an on-topic page to another may require a large number of hyperlink hops. Faith in the next visited page decreases significantly. The focused crawler is likely to stop exploring in the direction without reaching the other on-target page. Even if the remote on-target page is reached, there may be a large number of irrelevant pages downloaded already. We believe the situation can be greatly improved if topical links are incorporated in focused crawling. If a page has not only hyperlinks but also topical links that point directly to pages with similar topical interest, an enhanced focused crawler can quickly navigate through the topical link structure and dive into a pool of relevant pages.

Is it possible to build an explicit topical link structure? Studies have shown that topical locality exists in the web [8, 9, 10, 11]. We extract a topical crawl map from pages collected in past crawls, combine the topical crawl map with hyperlink structure to accelerate the crawling process and improve the harvest rate.

The topical crawl map benefits a focused crawl of the same topic by letting the crawler navigate through a network of highly relevant pages efficiently. The topical crawl map also benefits future crawls of related topics. When the enhanced crawler finds out that a fetched page is deviating from the target topic, it consults the topical crawl map for a redirection. For example, when the enhanced crawler encounters a “Football” page during a focused crawl for “Basketball”, it will be given a “Basketball” link extracted from the topical crawl map to redirect the crawl path.

We test our idea by performing focused crawls for a range of topics. We show that the topical crawl map not only speeds up the crawling process for the same topic, but also help future crawls of related topics.

4.3 Design

In this section, we describe the design of our enhanced focused crawler.

4.3.1 Enhanced focused crawler

Although hyperlinks show the linkage relationship among web pages, they do not capture the topical relationship in the web sufficiently.

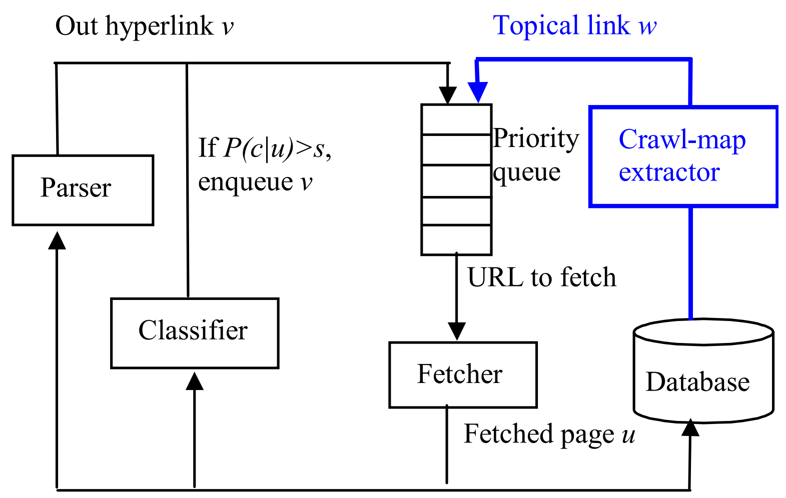


Figure 4.3. Enhanced focused-crawler

We extract topical relationship from the pool of pages we have crawled and utilize it in focused crawling. Figure 4.3 shows the components of our enhanced crawler. We add a crawl map extractor in addition to the baseline crawler. The crawl map extractor is responsible for gathering all pages from past crawls and performing topical relationship analysis on them. It generates a topical crawl map as output. The topical crawl map consists of topical links connecting pages with similar topical interest together. We store the topical crawl map in a MYSQL database on the same machine as the enhanced crawler. At crawl time, the enhanced crawler follows both hyperlinks parsed from the page and topical links selected from the topical crawl map.

There are two kinds of topical crawl maps. A same-topic crawl map is a <URL_A, URL_B> mapping. The mapping links URLs selected from previous crawls together by their topical relevance. All pages with similar topical interest are linked together. How to generate these links is discussed in the next section. We call these links “topical links”. When URL_A is fetched in a future crawl, not only its hyperlinks but also its virtual links (such as URL_B) will be put on to the task queue. The addition of the same-topic crawl map achieves potential improvement in one of the problem areas we point out in last section. We illustrate it in Figure 4.4. For nodes like n2 and n3, because they reside in highly valuable areas, there will be topical links pointing to them from the root and topical links connecting these two areas directly. Whenever a future crawl of the same topic occurs, the crawler can reach n2 or n3 much faster via the topical links and navigate from each other efficiently. We call this kind of links "navigation links". They serve the purpose of fast navigation among pages with similar topical interest.

Another kind of topical crawl map is related-topic crawl map. It is a two level structure. Table 1 on the next page shows an example. The first level is a <Topic_A, Topic_B, Portal_URL> triplet. The second level is the <URL_A, URL_B> mapping. Topic_A is the topic of a past crawl. Topic_B is the topic of future interest. Portal_URL records a URL as an entry point to the topical link structure. An entry of <Football, Basketball, Basketbal_URL> means Basketball_URL is a portal "Basketball" page found in a previous crawls for "Football". After performing topical relationship analysis on pages collected in previous "Football" crawls, the crawl map extractor builds topical link structure among all "Basketball" pages and makes Basketball_URL an entry point to access this group of pages. In a future crawl for "Basketball", when the enhanced crawler meets a "Football" page, it will consult the topical crawl map for a "Basketball" portal page. Instead of stopping at this "Football" page, the enhanced crawler will put the portal page on the task queue for exploration. And the topical links pointing from Basketball_URL will be added to the queue accordingly. The addition of the related-topic crawl map helps the crawler in the other problem area with a baseline crawler. In figure 4.4, for nodes like n1, it is likely that a crawl for a different topic *c'* encounters this page and finds that n1 belong to topic *c*. After performing the topical relationship analysis for that crawl, a record of < topic *c',* topic *c*, n1> will be added to the crawl map. When a future crawl of topic *c* happens, the enhanced crawler will redirect to n1 if it sees its ancestor. We call this kind of topical links “redirection links”. They redirect the crawler back to the right course.

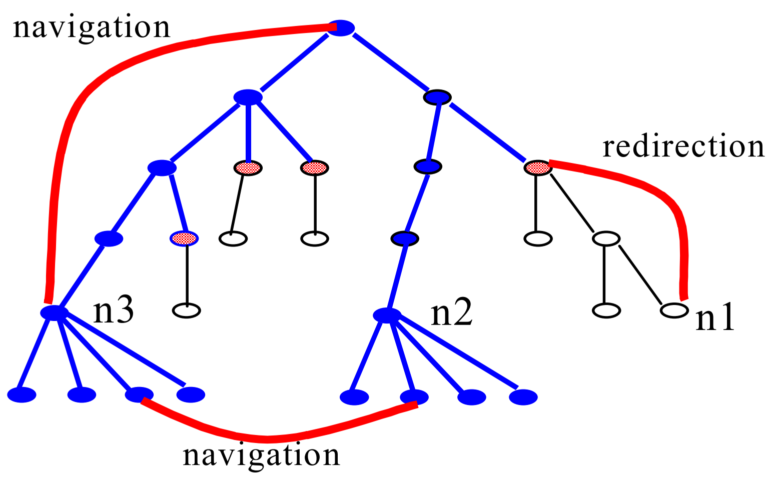


Figure 4.4. Crawl path for enhanced crawler

Table 4.1. Topical crawl-map structure

URL	Virtual Link
http://www.sandbox.com/	http://www.dukeupdate.com/
http://www.sandbox.com/	http://www.cnnsi.com/basketball/college/men/teams/abf/
http://www.dukeupdate.com/	http://www.miac-online.org/wombb.html
…	

Furthermore, we need to highlight the different usage scenarios for these two kinds of topical crawl maps. In a future crawl for the same topic, we only put topical links on the task queue when the crawler encounters the exact page that has been visited before. In a future crawl for a related topic, we loosen the requirement.

Whenever the enhanced crawler encounters a page belongs to the related topic we have crawled before, the process to add topical links is triggered. We do not require the crawler to meet the exact page that was crawled before. In other words, in a future crawl for “Basketball”, whenever a “Football” page is met, the portal URLs will be added onto the task queue. We do not require that the “Football” page has been seen in previous analysis. The crawl map may be entered at any point for the same-topic map, while it can only be entered via the portal URLs for a related-topic crawl map.

4.3.2 Topical crawl map generation

In this section we discuss how the topical crawl map is generated. We identify the hub pages in crawled pages. The detailed algorithm is as follows. Suppose *U* is the set of crawled pages for topic *c*, *c’* is the topic of future interest.

```
Function GenerateTopicalCrawlMap {
    1. Select subset $U'$ such that: for each $u \in U'$, $P(c'|u) > 0.8$
    2. Extract a sub-graph $G'$ on $U'$, considering only hyperlinks among pages in $U'$
    3. Calculate out-link counts $OC(u)$ for each $u \in U'$
    4. Select top $N$ nodes with the highest $OC$, we call them portal nodes. Make a
       complete graph among them. $N = max(2\% * total node, 10)$
    5. Connect each unselected node to one of the portal nodes uniformly yet satisfy-
       ing two requirements:
    6. Number of topical links for each portal node is smaller than its $OC$
    7. Each unselected node is linked to a portal node other than its direct parent
    8. If $N$ cannot satisfy above requirements, increase $N$ to $2*N$.
}
```

Figure 4.5 shows the structure of a topical crawl map. All the topical links in the graph are bi-directional. The solid nodes in the graph are portal nodes. Each of them has topical links to several pages of the same interest. During the crawling process, we want the topical links to participate side by side with hyper links. However, we do not want either of these two types of links to dominate. If hyperlinks dominate the crawling process, the focused crawler will degrade to a base-line crawler. If topical links dominate, the crawling process will follow mostly the topical links without exploring new areas pointed by the hyperlinks. Our design of the crawl map tries to maintain the balance between topical links and hyperlinks. Our algorithm ensures that the number of topical links for any of the portal node will not exceed the number of its hyper links. We also make sure that portal nodes point to pages distant from themselves. Therefore the crawler could be redirected to the remote and new areas of the web besides re-crawling the adjacent areas. We rank the pages by the number of topical links going out from them. Since we have complete information about the sub-graph, the rank calculation is straightforward without multiple iterations. A simple program can perform the calculation for a sub graph with tens of thousands of nodes in seconds.

After a new crawl is performed, newly found good pages can be merged into the existing topical crawl map right away. The crawl map extractor augments the sub-graph and performs the rank calculation once more. After each crawl, several sub-graphs in regard to different future topics are built or updated. By utilizing the topic matrix studied by Chakrabarti et *al.* [11], we only need to extract sub-graphs for topics closely related to the crawled topic and perform analysis accordingly. For example, after a “Football” crawl, we only need to perform topical analysis for topic “Football” and several closely related topics such as “Basketball” and “Baseball”. We can safely overlook non-related topics such as “/Science/Astronomy”.

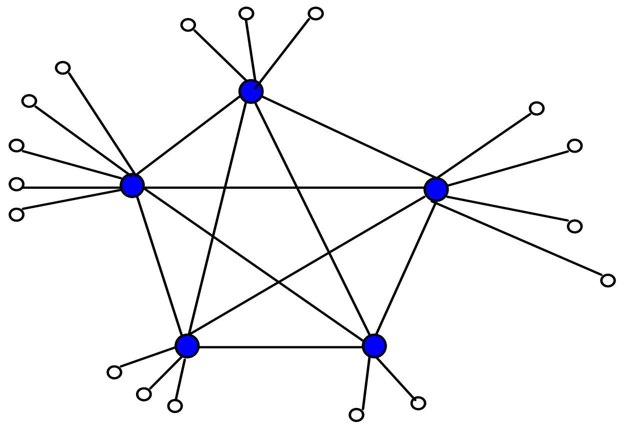


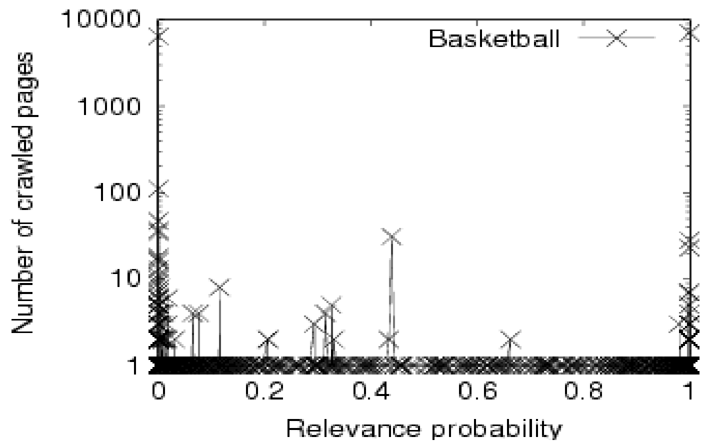
Figure 4.5. Topical crawl-map structure

4.3.3 Priority of topical links

The priority of topical links directly affects the order of unvisited URL frontier, thus controls the expansion of the crawl path. For hyperlinks, we set their priorities as *0.6* P(c|parent).* For topical links, we set their priorities as *0.6*. This definition shows that we view the topical links as having a perfect parent with *P(c|parent)* equal to 1. Setting the upper bound of hyperlinks' priorities to be 0.6 gives flexibility to adjust the relative importance between topical links and hyper links.

Such a definition reflects our design choices. How privileged do we think topical links are? They can be viewed superior to hyperlinks and should be visited before any hyperlinks are visited. If this is the case, we need to assign the highest priority to topical links. They can also be viewed inferior to hyperlinks, thus should be visited only after all the hyperlinks are visited. In this case, their priorities should be assigned to a lower value comparing with hyperlinks. Both views have some undesirable effects. By giving topical links absolute advantage, the crawler will always re-visit the pages in the topical crawl map first. Therefore it is more likely to explore paths that have been visited in past crawls. Exploration of unvisited paths will happen in very late stage of the crawl. This approach is effective at re-crawling highly relevant pages. Re-crawling is very important in focused crawling because the content and link structure on the web change frequently. However, it sacrifices the chance to find fresh, unseen pages early on. The latter approach gives topical links lower priority. It will greatly forfeit the power of previous knowledge and potentially degrade the enhanced crawler to a baseline crawler.

We view topical links as having equal importance compared to hyperlinks. We want the crawler to visit topical links along with hyperlinks. We want to see paths from hyperlinks and topical links exist simultaneously in the crawling process. We show in 4.6 that pages crawled by a baseline crawler exhibit bi-model distribution. We crawled 15,000 pages for both Basketball and Economics. In the set of pages our crawler visited, we observed the following pattern. If a page turns out to be a good page, its relevance probability is very likely to be close to 1.0. By assigning priority 0.6 to topical links, topical links are viewed equally with hyperlinks from a good parent page. This way, topical links and hyperlinks are naturally mixed in the crawl process. The goals of utilization of previous knowledge and freshness of newly crawled pages are satisfied at the same time. The priority of the topical links can be changed dynamically. We can increase the priority to 0.8 in order to favor re-crawling pages from the topical crawl map. We can also lower the priority to 0.4 in order to favor crawling fresh data.



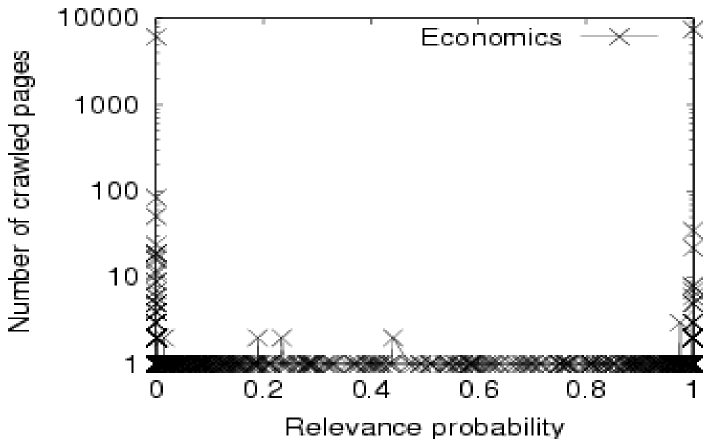


Figure 4.6. Bi-model distribution of relevance probability (15,000 crawled pages for each crawl)

4.3.4 Comparison among content caching, URL caching and topical crawl map

In this section, we compare our design with two alternatives: content caching and URL caching.

Content caching means after each crawl, the crawler stores all the pages crawled into their appropriate categories. In a future crawl, when a URL needs to be visited, the crawler will use its locally cached copy instead of fetching the page from the web. We believe such a simple caching solution is not suitable in focused web crawling for two reasons: First, web content changes constantly, the study conducted by [13] shows that on average a web page changes in several minutes. In such a highly dynamic environment, the performance gain from caching is not going to offset the loss of data freshness. Second, web pages change not only their textual content, but also their hyperlinks. For example, the CNN cover story pointing to a sports page about Red Sox's victory in one day may point to a political page covering the Presidential Election a week later. Because of the hyperlink change, a real crawl preformed in a future time may cover very different areas of the web. Content caching cannot capture dynamic changes in the web; therefore its effectiveness is limited.

URL caching means after each crawl, the crawler extracts the good pages from the result set and builds a giant start page containing all the good URLs. The crawler is assured of many good pages at the beginning of a future crawl for the same topic. This is useful when the main goal is to check the new content of the pages seen before. However, by giving previously seen pages top priorities, the crawler is not able to find fresh, unseen pages early on. The crawler needs to fetch tens of thousands of pages originating from the giant start page and links from the seed pages before it can freely explore other areas of the web. Treating the good pages found in previous crawls as an inseparable piece overwhelms the crawler and hinders its ability to explore unseen areas of the web.

We believe that utilizing history information and finding fresh data are both important in focused crawling. We do not push all the topical links onto the task queue at the very beginning. We mix topical links with hyperlinks gradually in the process of focused crawling by controlling the priorities. We can also dynamically adjust their priorities to favor the needs of re-crawling seen pages or exploring unseen areas. Mixing topical links and hyperlinks successfully is one of the central issues in our research. Our method is more dynamic comparing to content caching and URL caching in future crawl for the same topic. Besides benefiting future crawls for the same topic, topical crawl map also benefits future crawls for related topics. This idea is not seen in any caching method. In the case of future crawls for related topics, it is especially important to control the priorities. Suppose we have extracted 100 “Basketball” pages from a past crawl for “Football”. When we perform a future crawl for “Basketball”, we do not put these 100 pages extracted on the task queue at the beginning. We only put such topical links when it is necessary. We conjecture that “Basketball” pages extracted from previous “Football” crawl may not be as valuable as hub “Basketball” pages in the seed set. Only when the crawler hit a “Football” page and cannot explore any further in this path, we redirect the crawler to one of the “Basketball” pages extracted earlier. In this way, we ensure that the topical links are not competing with links from highly relevant page, thus cannibalizing the accuracy of the crawl.

4.4 Experiments and Results

4.4.1 Evaluation framework

For our experiment study, we used following modules:

- Crawler: We modify the w3c-libwww crawling library available at http://www.w3c.org/Library to include topical crawl-map extractor.
- Classifier: We use the public domain BOW toolkit and the Rainbow naive Bayes classifier created by McCallum and others [15]. Rainbow is fast enough to classify newly crawled pages right after they are fetched.
- Crawl-map generator: We write our own topical crawl-map generator in C, which takes responsibility of extracting sub graphs from crawled pages, updating topical crawl map and storing it in a MYSQL database.

4.4.2 Design of topic taxonomy

We download from Open Directory (http://www.dmoz.org) an RDF file consisting of 634,201 distinct topics. We parse the content RDF from the same site containing 4,130,596 sample URLs. We select "/Sports" and "/Science/Social _Sciences" as the top-level categories of our interest. We selected sub topics with at least 1,000 sample URLs. Table 4.2 shows the sub topics we select from "/Sports" and "/Science/Social_Sciences".

In each sub category, we select as many as 1,000 sample URLs and fetch these pages from the web. We also randomly select 5000 URLs from categories other than "/Sports" and "/Science/Socail_Sciences" and put them under directory "/Others". We use Rainbow classifier to build class taxonomies for "/Sports" and "/Science/Social_Sciences".

Table 4.2. Topic taxonomy

/Sports	**/Science/Social_Sciences**
/Baseball	/Anthropology
/Basketball	/Archaeology
/Cricket	/Economics
/Cycling	/Geography
/Equestrian	/Linguistics
/Football/American	/Political_Science
/Golf	/Psychology
/Hockey/Ice_hockey	/Sociology
/Martial_Arts	/Urban_and_Regional_Planning

/Motorsports	
/Paintball	
/Running	
/Skating	
/Soccer	
/Softball	
/Tennis	
/Track_and_Field	
/Volleyball	
/Water_Sports	
/Winter_Sports/Skiing	
/Wrestling	

4.4.3 Crawling the same topic

In this section we evaluate how much benefit we can harvest when the topic of interest does not change.

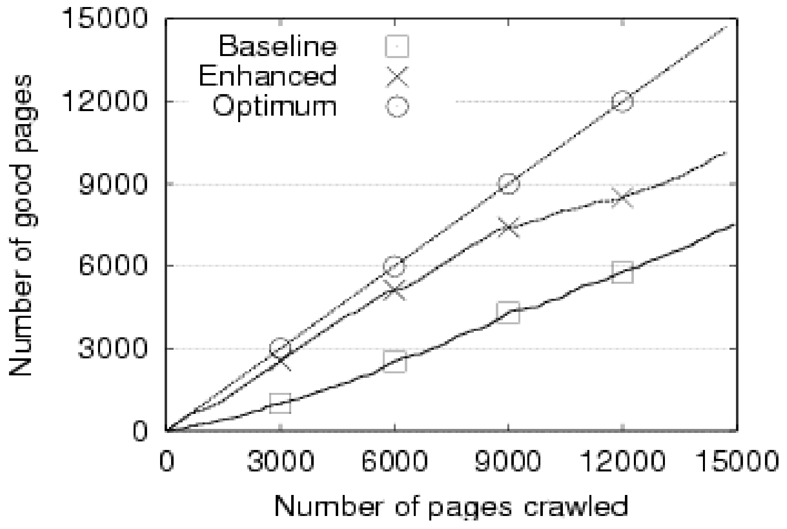
In figure 4.7(a), we show the result of crawling topic "/Sports/ Basketball". We let the baseline crawler crawl about 15,000 pages starting from 200 sample URLs randomly selected from the 1000 sample pages we fetched. Then we use our crawl-map generator to generate the topical crawl map from the result pages. There are 7,841 good Basket-ball pages extracted and linked in the topical crawl map. After that, we use our enhanced crawler to crawl again for the same topic, starting from the same seed set.

We plot the results for both the baseline crawler and the enhanced crawler in figure 4.7(a). X-axis shows the number of pages crawled. Y-axis shows the number of good pages, which are pages with relevance probability higher than *0.5*. The first curve from the bottom in figure 4.7(a) is the result from the baseline crawler; the second curve is the result from enhanced crawler. We also show the optimal curve (the diagonal curve in the graph) for comparison. The optimal curve represents the ideal situation when every crawled page is a good page. We observe that after starting off, the enhanced crawler quickly dives into a pool of relevant pages. In fact, at the beginning stage, the result from the enhanced crawler is very close to the result of an optimal crawler. This suggests almost all the pages that the enhanced crawler fetched are good pages. This behavior shows the benefit of the topical crawl-map. After the first crawl, we have built the topical crawl-map for the 15,000 pages fetched, linking good pages together via topical links. When the enhanced crawler starts from the same set of the seed pages, the topical links going out from these seed pages are be quickly put on the task queue of the crawler. In this way the enhanced crawler is supplied with a richer unvisited-URL frontier. Thus the enhanced crawler cuts out the intermediate steps and quickly visited a large collection of on-target pages. After fetching about 9,000 pages, the superior performance from the beginning started to downgrade towards a baseline crawler. This is because the initial benefit from the topical crawl map from the last crawl has been exhausted. The enhanced crawler starts to encounter unknown region of the web and it gets harder and harder to find on-target pages. One thing worth noticing is when the enhanced crawler crawled 9,000 pages, it is actually further away from the seed set comparing to the baseline crawler because it uses the topical crawl map to skip a lot of intermediate pages and reach further areas in the web. As we have discussed before, every jump in the crawler is a guess, the enhanced crawler is crawling a harder area comparing to the baseline crawler.

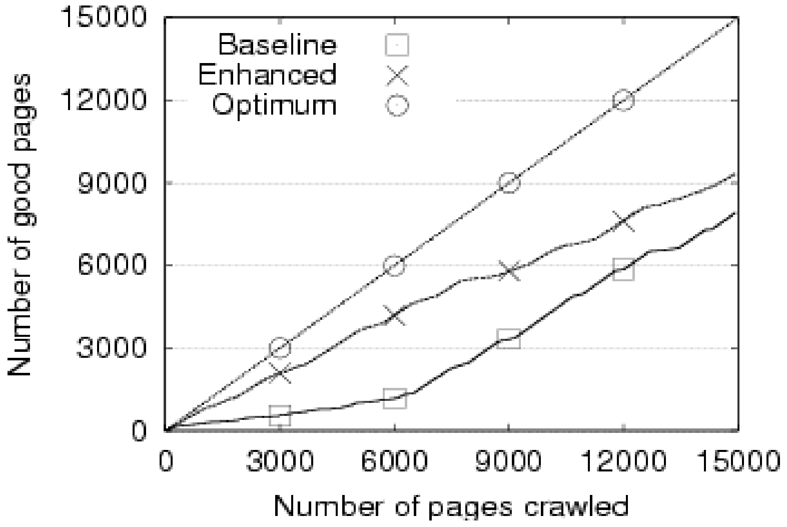
After crawling 15,000 pages, we see that the baseline crawler finds 7,841 good pages and the enhanced crawler finds 10,170 good pages. The enhanced crawler finds 2,329 more relevant pages. It translates to 30% improvement.

We perform the same experiment for "/Science/Social_Sciences/ Economics". The charts show similar trends with the Basketball crawl. We observe a big benefit at the beginning of the enhanced crawl. The benefit faded after reaching about 7,500 good pages. For this topic, the enhanced crawler outperforms the baseline crawler too. The baseline crawler finds 7,956 good pages and the enhanced crawler finds 9,360 good pages. The enhanced crawler improved the harvest rate by 18%.

As we have discussed earlier, the performance gain originates from the effectiveness of the "navigation links". By linking highly relevant pages together via topical links, the enhanced crawler is able to quickly navigate through the network of good pages and explore a better frontier. This directly shows in the high harvest rate at the beginning of the enhanced crawler. Since the number of pages in our collection is limited to 15,000 pages, the benefit of the topical crawl-map would be limited too. With a much bigger collection of pages, we hope to see the benefit of the topical crawl map not only exhibit it at the beginning of the enhanced crawling process, but also in later stage of the crawling process too. We imagine some of the topical links will lead the crawler to some rich area further away from the seed pages and provide guidance on navigating those areas.



(a) Baseball



(b) Economics

Figure 4.7. Same topic re-crawls

4.4.4 URL overlaps

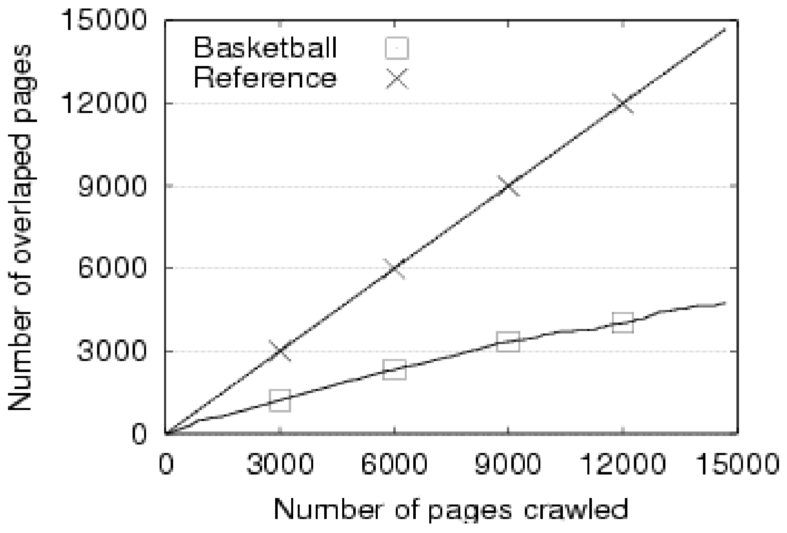
A general question one would ask is how does the enhanced crawler achieve its good performance? Is it merely benefiting from visiting pages it fetched before? What is the composition of the pages collected by the enhanced crawler?

In this section, we evaluate how well the topical crawl map guides the crawler by mixing hyperlinks and topical links in the crawling process. On one hand, if the crawler follows the hyperlinks in the frontier, it will crawl some of the pages visited by the baseline crawler. But in most of the cases, it will be led to different areas of the web. On the other hand, when following the topical links, the crawler will be led to on-topic pages early on and re-crawl the good pages seen by the baseline crawler first. The harvest rate would be high at the beginning since the crawler would meet more on-target pages by following the topical links. At the same time the crawler will be led to new areas not seen before, the harvest rate of these new areas are unknown. How to organize the priorities of hyperlinks and topical links will affect the collection of pages the crawler visits. We are particularly curious about what kind of effect the topical crawl map has on the ordering of the unvisited links in the URL frontier. From the pages fetched by the enhanced crawler, we look into the percentage of previous seen pages verses fresh data.

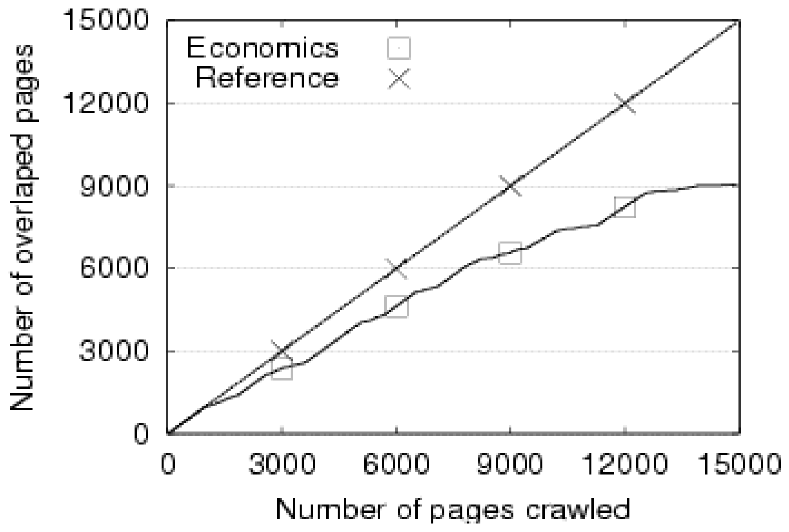
We plot the URL overlap for the experiments discussed in the last section. X-axis is the number of pages crawled. Y-axis is the number of pages that are seen in the baseline crawl. In figure 4.8(a), we show the URL overlap for the two “Basketball” crawls starting from the same seed set. We observe that the enhanced crawler is able to mix topical links and hyperlinks in the crawling process and balance the goals of re-crawling old pages and exploring new areas. We see that in the second crawl the enhanced crawler visits about 4,700 pages seen in the baseline crawl. 70% of the time, the crawler is visiting unseen pages.

In figure 4.8(b), we show the URL overlap for the two “Economics” crawls. We observe that the URL overlap is higher than it is in the “Basketball” figure. Out of the 15,000 pages crawled, 9,000 are pages seen in the baseline crawl. In the “Economics” crawl, 60% of the time, the crawler is re-crawling pages seen before and 40% of the time it is visiting unvisited frontier.

We believe this is because “Basketball” pages tend to have a large number of hyperlinks pointing to relevant Basketball pages. Therefore, in the task queue, the percentage of hyperlinks is much bigger than topical links. It causes the crawler to explore more hyperlinks than topical links and visit more new areas on the web. On the contrary, in crawls for “Economics”, “Economics” pages do not have a large number of hyperlinks. Thus topical links have a strong presence in the task queue at the beginning. The enhanced crawler crawls more pages through topical links initially, resulting in higher URL overlap.



(a) Basketball



(b) Economics

Figure 4.8. URL overlaps

4.4.5 Finding new pages

We have shown in previous sections that the enhanced focused crawler is able to find on-target pages quickly from its knowledge of topical linkage among crawled pages. Nevertheless, maintaining a good balance between finding on-target pages quickly and exploring new areas in the web is an important metric we would like to evaluate.

Being able to reach a lot of good pages consistently is very important in some of the usage scenarios. The ability to explore unknown areas on the web is crucial in other cases. A focused crawler that can extract the most recent changes on the web for specific topics would have wide applications. For example, a focused crawler that is able to find newly added resumes on the web with a computer science degree and interest in finance could be a useful application for recruiters. A focused crawler that is able to find the newly published papers on quantum computing around the world would be very valuable to researchers. These goals are not easy to achieve with a Google-like search engine. Since most search engines crawl the web without a particular topic, they will need to crawl a much bigger universe of the pages. This means the freshness of the search result cannot be guaranteed. It is different in the case of a focused crawler. A smart focused-crawler crawls the web on demand and returns fresh results in merely hours.

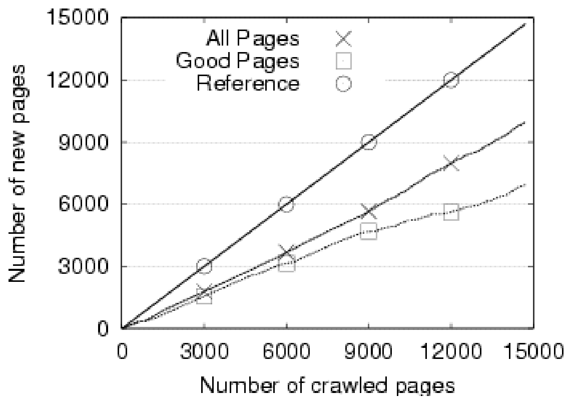
In this section, we evaluate our enhanced crawler on its ability to explore new areas in the web. Using the same experiment set-up, we made the enhanced crawler re-crawl 15000 pages for topics “Basketball” and “Economics”. Figure 4.9 shows how well the crawler performs in exploring new areas on the web.

In the chart, X- axis represents the number of page crawled. Y-axis shows the number of pages that have not been seen before. The curve in the middle tells how many pages that have been reached in the re-crawl are new pages. For topic “Basketball”, we see that out of the 15000 pages crawled; about 10,000 pages are new pages. The lower curve tells how many newly crawled pages are in fact on-target pages. In the case for “Basketball”, about 7000 newly crawled pages are good pages. This represents a 70% on-target rate. From the results of previous experiments, the baseline crawler finds 7,841 on-target pages out of the 15,000 crawled pages. The on-target rate for the base-line crawler is 52%. We can see that the enhanced crawler is more successful in crawl-ing unknown areas on the web. Such advantage is the result of the topical crawl map. The new areas pointed by the topical crawl map tend to be more resourceful than the areas pointed by the hyperlinks on the pages.

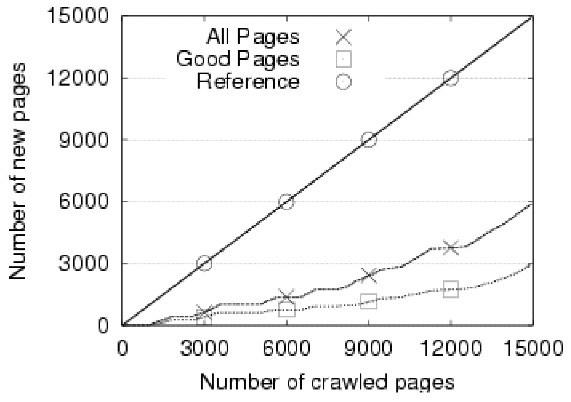
In the case for the “Economics” crawl, out of the 6000 new pages that crawler has found, about 51% of them are on-target pages. Comparing to the 52% on-target rate of the baseline crawler, the enhanced crawler shows comparable results without a big ad-vantage. By analysing the topical map of the “Economics”, we saw the Economics pages are rather clustered together. Thus, there are few topical links pointing to outside areas.

Since very little knowledge is passed to the enhanced crawler, the crawler behaves similar to a base line crawler in exploring the new areas on the web.

The enhanced crawler extracts topical information from pages it has seen before. It is intuitive that such information will be beneficial for future crawls of the same areas on the web. Our result shows that such information has benefit for the crawler to explore unknown areas on the web as well. We believe it is not the simple collection of the information but the interconnections among them that produce the advantage for the enhanced crawler



(a) Basketball



(b) Economics

Figure 4.9. Finding new pages

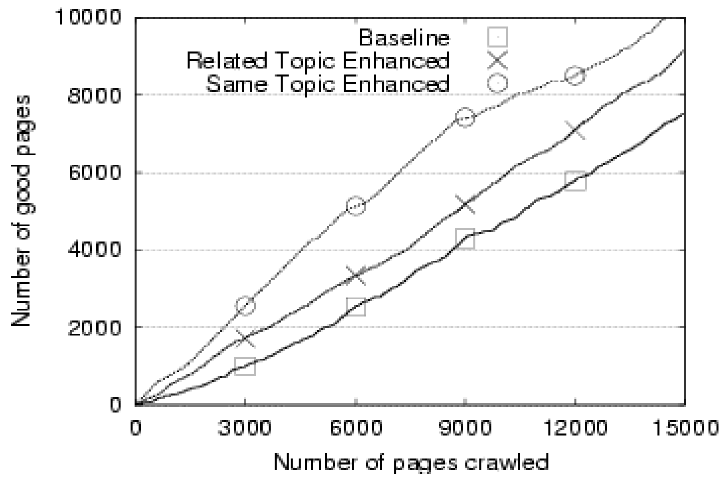
4.4.6 Crawling related topics

In this section, we evaluate how much benefit we can harvest by using the topical crawl map in crawls for related topics. An example of crawling related topic is described as follows: we first perform a crawl for topic "/Sports/Football/American" and extract topical crawl-map from the result pages. Then we perform another crawl on a related topic such as "/Sports/Basketball" utilizing the topical crawl map extracted earlier form the "Football" crawl. We show that topical crawl map extracted from the "Football" crawl can improve the performance of the second crawl on "Basketball".

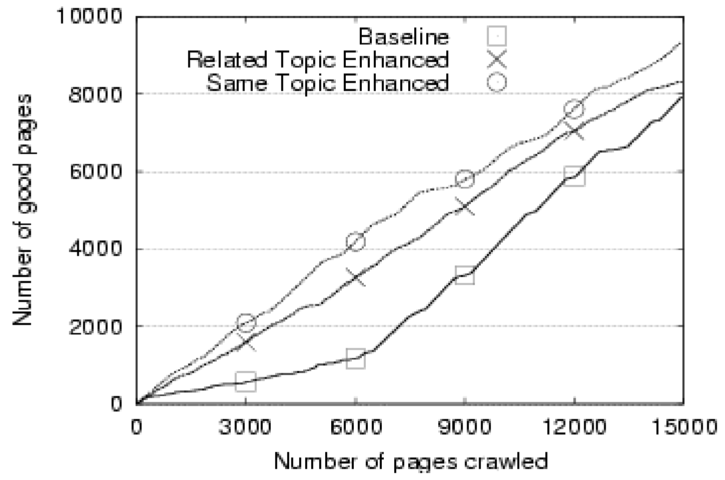
We conduct two sets of experiment. One is in the "/Sports" category, the other is in the "/Science/Social_Sciences" category. We first crawl 15,000 pages in a focused crawl for "/Sports/Football/American". From the pages crawled, we are able to extract 132 pages that are good "Basketball" page (with *P (Basketball | u)>0.5*). It is less than 1% of the total pages crawled. We build a topical crawl map on these 132 pages. Next, we perform a focused crawl for "/Sports/Basketball" with the help of the topical crawl-map we just built. We start the crawl using the same seed pages. Whenever a page within these 132 pages is reached, the corresponding topical links will be added to the task queue of the enhanced crawler and the crawler will be redirected to the areas pointed by the topical links.

We compare the result of this crawl with the result for baseline crawler. We show the result in figure 4.10(a). The first curve from the bottom is for the baseline crawler. The upper curve is the result curve for the enhanced crawler. We also include the same topic re-crawl results from last section for comparison. We see from figure 4.10(a) that the enhanced crawler finds more relevant pages very early on. During the whole crawling process, it consistently outperforms the baseline crawler. After crawling 15,000 pages, the enhanced crawler is able to find about 9,169 good pages. It is about 17% improvement comparing to the 7,841 pages found by the baseline crawler. It is to our surprise that with less than 1% redirection, the enhanced crawler improves the accuracy of the crawl by as much as 17%. It convinces us that monitoring the relevance probability closely in the crawling process is a decisive factor for the focused crawler. A smart crawler should not blindly explore as many pages as it can handle. It should closely look at every path it is exploring, prune the unpromising paths and redirect to the right course as soon as possible. We also observe that by using redirection links, the enhanced crawler has chances to explore areas that it would not normally reach in a baseline crawl. In the experiments, we see the enhanced crawler found good pages that are not seen by the baseline crawler. If we look at the 132 “Basketball” pages found in the “Football” crawl, we find that most of them are very good portal pages containing a large collection of Basketball links. Once a page in this collection is reached, it will lead the crawler to some rich area in the web. This further explains the seemingly large benefit from such a small topical crawl map.

In figure 4.10(b), we show the result of an enhanced crawl for “/Science/Social_Sciences/ Economics” with the help of the topical crawl map extracted from a previous crawl for “/Science/ Social_Sciences/Psychology”. The crawl map consists of 124 pages from the “Psychology” crawl that belong to “Economics” category. We see that the effect of redirection links is prominent throughout the crawling process.



(a) Basketball



(b) Economics

Figure 4.10. Crawling related topics

4.5 Discussion

In this chapter we introduced the concept of topical links. We created two kinds of topical links, navigation links and redirection links. We have conducted experiments exhibiting the effectiveness of the navigation links and redirection links separately in previous section. In production systems, the two kinds of topical links should be used together. Since navigation links and redirection links are used at disjoint situations in the crawling process, the combination of them could contribute even greater performance improvement for the enhanced crawler.

Topical links are as important as hyperlinks in the web. We suggest that the html source of the web page should include a section for topical links. An enhanced web page with topical links would look similar to the example we show in figure 4.11. Topical information can be maintained either by crawlers or by web-masters. An enhanced crawler can extract topical relationship from the pages it has crawled and push this topical similarity information to web masters. Web masters can thus keep explicit links pointing to each other and strengthen the structure of web community with similar topical interest. A previous study [14] shows that large percentages of emerging web communities are not aware of the existence of other members and are under-represented in web directories such as Dmoz. Pushing topical relationship to individual content-issuers would be especially useful in this sense. In the case that web masters are not willing to keep topical links for competition reasons, crawlers can take the responsibility of keeping these topical links. Such topical information will be valuable for customers looking for different options.

```
<HMTL>
<BODY>
<ORIGINAL SOURCE>
…
</ORIGINAL SOURCE>
<TOPICAL LINKS>
<TOPIC A>
<LINK> URL_A1 </LINK>
<LINK> URL_A2 </LINK>
</TOPIC A>
<TOPIC B>
<LINK> URL_B1 </LINK>
</TOPIC B>
…
</TOPICAL LINKS>
</BODY>
</HTML>
```

Figure 4.11. Enhanced web page source

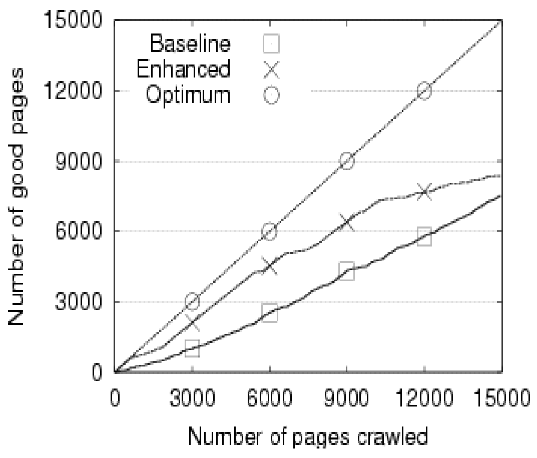
4.5.1 The evolving Web

In November 2005, the Internet fathers were awarded the Presidential Medal of Freedom in United States. Robert Kahn and Vint Cerf invented and implemented the first version of TCP/IP protocol in 1973 and gave birth to the Internet. Continuous developments of the Internet among the research community made it one of the most significant inventions in the history of technology advancement. Millions of people embark on the Information Highway via the Internet and make information available to the most remote area on earth. The World Wide Web has become the most important means of accessing information online. In the early days of the Web, the number of total web page was in the hundreds. Today, this number is in the billions. Google current indexes about 5 billion web pages and researchers believe it only accounts for less than 10% of the entire web.

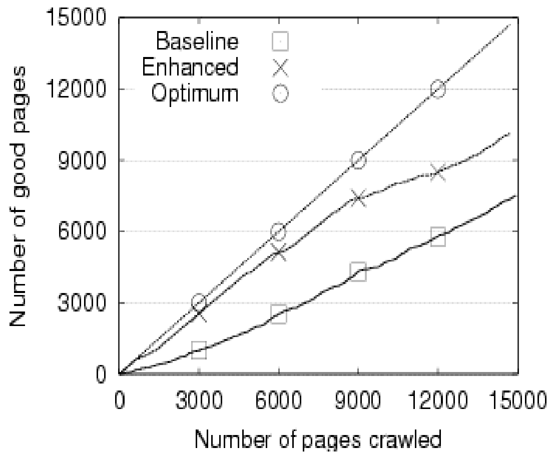
If we look at the web through a magnifier and trace its change, we would find some pattern from the web evolvement. A research group in IBM Almaden research center conducted a series of experiment to trace the changes on the web. [14] They focused at finding the communities (group of individuals who share a common interest, together with the web pages most popular among them) on the web and track the pattern of their changes. They categorized web pages in such communities into core pages and fan pages. Core pages are strongly connected parts in such communities. Their results suggested that due to the distributed and chaotic nature of the web, there are many more implicitly defined communities than those explicitly defined by online directory such as Google, Yahoo and Goecities. They estimated that there were about 130,000 communities on the web. Over a period of eighteen months, they found that only 30% of the communities that were found at the beginning of the period were fossilized. This suggests the majority of the web communities maintained a strong presence on the web. The core pages in these communities were long-lived. They were able to recover the communities in an evolved web through the core pages they collected at the beginning of the experiment period.

Their notion of web community largely reflects the social relationship of the web pages. The core web pages with higher number of fans correspond to the web pages with higher social capital. From their result, we hypothesize that web pages with higher social capital are rather stable during the web evolution and we should be able to find high quality topical pages in the web by simply using these pages as hubs to start. If this is true, the topical linkage information should survive the web evolvement and continue to contribute to the performance of the enhanced crawler.

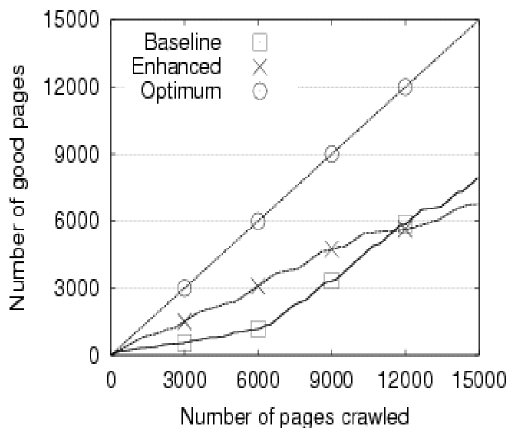
We conducted the same experiment six months later using the topical linkage information we found in earlier experiment. We show our result below.



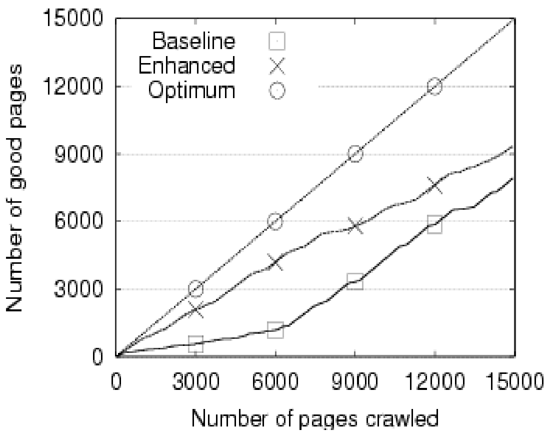
(a) Basketball: Six months later



Original plot



(b) Economics: Six months later



Original plot

Figure 4.12. Effects of the evolving web

Figure 4.12 shows four plots. The two plots on the left were results of the focused crawls we conducted six month later. We include the two plots from the original experiments for comparison. In the focused crawl for “Basketball”, we see that with the help of topical map gathered six months ago, the enhanced crawler is able to find more on-target pages compared to the baseline crawler. This suggested that the crawl map we generated earlier still provided useful information today. A large number of web pages we collected from the crawls six months ago are not only still valid itself but also provide links to good pages.

The result from a re-crawl of “Economics” shows a different story. From the plot, we see that the topical crawl map provides guidance to the crawler initially. The focused crawler was able to find more good pages compared to the baseline crawler. However, after crawling 12,000 pages, the performance of the focused crawler degraded. It was an interesting behavior. It suggested that the first level of the topical crawl map is useful, but following the links of these old pages does not yield good results. This could be resulted from the structural change of the web. The links from the old pages might not be valid anymore, causing the crawler to visit invalid pages.

4.5.2 Dependence on accurate classification

Classifiers play a crucial rule in focused crawling. Prior to the crawling process, a collection of seed pages is used to train the classifier. We expect the classifier to distinguish pages belong to the target topic from other pages. The common practice is to use manually built topic taxonomy such as dmoz.org or yahoo directories as the training set. Therefore, the classifier is fine tuned to provide the correct classification for such topic taxonomy. It is sensitive to not only the structure of the taxonomy, but also the seed pages within the directories.

During the crawling process, classifier gives the probability that a newly fetched page belongs to the target topic and decides about how to redirect the crawl path based on the classification results. A slight difference in the classification result could induce a major difference in the crawling path. The focused crawler would direct to different areas of the web if the results from the classifier were different.

Naturally we would like to choose the best classifier for our focused crawler. However, the definition of the best classifier is not crystal clear. We have studied several widely used methods in classification, including Naïve-Bayes, FIDF/Rocchio, Probabilistic Indexing and K-nearest neighbor. There are no proper measures to compare the classification results among them. As we have said earlier, the classification results depend keenly on the topic taxonomy and the training set. One classifier could be fine turned to achieve high accuracy for a particular training set. However, there is no guarantee that this classifier will work equally well for other kinds of test data. Because of the dynamic nature of the web, the composition of the web pages a crawler encounter differs largely from session to session. Although we can fine-tune the classifier for the collection of seed pages, there is no possible way to prepare the classifier for the collection of web pages it will encounter in the crawling process.

It is very hard to compare classifiers beyond the obvious measures using the same training data. We choose to make our implementation modular so that we can plug in different classifiers and compare the result. Upon evaluating several classifiers in our crawler, we found that there is no visible difference in the quality of web pages crawled. Therefore, we choose Naïve-Bayes method in our enhanced crawler since it is the fastest. We have tested the Naïve-Bayes classifier on a range of topic taxonomies from Open Directory Project. For a large portion of the topics in the taxonomies, the Naïve-Bayes classifier can provide higher than 50% accuracy in classification. (Our test data are fetched web pages from Dmoz directory. We assume 2:1 split for training and testing data.) We hope more sensible measures and detailed analysis for evaluating different classifiers will be developed among the research community in the near future. We will revisit this topic then.

4.5.3 Reliance on categorization

Everyone in the web crawling community knows the importance of start pages. Web contains massive amount of information, no single crawler is able to traverse the whole web within the time and space constraints. How to reach most valuable pages in the shortest amount of time is the central theme for web crawling. Past research has indicated that a smart crawler should not only selectively follow hyperlinks that would lead to diamond pages, it should also choose its start point very carefully. A lot of research efforts have been spent on fine-tuning the set of start pages for a web crawler. They follow forward links and backward links to extend the crawling frontier and calculate values for each of the pages in the frontier to choose the best seed pages to start. They have shown that a good set of start pages contributes significantly towards the quality of later crawled pages. Since start pages are the only definitive information that is fed into the crawler, fine-tuning is well worth the effort.

In the context of a topical crawler, we not only depend on the quality of start pages but also the quality of the categorization. The taxonomy of the topics is the input fed to the classifier. The accuracy of the taxonomy directly translates into the ability of the classifier to correctly classify newly fetched pages into different categories. It is fortunate that there are human built web directories such as Yahoo and Dmoz. These online directories give a first order of taxonomy for the classifier. Nevertheless, such online directories have their own limitations. First, the categories in their taxonomy are somewhat too broad to catch the fine difference for some of the topical groups. The hierarchy of such taxonomy does not represent the topical communities on the web accurately. They often underrepresented web communities that cannot be entirely categorized into their hierarchy. Second, since humans build the online directories, a substantial number of pages are not content pages to the crawler. They are portal pages with “obvious” links to content pages. It is natural for human to figure out the “obvious” link to follow in these portal pages, it poses extra complexity for crawlers and sometime confuses them.

Ideally, we would like to conduct an extensive general web crawl and capture a substantial portion of the web. Then, we can try to construct a categorization on this collection. If it was done successfully, we believe the quality and efficiency of our crawler will be largely improved. However, constructing such a categorization is not an easy task itself. In order to do the categorization right, we need great classifier again. However, great classifier relies on extensive amount of training data to initiate, which requires humans to manually classify an enormous amount of web pages in the end.

We believe a better categorization built by humans will be beneficial to the entire web community. More and more companies are entering the web crawling and searching territory, some company may be willing to spend a fortune to build such a categorization for competitive reason. It is realistic to think that more benefit can be harvested in the near future in the presence of better online categorization.

4.5.4 The combination of navigation links and redirection links

In previous sections, we have reasoned that both navigation links and redirection links are useful in topical crawler. We have evaluated these two types of links in our experiment sections. We learned from the study that navigation links are effective in re-crawling the same topic and redirection links are effective in quickly switching from topic to topic. However, we have not tested the behavior of the focused crawler if we combine navigation links and redirection links in the same experiment setting. This would be an interesting test to conduct. Intuitively one would think navigation links and redirection links serve different purpose and their usages are disjoint. Therefore the combination of these two types of links would generate even better results. We have not seen proof of such a claim. In the process of web crawling, the sequence of web pages visited is the most important factor of the success of the crawler. How navigation links and redirection links interact with each other affects the results of the crawler. Are they really disjointed? Is one kind of links dominating in the crawling process? These are the questions we can explore further.

4.6 Conclusion

In this chapter, we answered the question whether there are topical linkages in the web. We showed our approach to find such topical linkages. We developed an enhanced focused crawler to harvest the additional topical information in the web. We designed a set of experiments to evaluate the effectiveness of the focused crawler from different perspectives. We discussed the dependence on the classifier and initial categorization. We conclude our finding in following statements:

- Following topical link structure is very useful in focused crawling. We have shown that by following topical link structure, the enhanced crawler is able to find a richer set of relevant pages and achieve higher harvest rate.
- Topical crawl map can be extracted from pages we have crawled before. We have devised a method to create two kinds of topical links serving both fast navigation and topical redirection purposes. We have shown an efficient storage scheme for storing the topical crawl map.
- Future crawls for the same topic and related topics can both benefit from the topical crawl map extracted from past knowledge. In the case of future crawls for the same topic, the topical crawl map guides the crawler to fast navigate through the network of highly relevant pages. In the case of future crawls for related topics, the topical

crawl map closely monitors the deviation from target topic and provides redirection when necessary.

We consider building a more comprehensive topical crawl map covering a wide range of topics as a future direction. We would like to incorporate the topic citation matrix [11] in the creation of topical crawl map for related topics. Topical citation matrix is a matrix recording the pair-wise relevance among topics. We are also considering using textual features and distance measures in generating the graph structure for topical links. We will discuss the future works in details in chapter 6.

Chapter 5

BuddyNet: History-based P2P Search

5.1 Introduction

A decade after its birth, the Internet continues to deliver rapid growth and evolution in surprising ways. Peer-to-Peer (P2P) networks have become one of the fastest growing Internet applications [34] from the first introduction of Napster in 1999. Recent studies have shown a dramatic shift of the Internet traffic away from HTML pages to multimedia files shared in a P2P fashion. A March 2000 study at the University of Wisconsin found that the bandwidth consumed by Napster had surpassed the HTTP bandwidth [40]. Two years later, a University of Washington study showed that P2P file sharing dominates the campus network, consuming 43% of all bandwidth compared to only 14% for WWW traffic [41]. Without any doubt, P2P file sharing has already represented large portion of the Internet information needs and will continue to increase its dominance.

Today's P2P systems can be characterized into two classes. An *unstructured* P2P overlay network, such as Gnutella or Kazaa, builds an unstructured overlay network over the peers. A Gnutella-like system is simple and easy to adapt to dynamic situations when peers join and leave the system. Nevertheless, it is not scalable. When the number of peers increases, the number of messages propagated in this system increases dramatically and the latency to locate the content increases accordingly. Another class is *structured* P2P overlay networks. Most of them are based on the Distributed Hash Table (DHT) abstraction [25, 30, 31, 32, 33] . A DHT system organizes peers into a well-defined structure and controls the data placement and overlay topology. DHT's deterministic content locating and routing solve the scalability problem. However, DHTs require great effort to incorporate query models for keyword search [35].

The simplicity and adaptive features of unstructured overlay systems are very appealing for real-world P2P applications. The only obstacle is its scalability. Freely evolved P2P systems have shown tremendous similarity with social networks. User interactions and activities in P2P systems exhibit "small world" phenomena. [26, 27, 29] We believe that there is a way we can utilize these characteristics to make unstructured overlay systems scalable. Our design philosophy originates from a simple observation: If a peer has satisfied a large percentage of queries originating from another peer, this peer is more likely to satisfy future queries from the same peer. Looking at real-life experiences, we can see this simple observation being exemplified in various social contexts: people continue buying goodies from their favourite stores, people rent movies following the same reviewer's recommendation and on Ebay [39], people bookmark their favourite sellers, etc. We also see that such interest-based localities are being harvested for all purposes: Retailers are diligent at sending catalogue to their past customers, book clubs periodically select new books for their customers based on their pervious purchases, web pages become more and more personalized. If P2P systems have so much resemblance with social networks, it may also be true that interest-based locality exists in P2P systems. We see an analogy between the human society and P2P systems. Peers in a P2P system very much resemble individuals in a society. They possess not only physical capital but also social capital. In other words, besides of possessing a collection of files as its physical capital, a peer in a P2P system also possesses social capital in the form of its location in the network and its connectivity with other peers. It is natural to think that by recognizing and utilizing the social capital in P2P system, we would improve the efficiency of the entire system and serve the needs of individual peers better.

This simple observation sounds compelling, but to the best of our knowledge, there is no study that has proved its validity. Conducting an analysis on the behavior of existing P2P system seems to be the first step towards this goal. Links between peers exists for different reasons. Some links provide the basic connectivity of the system. Other links exist because of physical vicinity. We focus on links that exist among peers with similar interests. In other words, we are interested in the interest-based locality of P2P systems. We believe these links best captures the social relationship between peers and they are the key to improve the efficiency of the entire system. We conduct a novel evaluation study on Kazaa traffic focusing on the interest-based locality property. Our analysis validates our observation and shows us how to harvest the interest-based locality to improve performance.

Based on our findings, we propose a history-based search algorithm and a self-organizing topology adaptation mechanism, called BuddyNet. The proposed system has two desirable features. First, BuddyNet is a loose structure on top of the underlying overlay; it does not impose any constraints on data placement and topology. As a result, it does not affect the correctness of the underlying system; it only tries to improve the performance of the system. Second, the information kept at each peer directly benefits that peer. Peers do not need to keep arbitrary information or perform extra operations for the common good. This conforms to each peer's selfish behavior [27].

We discuss our evaluation study in section 5.2, system design in section 5.3, simulation model in section 5.4 and its performance in section 5.5. We conclude our work in section 5.6 and compare it with some related work in section 5.7.

5.2 Does Interest-based Locality Really Exist?

It has long been speculated that interest-based locality existed in P2P systems. Different schemes were proposed to harvest this kind of locality. [21, 24] However, to the best of our knowledge, there is no study performed on real world trace to validate this speculation. In this section we describe an evaluation study performed on a recently collected Kazaa trace and focus on the interest-based locality exhibited in this dataset. We display several facts about the interest-based locality in P2P system and give hints in designing a system using these findings.

We use data collected in a previous work [23]. The data collection process is as follows: A caching server is installed at the border between the local Kazaa user base of a large ISP and the Internet cloud. For each TCP connection, for both directions (in and out), a Layer 4 switch inspects the first few packets to detect Kazaa download traffics. If download traffic is detected then the switch redirects it through the caching server. Thus the caching server is able to log all downloads performed by local Kazaa users. The data collection period lasts for a year. There are no significant changes in traffic characteristics during this period. Therefore we use a part of the dataset for our analyses below. Table 4.1 summarizes the main characteristics of the collected data. *Consumer* describes a node that initiates download sessions. *Provider* describes a node that satisfies the query and provides the file for downloading. We use *peer, node* and *user* interchangeably in following sections.

Table 5.1. Characteristics of the collected Kazaa trace

Data collection period	2/5/03—2/11/03
Number of downloaded files	1.2 * 10^6
Number of unique files	~130,000
Number of consumers	>90,000
Number of providers	>190,000
Bytes transferred	~6TB

Figure 5.1 shows the activity distribution of the peers. Y-axis shows the number of downloads for each user. On X axis, users are ordered in decreasing order of the number of downloads they initiate. Logarithmic scale is used on Y-axis.

The activity levels for different users are widely varied. A few users issue as many as 10,000 requests; about 90% of the users issue less than 10 requests. Who answers these queries? Does a user get data from a large group of random users or does it always get data from a small group of focused users?

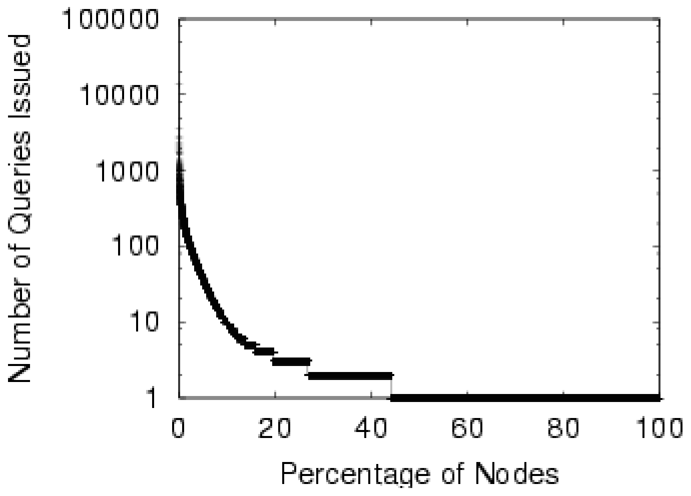


Fig. 5.1. User activity distribution

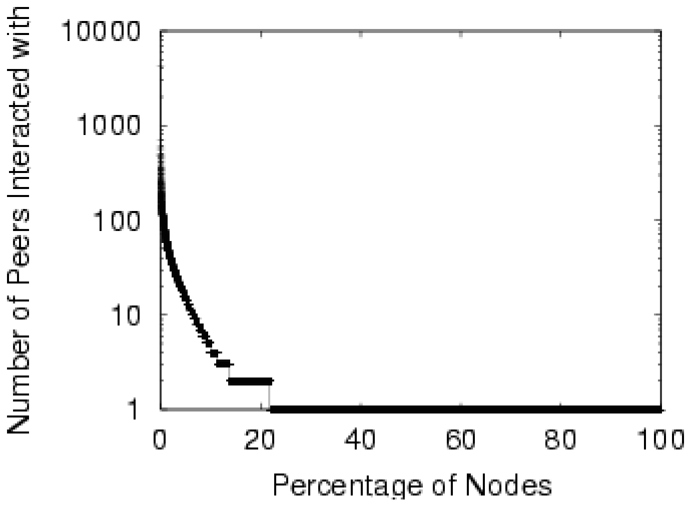


Fig. 5.2. User interaction distribution

We further look into how many different users a user actually gets data from. We plot the number of users from whom a user has downloaded data over the user name space in figure 5.2. Logarithmic scale is used on the Y-axis. From the graph, we see that more than 75% of the users have their queries satisfied by a single user. Over 85% of the users only need to keep two other users in their address book. About 95% of the users can satisfy their queries by asking less than 10 other peers. On the other hand, there are few users that get data from more than 100 other users.

From figure 5.2, we conclude that users can be classified into two categories. Users in the first category have focused interests and get data from a small group of other users. Recognizing other peers that supply content to them and establishing direct links to these buddy peers will most likely satisfy their future needs. Users in the second category have general interests and get data from a wide range of other users. It is not wise to keep direct links to all the peers that have answered its past queries. It is space inefficient to keep all of them in the address book and bandwidth consuming to ask all of them when future queries come. For users with general interests, we need to distinguish those buddies that have the highest probabilities to satisfy their future queries.

For each node, the probability that it will download from a node again if this node has satisfied N queries in the past is calculated as follows:

$$P(j,N) = \frac{\sum_{i\in G(j)} V(j,i,N+1)}{\sum_{i\in G(j)} V(j,i,N)} \qquad (1)$$

$$V(j,i,N) = \begin{cases} 1 & T(j,i) \ge N \\ 0 & T(j,i) < N \end{cases}$$

P(j,N) denotes the probability that peer *j* will download from another peer if peer *j* has downloaded *N* times from the same peer in the past.

T(j,i) denotes the number of times that peer *i* has satisfied queries issued by peer *j*.

G(j) represents the group of peers from whom peer *j* has downloaded.

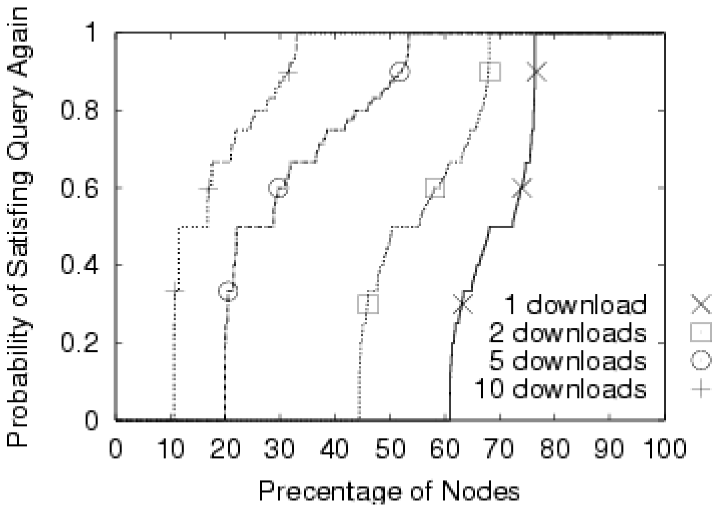


Fig. 5.3. Probability of satisfying query again

We plot the probability of satisfying a query again in Figure 5.3. X-axis extends over the user name space. Y-axis shows the probability of satisfying a query again. From the curve representing 1 past download (right most curve), when Y equals to 0.5, X is approximately 70%. It means that 30% of the users (from 70% to 100% on X-axis) will download from the same peer again with probability higher than 0.5. When there are 2 past downloads, about 50% of the users will download again with probability higher than 0.5. With more and more downloads in the past, such as 5 to 10 downloads, the probability to download again increases accordingly. With 10 downloads in the past, about 65% of the user will download from the same user again with probability 1. This result indicates that history information can help us identify those users that have the highest probability to satisfy future queries.

We happily see that our analysis result supports our observation that interest-based locality does exist in P2P system. We demonstrate that higher number of past downloads is a good indication for peers that have high probability to download from in the future. Building on our findings, we propose a history-based search algorithm and topology adaptation scheme. We let every peer keep direct contacts to peers that have highest probabilities to answer future queries. We call these peers its *buddies.* Peers consult their buddies first before asking the general public when they issue queries. From the above analysis, we realize that peers in the P2P system can be classified into two categories. On one hand, most of the peers get their queries satisfied with help of less than 10 other peers. On the other hand, a small percentage of peers do have wide interests and gets data from more than 100 other peers. We realize it is not the best strategy to make every node keep contacts to all of its buddies. How many direct links to its buddies should a peer maintains? If the number is too small, frequent insertion and deletion may occur when the actually number of buddies exceed the size of the list. If it is too big, it not only takes space at user side, but also consumes a lot of bandwidth when using these direct links to locate content. We need to find a suitable size that maintains the balance between these two ends. We perform following experiment on the trace. We first selected from the trace those nodes that issued at least 10 queries in the whole period. We make every node keep a buddy list with maximum size N. The least recently used (LRU) entry is replaced when the list is full. We test value 2, 5, 10 and 20 for N. We also test a special case where N equals to infinite, which means a node can keep as many buddies as needed. If a node in the buddy list satisfies a new query, we call it a “hit”. We plot the hit rate for different buddy list size in figure 5.4.

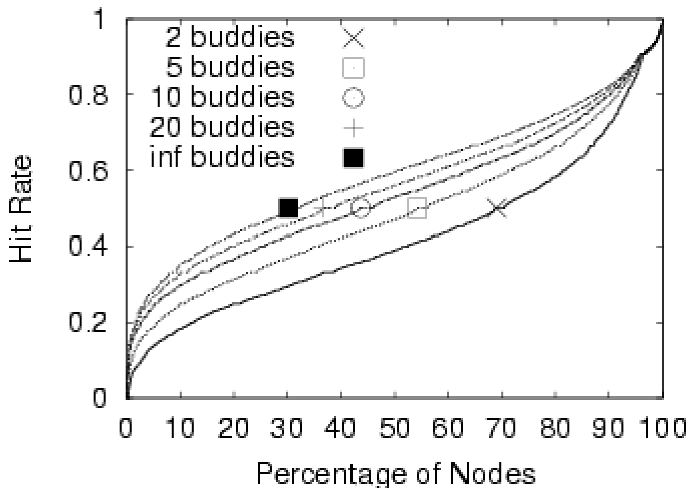


Fig. 5.4. Hit rate using different buddy list size

From figure 5.4, we see that with as few as two buddies in the list, nodes start to get benefit. Median hit rate is over 0.3, which means 30% of all the requests are satisfied by asking the two buddies. For the rightmost 10% of the nodes, the hit rates are over 0.8. The bigger the list size, the higher the hit rate. With list size equal to 20, the curve is very close to the optimal curve. We choose list size of 10 for our system. It not only provides decent hit rate, but also is small enough to add negligible overhead.

5.3 System Design

From our analyses in section 4.2, we show that interest-based locality exists in Kazaa traffic. Queries should be directed to a focused set of nodes instead of flooding to all the nodes. We propose a history-based scheme to meet this goal. With very little bookkeeping of the past query statistics, each node is able to identify a subset of nodes that can satisfy its future queries with high probability. Furthermore, peers cluster together by their mutual interests as more and more queries are issued in the system. The entire system evolves to a better-connected shape when time goes on. In following sections, we describe the design of system architecture and choice of algorithm.

5.3.1 BuddyNet Architecture

We add an additional layer on top of the peer-to-peer system's overlay. We call it BuddyNet. Besides keeping the links to their neighbors in the original overlay, peers also keep links to their buddies. Buddies are selected based on past query statistics. BuddyNet forms a loose overlay on top of the original unstructured overlay. The goal of the BuddyNet architecture is to let each peer contact those peers that have the highest probability of answering its future queries via these links, therefore decrease the system load and shorten the hop-by-hop delay. Furthermore, we adopt the one-hop replication technique into the BuddyNet architecture. The one-hop replication scheme used by Chawathe *et al.* [22] lets each node actively maintain an index of the content for each of its neighbors. We believe that instead of maintaining indexes for its neighbors, it is more appropriate for a peer to maintain indexes for its buddies. Storing indexes of neighbors' content requires a peer to maintain *arbitrary* data for the common good, while storing indexes of its buddies' content is directly self-beneficial.

Figure 5.5(a) shows the original unstructured overlay network. Peers communicate with each other via their overlay links. Figure 5.5(b) depicts two BuddyNet links for the top-left node. When this node issues a query, it tries to locate the content via BuddyNet links first. If the content is not found, the query is propagated in the system through the underlying overlay links. Figure 5.5(c) shows that of the two BuddyNet links, one link is in fact an index link (shown in bold), which means the top-left node keeps the index of the bottom-right node.

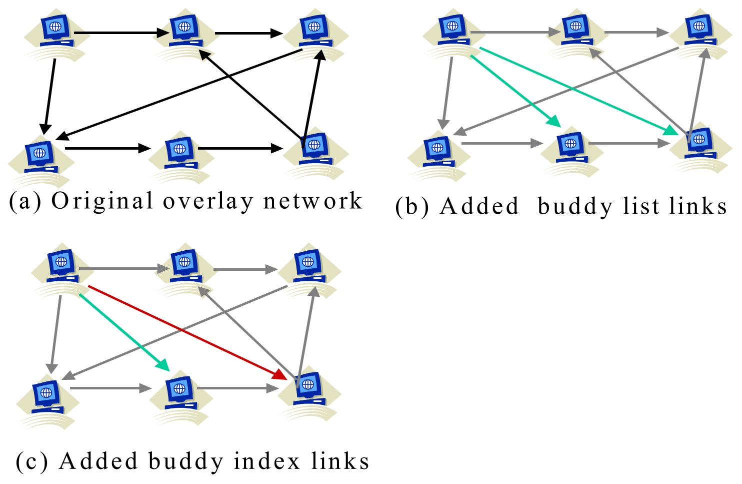

(a) Original overlay network
(b) Added buddy list links
(c) Added buddy index links

Fig. 5.5. BuddyNet architecture

5.3.2 Buddy List

Every node in the system keeps track of past query statistics by keeping a buddy list. The buddy list is a linked list. Every entry in the list is a tuple of (nodeID, response-Count). Response count records how many times a buddy has satisfied a query. The buddy list is sorted based on reponseCount and the age of the entry.

Whenever a node receives a response for its query, it checks its buddy list to see whether that responding node is in the list. If it is in the list, it increases the response count of that node and re-inserts it into the list. If it is not in the list, a new entry is created as (nodeID, 1) and inserted into the list. If buddy list reaches its capacity, the oldest tuple with the lowest response count is removed. Figure 5.6 illustrates the insertion and deletion process of buddy list with an example.

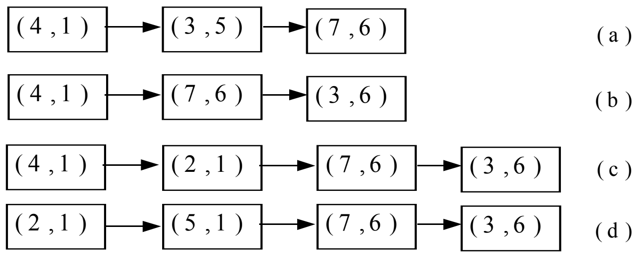


Fig. 5.6. Insertion/deletion for buddy list, size=4: a) Before b) After receiving a response from node 3 c) After receiving a response from node 2 d) After receiving a response from node 5

For different search methods, the buddy selection process may be slightly different. For a random walk scheme, we could insert the first response that comes back. For a flooding method, we could choose a response to insert randomly from all the responses received. We can also insert all the responses into the list. Currently, we insert the first response into the buddy list. We explain how the buddy list adapts to load imbalance in a later section.

For any replacement algorithm, the main concern is whether the workload shows characteristics of sequential access. In our case, if a node continues receiving responses from a large number of peers randomly, its buddy list will constantly add new entries and expire old entries. We show in our simulation that after a short warm-up period, the buddy list becomes stable. Newly created entries only affect the head of the list. The tail of the list is relatively stable with buddies having high response count.

5.3.3 Buddy Index

Besides the buddy list, every node also keeps indexes for a subset of nodes. In order for node A to keep node B's index, node B must satisfy two requirements. First, node B must be in node A's buddy list at present. Second, node B must be in node A's buddy list for long enough time.

We checkpoint the buddy list at fixed interval. At every checkpoint, a special procedure is invoked to check and update the status of the buddy list and request the index from its buddies if needed. For every node in the buddy list, it checks whether it was also in the buddy list at the last checkpoint. If it was and we do not already have its index, an INDEX_REQUEST message is sent to the node. Upon receiving the INDEX_REQUEST message, the node sends its local index. We log the status of the buddy list to be used at the next checkpoint. When a node is removed from the buddy list, its index is also purged.

After the first transfer of the index, nodes periodically exchange indexes in an incremental fashion. An incremental index transfer can also be triggered when a peer finds out that its copy of a buddy's index is out-of-date by an incorrect response to its query or failed downloads. The communication of an index transfer is a point-to-point message with tens or hundreds bytes message body. While the communication cost of a query is tens or hundreds of message with a several byte message body.

Transferring indexes among peers can be an expensive operation. If it happens often, it will increase system load and cause the system to perform badly. We believe that since a peer only shares its index with peers that have similar interests, its buddy list will become stable after a short warm-up period. Transfers of indexes among peers will be very infrequent and utilize only a small fraction of available system resource. Our simulation result supports this hypothesis.

5.3.4 Dynamic Adaptation

BuddyNet dynamically adapts its structure based on successful or failed query response. When a response is received via buddy index or buddy list, the peer will subsequently attempt to download the file from this buddy. At this time, if it finds out that this buddy is down, it decreases the response count for this buddy in its buddy list and re-inserts it at the correct position. In this way, buddy list monitors the stability of the buddies. Under dynamic situations when nodes join and leave frequently, using the buddy list can shield peers from the instability of the network. It also prevents popular nodes from being overloaded. If a lot of nodes contact the popular node via their buddy links, the popular node can choose not to respond to some of the queries. The requesting nodes will treat this node as down and move it to a lower priority position in its buddy list and send queries to other buddies in its buddy list. Load balance is thus achieved.

5.3.5 Search Algorithm

A query is propagated in our system as follows:

When a node issues a query, it first checks its buddies' indexes kept locally. If the content is not found, it asks all the nodes in its buddy list by sending out a QUERY message with Time-to-live (TTL) equal to 1 to each of them. If no node in its buddy list responds to the query, it performs a random walk on its neighbors to locate the content.

Upon receiving a query, each peer checks its local storage to see whether it can satisfy the query. If it has the content, a RESPONSE message is sent back. If the content is not found in its local storage, it checks its buddy index. If one buddy's index satisfies the query, a RESPONSE message is sent back on behalf of that buddy. If it fails, the node uses a random walk to forward the query as long as the TTL of the message is greater than zero. After sending out a request, a node may receive multiple responses. It chooses which node to download from and sends out a FETCH message to that node directly.

Sending extra messages to peers in the buddy list is only used at the query-originating node, not at intermediate nodes. In this way, even in the worst case, a single query will only generate as many extra messages as the size of the buddy list (we choose it to be 10) compared to the random walk scheme. We do not require any node in the system to take extra responsibility to delegate queries for others.

5.4 Simulation Model

In this paper, we use a trace-driven simulator on a subset of the dataset described in pervious sections. We randomly select 2,000 users who issue more than 10 queries in the trace. We filter out the sub-trace that contains only queries initiated by these users. There are 23,262 queries issued in total. There are 7,091 providers that supply the data. Filtering out providers that do not supply data decreases the network size. A smaller size network can make the simulation run faster. However, filtering out non-participating nodes does not bias the experiment results. This is because the alternative systems (FLOOD and RANDOM_WORK) we are comparing to degrade their performance quickly when the size of the network increases. By reducing the simulation size, we were actually favoring our comparison systems. In the sub-trace, consumers only request content; they don't supply contents to other nodes. Providers only supply contents; they don't originating queries. This separation does not affect our results since the querying behavior and the responding behavior are orthogonal. We can easily merge a consumer and a provider to make a full functioning node. We think the separation makes the simulation results easier to interpret.

Our simulator proceeds by having peers issue queries sequentially. At any time, a peer *i* in the network may be actively issuing queries, responding to queries or down. Upon issuing a query, a peer waits for incoming responses. Since we are interested in relative system load and average path length to locate the content, not absolute time, having queries executed in sequential order does not affect the correctness of our performance evaluation.

Freely evolving P2P networks have been shown to exhibit power-law network characteristics [37]. Hence, peer degrees in the simulation are governed by a power-law distribution. Upon joining the network, a peer connects to a node *i* with probability $\frac{d_i}{\sum_{j \in N} d_j}$, where N is the set of nodes currently in the network and d_i is the node degree of peer *i*. Every peer in the system maintains a minimal degree. Figure 5.7 shows the distribution of node degree for the simulated overlay network of 9,091 nodes with minimal degree equal to 4. Both X and Y-axes are log scaled.

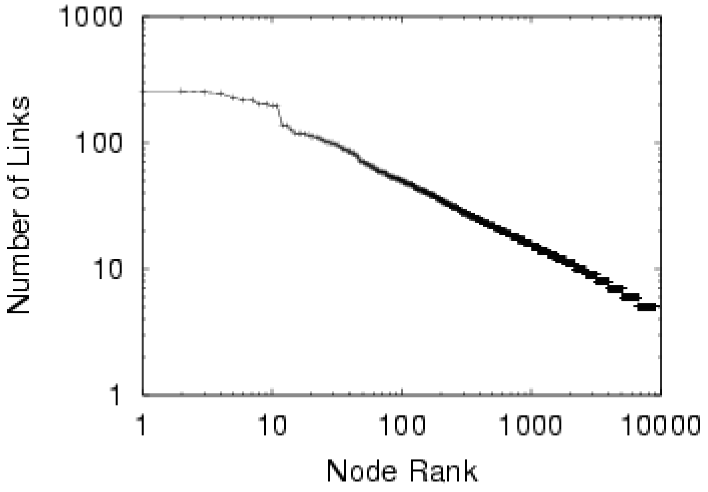


Fig. 5.7. Distribution of node degrees

Figure 5.8 shows the content distribution of the simulated system. For each individual file in the system, we count how many peers possess a copy of the file. We plot the number of copies for each file. We see that in the simulation, file distribution is governed by a Zipf distribution. We assume a pool of N peers, and each peer has a certain probability of being online, assigned based on the statistics collected in [27].

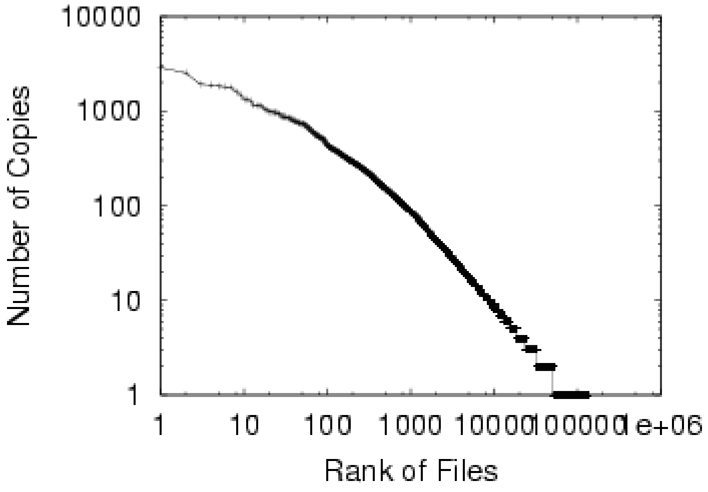


Fig. 5.8. Distribution of files

5.5 Performance Evaluations

5.5.1 Performance comparisons

In order to compare the performance of our algorithm with other search algorithms, we look at the following metrics:

1) **Msg/query (M/Q)**: The average number of messages propagated throughout the system for each query, which represents the system load. Whenever a node receives a message, we increase the total number of messages in the system by one. If seven nodes see a message, they will each count the message once using our metric.
2) **Path length (PL):** The average hop counts to reach the **first** response for a query. If multiple queries are issued to different peers before the first response comes back, all these queries will be counted.
3) **Success rate (SR):** The percentage of queries that have gotten responses.

We compare the performance of following three search algorithms.

1) **RANDOM_WALK**: Query is forwarded to a randomly chosen neighbor until the first response is received. Originating peer sends out 16 walkers at a time with the TTL (time-to-live) equal to 1024.

2) **INTEREST_SHORTCUT**: Nodes use the interest-based shortcuts scheme described in [24]. Peers keep shortcuts to other nodes that have similar interest and follow these shortcuts in query propagation. It is similar to our buddy list design without the buddy index and the dynamic adaptation feature.

3) **BUDDY_NET**: Peers keep track of past query history, including the buddy list and the buddy index, and perform history-based search during query propagation.

5.5.1.1 Baseline performance comparison

In the base line case, we assume all the nodes are always active and respond to all the queries they receive. We discuss performance comparison under dynamic situations later. We run the simulator for the first 10,000 queries to warm up and record the result for queries 10,001 to 20,000.

Table 5.2. Baseline performance comparison

	RANDOM WALK	INTEREST SHORTCUT	BUDDY NET
M/Q	3,842	1,370	1,122
PL	88	31	28
SR	0.97	0.98	0.98

From Table 5.2, we see that BUDDY_NET outperforms both RANDOM_WALK and INTEREST_SHORTCUT.

Using the BUDDY_NET algorithm, the system load is reduced by 70% compared to RANDM_WALK and by 20% compared to INTEREST_SHORTCUT. We believe the benefit coming from the use of the Buddy List. By asking buddies in the buddy list first before propagate the query to all the neighbors, the originating node is likely to get to a responder within a much smaller group. Thus, the number of messages generated in the system is largely reduced. At the same time, BUDDY_NET also achieves a lower average path length.

5.5.1.2 Performance under dynamic situations

In this section, we discuss the effects of participation dynamics.

In the real world, peer-to-peer systems are highly dynamic, with nodes joining and leaving constantly. Participation dynamics can affect both the system load and the latency to locate the content. Nevertheless, BuddyNet is designed to adapt to the participation dynamics in peer-to-peer networks. Short-living nodes are more likely to age out, and stable nodes are more likely to accumulate higher response counts and stay in the buddy list. In this way, the buddy list is also an indicator of node stability. When a peer asks its buddies about a query, it is more likely to reach a responsive node. The BuddyNet algorithm greatly reduces the scope of query propagation. This is particularly useful in dynamic situations. A query is more likely to be resolved within a small group of nodes; therefore, the dynamic behaviour at other nodes has small probability to affect the performance of the query.

In Table 5.3, we show the simulation result under dynamic situations. In this simulation, we do not assume all the nodes are always up and respond to queries. Instead, we allow nodes to join and leave the network frequently. We use the same setting as in last section, but assign each peer's uptime based on the uptime distribution in [7].

Table 5.3. Performance under dynamic situations

	RANDOM WALK	**INTEREST SHORTCUT**	**BUDDY NET**
M/Q	6,607	4,419	2,299
PL	139	78	26
SR	0.88	0.87	0.92

We see that in dynamic situations where nodes join and leave frequently, both RANDOM_WALK and INTEREST_SHORTCUT schemes degrade significantly. Both schemes generate significantly more messages; have longer path length and their success rates are dropped.

In the case of RANDOM_WALK, having peers behave dynamically in the system means the success rate of the walkers are greatly decreased. If any peer on the chain from the originating node to the responding node leaves the network, the response would not be found and returned to the originating node. Thus, new walkers need to be sent out and more messages needs to be propagated in the system. As the result, the path length of the query increased a lot.

In the case of INTEREST_SHORTCUT: although peers keep shortcuts to other peers with similar interest, they do not adjust these shortcuts dynamically based on the behavior of the peers in the list. When the peer-to-peer network experience a lot of instability, the shortcuts cannot shield peers from it. Queries are sent to inactive peers and need to be redirected later. This explains the large increase of the system load and the path length.

On the contrary, BUDDY_NET still provides reasonable performance. We believe this is the contribution of the dynamic adaptation mechanism in BuddyNet system. By adjusting the buddy list dynamically based on the behavior of the peers, we tend to keep the active nodes in the list and age out the inactive nodes. It successfully shields nodes from the instability in the network and achieves robustness against participation dynamics.

5.5.1.3 Performance under different replication factors and system sizes

Replication factor and system size are two important characteristics affecting the performance of a P2P system. Generally speaking, when the replication factor is high, meaning there are more copies of the same file in the system, the performance of the system will improve. We would like to verify how much benefit we can get from the BuddyNet system under different replication factors. It is easy to imagine that BuddyNet can perform well under decent replication factor since it is relatively easy to locate content. Does it perform well under low replication factor?

The most important problem of P2P system is scalability. No matter how well an algorithm works in a small setting, extending it into a large-scale system is the real test. In a confined setting with a small number of nodes, it is possible to sacrifice system bandwidth to achieve desired result. However, such an algorithm will not be successful in a large-scale system since the number of the nodes will magnify the expense of handling extra messages. We look at how BuddyNet performs under different replication factors and system sizes. Since FLOOD is 4 orders of magnitude worse in terms of system load than RANDOM_WALK and BUDDY_NET, in this section, we mainly compare the performance of RANDOM_WALK and BUDDY_NET under different system sizes and replication factors.

Figure 5.9 shows the system load and average path length for different algorithms at replication factor 0.01, 0.1, 0.5, 0.75 and 1 with system size of 1000 and 5000 nodes. Replication factor is the number of nodes that possess a file divided by the number of all nodes averages across the collection of files in the system. For example, replication factor 0.5 means on average for every file in the system, half of the nodes have this file locally. We also show the hit rate for BUDDY_NET algorithm under different settings. The simulation setting for the network with 1000 nodes is the same as in section 5.2.1.1. For the network with 5000 nodes, there are 262,404 files in total, distributed into 300 categories. Every peer maintains at least 3 neighbors and is interested in at least 5 categories. We run the simulator for 1000 cycles to warm up, and record the result for cycle 1001 to cycle 1100. There are 219,493 queries issued during these 100 cycles.

From Figure 5.9, for the simulation with 1000 nodes, we see that at replication factor 0.01, the system load for RANDOM_WALK is 2,015 M/Q. It needs 172 hops on average to locate the content. For BUDDY_NET, the system load is 130 M/Q and the average path length is 8.02 hops. 50% of the queries hit in the buddy list or the buddy index. BUDDY_NET achieves 15.5 times reduction in terms of system load and 21.5 times improvement in search efficiency.

The system load and average path length decrease when replication factor increases. At replication factor 1, the system load for RANDOM_WALK is 428 M/Q; the average path length is shortened to 31 hops. Again, path length is the number of messages issued before the first response comes back. In the case of RANDOM_WALK, the originating peer always issues 32 queries in parallel. For BUDDY_NET, the system load further decreases to 2.47 M/Q, and the average path length is shortened to 0.4 hops. The hit rate achieves 99.8%. BUDDY_NET again outperforms RANDOM_WALK by 173 times reduction in system load and 77.5 times improvement in search efficiency.

For the simulation with network of 5000 nodes, at replication factor 0.01, the system load for RANDOM_WALK is 7,628 M/Q; the average path length is 366 hops. The system load for BUDDY_NET is 1,070 M/Q and the average path length is 75 hops. The hit rate is 31%. At replication factor 0.1, 0.5 and 0.75, the system load and the average path length keep decreasing for both algorithms. At replication factor 1, the system load for RANDOM_WALK is 1,222 M/Q; the average path length is 88 hops. The system load for BUDDY_NET is 3.8 M/Q and the average path length is 0.34 hops. In this simulation, we only run the simulator for 1000 cycles to warm up because it takes a long time, usually several hours. We believe that if we warm up the simulator for a longer period, the hit rate of the BUDDY_NET algorithm will further increase to about 50%. Therefore, the system load and the average path length will be further improved.

From these figures, we can see that under higher replication factors, both algorithms achieve low system load and short average path length. However, at larger system scale, RANDOM_WALK needs to propagate more messages throughout the system in order to find an answer. On the contrary, BUDDY_NET generates considerably less messages to locate the content. BUDDY_NET method is more likely to scale.

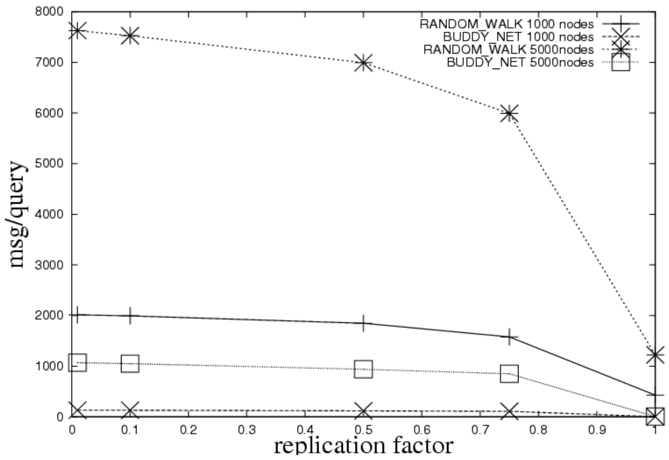


Fig. 5.9(a): System load

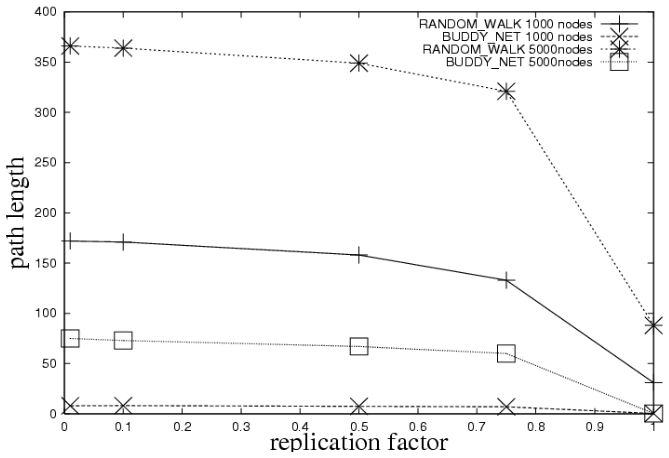


Fig. 5.9(b): Average path length

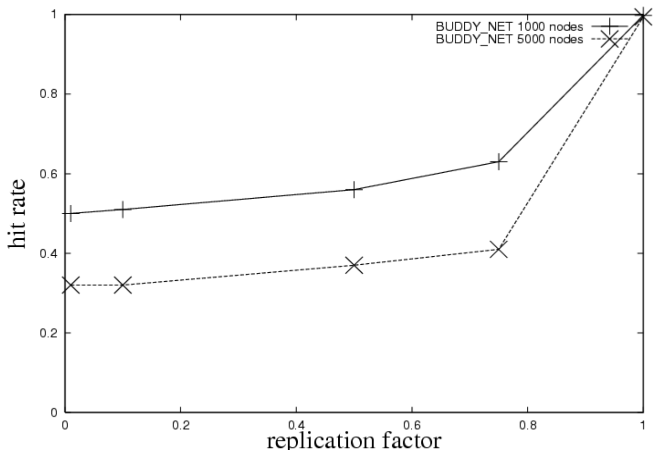


Fig. 5.9(c): Hit rate

Figure 5.9. Performance under different replication factors and system sizes

5.5.2 Analysis of History-based Algorithm

Our proposed system achieves its effectiveness by keeping track of past query history. However, keeping states in each node and exchanging information among nodes may be expensive. In the following section, we show that in our system we achieve good performance with every node keeping a very small amount of history information. In addition, the information exchange among nodes happens infrequently and uses a very small fraction of the available system bandwidth. In this session, we refer to the simulation setting in section 5.5.1.1.

5.5.2.1 Effect of Keeping History Record

Every node in the system keeps two kinds of history information. First is the buddy list, which is a linked list that keeps track of its buddies. In our simulation, we choose the maximum size of the list to be 10, which yields very good performance. The second part is the buddy index. If peer A has stayed long enough in peer B's buddy list, peer B will request peer A to send its index to B. Since every node can only hold indexes for nodes in its buddy list, at any time, every node will hold indexes for at most 10 peers. However, if the buddy list is not stable, nodes might end up requesting index from other nodes and aging it out quickly. We show that in our simulation, the buddy list reaches a stable state quickly and transfers of indexes are kept to minimum.

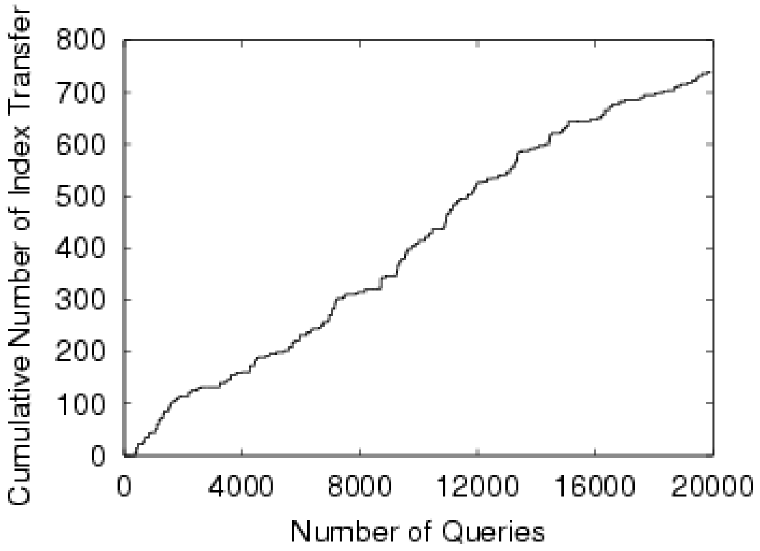


Fig. 5.10. **Number of Index transfers vs. number of queries**

Figure 5.10 plots the number of cumulative index transfers against the number of queries issued in the simulation. We can see that at the beginning of the simulation, nodes quickly accumulate buddy indexes and gather information for their buddy list. For the first 2000 queries, there are about 100 index transfers in the system. Which accounts for 5% of the total queries. For the 10,000 warm up queries, 408 index transfers incurs. The percentage of queries used for index transfer is about 4%. For query 10,001 to 20,000, there are 332 index transfers in total. Index transfer accounts for 3.3% during this period. We can see that after more and more queries are propagated in the system, most of the nodes built up a stable buddy list and require less and less index transfer. Since the peers dynamically adapt their buddy list, some aged buddies will be dropped from their buddy list and new buddies added. Index transfers will continue to happen, but accounts for a very small percentage of the total queries.

5.5.2.2 Hit Rate

Now we look at how effective a buddy list is. In our simulation, we count how many queries are satisfied by either checking the buddy indexes kept locally at the originating node or by directly asking its buddies. Figure 5.11 presents the hit rate curve for the 20,000 simulated queries. Within the first 2,000 queries, peers quickly gather information about its buddies and the hit rate continues to increase. At the end of query 2,000, the hit rate has exceeded 80%. For query 2,000 to 20,000, the hit rate remains at around 80%. It is larger than the hit rate we observed from the evaluation study in section II. This is because peers not only benefit from the indexes it kept locally and its buddy list, they also benefit from the indexes kept at its buddies. From this figure, we can see that with only 10 entries in the buddy list, the system achieves very high hit rate. 80% of the queries can be satisfied within one hop. And the high hit rate stays stable, which also indicates that there are few replacements for the buddy list, thus few index transfers are needed.

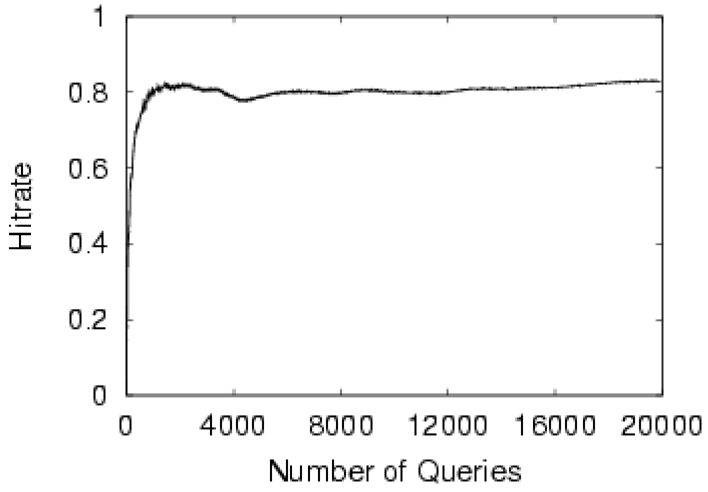


Fig. 5.11. Hit rate

5.5.2.3 Effect of clustering by interest group

The adaptation process of the buddy list provides a natural way to cluster nodes into different interest groups. We show the result of running a SVD Centroids plot over the BuddyNet links for a system of 1000 nodes using CVIZ software [25]. Every node has 1000 attributes recording whether there is a buddy list link between this node and other nodes in the system. We run SVD Centroids plot over these 1000 attributes for all the nodes in the system. In Figure 5.12, every dot in the plot represents a node in the system. We can see that nodes form several clusters. All the nodes within a cluster are tightly connected to each other. By spot-checking some nodes, we see that nodes in the same cluster share similar interests and are within short distance to each other.

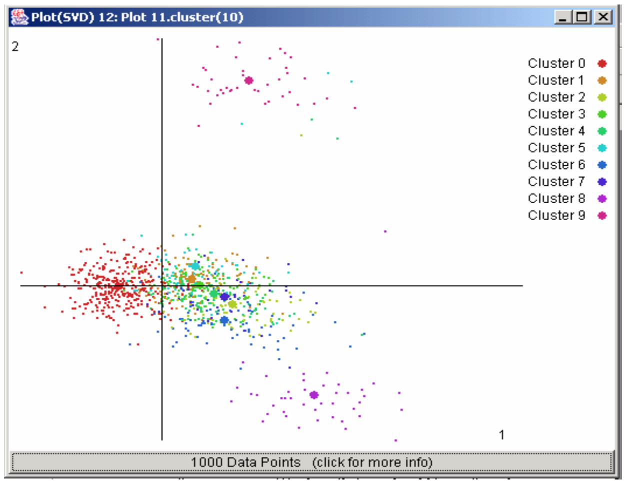


Fig. 5.12: Clustering effect

5.5.3 Factor Analysis

Our results in section 5.5.2 indicate that BUDDY_NET outperforms RANDOM_WALK and INTEREST SHORTCUTS in terms of system load and path length to locate the content. In this section, we pay special attention to how much each individual component of our algorithm contributes to the performance advantage. We show that simple addition of the one-hop indexing or the buddy list is not able to achieve the best performance. It is the *combination* of the buddy list, the buddy index and its dynamic adaptation mechanism that utilizes the distinct characteristics of P2P systems' workload and achieves large performance advantage.

We compare our system with the following two algorithms. First, we add buddy list on top of RANDOM WALK and we call it RANDOM_WALK_LIST. This is similar to INTEREST SHORTCUTS [24]. Second, we add one-hop indexing into RANDOM WALK, in this case each node keeps indexes for 10 of its neighbors. We name this method RANDOM_WALK_INDEX.

Table 5.4 shows the result of these three algorithms under the same simulation setting as in section 5.5.1.1, with 9,091 nodes running for 20,000 queries. We also list the result for RANDOM_WALK from section 5.5.1.1 for comparison.

Table 5.4. Factor analysis

	RANDOM WALK	**RANDOM WALK LIST**	**RANDOM WALK INDEX**	**BUDDY NET**
Msg/query	3,842	1,370	3,459	1122
Path length	88	31	67	28
Success rate	0.97	0.98	0.99	0.98
Hit rate	N/A	0.85	N/A	0.86

Here, we see that adding the buddy list technique to random walk improves the performance. The system load is reduced from 3,842 M/Q to 1,370 M/Q. The average path length is shortened from 88 hops to 31 hops. This simple technique contributes to about 60% of the performance improvement. But as shown in section 5.5.2.2, this simple scheme does not perform as well in dynamic situations. Adding one-hop indexing to random walk, the system load is reduced to 3,459 M/Q and the average path length is shortened to 67 hops. This represents 10% performance improvement. This is because neighbors of a peer might not share similar interests as the peer; therefore indexing its neighbors content is not very effective. As we have said, queries should be directed to a focused set of peers who share similar interests, such as its buddies. BUDDY NET outperforms both RANDOM WALK LIST and RANDOM WALK INDEX.

5.6 Conclusion

Our research originated from a simple observation: P2P systems resemble human society; therefore social capital exists in P2P systems. We start with an analysis of the interest-based relationship among peers in a P2P system. We have conducted the first evaluation study of the interest-based locality on a collected Kazaa trace. Our analysis shows that there is strong interest-based locality in Kazaa and peers can be classified into interest-based groups. Our analysis suggests that interest-based locality can be exploited by utilizing the history information of peer behaviors.

We propose a history-based peer-to-peer search algorithm and self-organizing mechanism. We designed a unique architecture for P2P systems called BuddyNet. BuddyNet integrates the buddy list, one hop indexing and dynamic adaptation techniques and utilized an improved search algorithm.

We have evaluated the BuddyNet system through simulation. We not only tested BuddyNet under ideal cases where the P2P system remains stable. We also tested it under dynamic situations where peers join and leave the network frequently. Our simulation results show that BuddyNet performs well in both situations. Under the dynamic setting where network instability is high, BuddyNet shows its superiority by utilizing the buddy list to shield peers from it. We also evaluate BuddyNet under different replication factors and system sizes. Our results suggest that BuddyNet beats RANDOM_WALK by more than 20 times in terms of reduction of system load and latency to locate content. BuddyNet give decent performance when the P2P system scales. We conduct factor analysis trying to find out how each technique and algorithm in BuddyNet affects its behavior. We found that every technique contributes to the success. The seamless integration of buddy list, one hop indexing and dynamic adaptation achieves more benefit than the simple addition of these techniques.

We have demonstrated that with simple modifications to the Gnutella protocol, the scalability problem can be overcome. Careful analysis of the behavior of peers in these P2P systems provides hints as where to find such simple modifications.

Chapter 6

Conclusion and Future Work

6.1 Summary of Dissertation

In this dissertation, we try to explore the social networks in computer systems and look for opportunities to improve the performance and efficiency for different computer systems by utilizing such social relationship. We designed and implemented computer systems in two popular domains: Topical Crawler in web crawling and BuddyNet in peer-to-peer systems. We have adopted the social capital concept from sociology and drawn analogy between large scaled computer systems and human society. We think that large scaled computer systems should have the ability to dynamically alter their structures based on the usage pattern. Nodes in such system should be assigned weights not only based on their content but also on their locations in the system and connectivity to other nodes.

Topical Crawler is an attempt to build such a system in the web crawling domain. We extracted topical information from the web and used it to build a topical map on top of the existing hyperlink structure. We built an enhanced crawler by making it aware of the topical linkage among pages. This topical crawler follows topical links side by side with hyperlinks. The pages reached by the topical crawler are integrated to generate a bigger and more accurate topical map. Results from our experiments show that the topical crawler is able to reach more on-target pages comparing to a basic topical crawler. Our enhanced crawler achieves a balance between re-crawling on-target pages and exploring new areas on the web. It shows higher efficiency even when crawling fresh areas that have never been seen before.

In peer-to-peer systems, the social relationships among nodes are more visible. We draw the analogy from peer-to-peer system to human society. Nodes in peer-to-peer systems are very similar to individuals in society. Similar to human beings, nodes in p2p system possess not only human capital but also social capital. The location of a node and its relationship with its neighbors are equally important compared to the content it owns. Our BuddyNet system is built on the belief that social linkage among nodes should be taken into account when building a successful p2p system. We collect behavior data at each node and establish direct links among nodes that interact with each other frequently. BuddyNet is an extra structure built on top of the overlay network of the system. Nodes use BuddyNet links to contact buddies and retrieve the data in a shorter delay. Our experiment results showed that from analysis of nodes' behavior data, it is fairly easy to extract and build a relational network among the nodes. Nevertheless, utilizing this social network generates superb system advantage. In our study, it not only reduced the load in the system, but also shortened the average response time for queries. The peer-to-peer system is allowed to alter its structure dynamically during the lifetime of the system based on its usage. The addition of BuddyNet allows the underlying peer-to-peer system to scale.

6.2 Future Work

Several interesting issues emerged from the study of Topical Crawler. We would like to consider them as future work.

6.2.1 Incorporating Topical Citation Matrix

Topical Citation Matrix [11] shows the affinity between different topics. A topical citation matrix records the pair-wise relevance among topics. E.g. how does the affinity between “Basketball” and “Football” comparing to the affinity between “Basketball” and “Running”? Figure 6.1 shows a 3d contour-plot of the 191×191 citation matrix displayed in their paper.

The topics are level 3 topics from Dmoz directory. The diagonal remains dominant since self-reference within a topic is strong. Finer horizontal lines emerge, showing us the most popular subtopics under the popular broad topics. Zooming down into /Arts, we see the most prominent bands are at /Arts/Music, /Arts/Literature and /Arts/Movies. Other, more meaningful target bands are found at /Computers/Security, /Recreation/Outdoors, and /Society/Issues. The plot also shows us many meaningful isolated hot spots, such as from /Arts/Music to /Shopping/Music and /Shopping/Entertainment/Recordings.

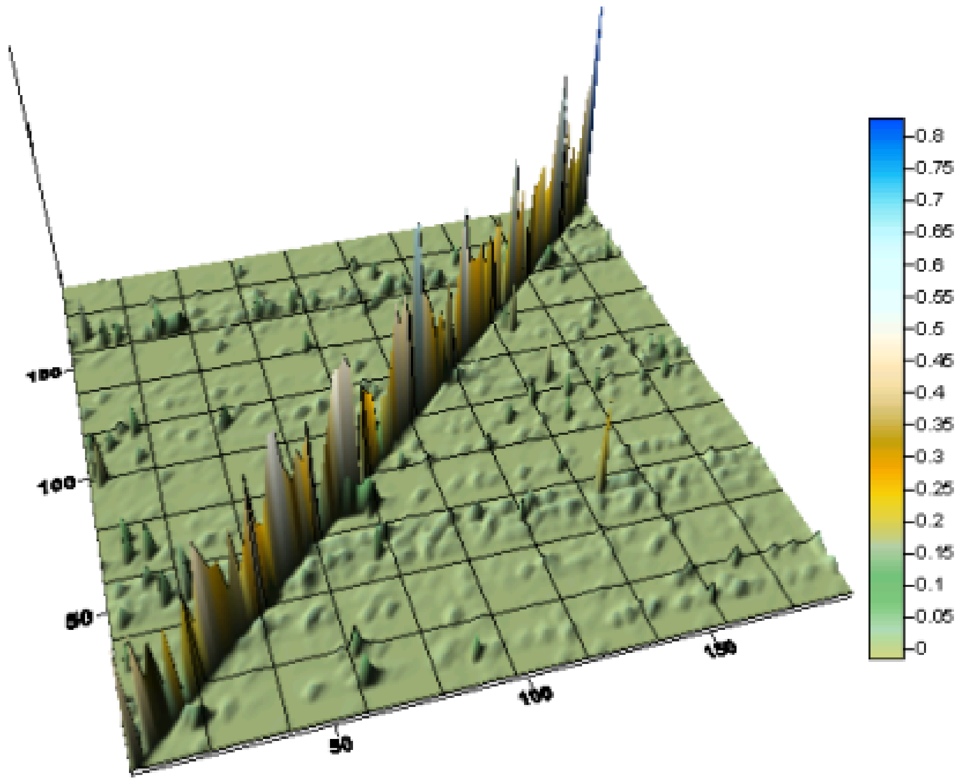


Figure 6.1: Contour plot of the 191-topic citation matrix.

We have shown that the Topical Crawler is able to achieve a better crawl result for a related topic using a previously collected topical map. The topical citation matrix gives a clear map indicating the relevance among different topics. By incorporating topical citation matrix in the topical crawler, it can behave smartly by correctly evaluating the closeness between topics.

6.2.2 Using textual features and distance measures in generating topical map

When constructing a topical map, we consider simply the topical score in our current implementation. In fact, a topical jump from one page to another by a user surfing the web may not be uniquely decided by its topical importance. Sometime, the textual features around it decide the possibility of leaping from one to another. Such textual features may contain information about the sibling nodes around it, its position among the sibling nodes and the html page structure of its ancestor. In [5], Chakrabarti *et al.* proposed to use the textual features around the crawled page as online feedback to train their apprentice crawler. They used two classifiers. The first classifier gathered broad training data for the second classifier. The second classifier, the apprentice acted as a reinforcement learner. It improved the harvest rate on the data collected by the first classifier. The classifiers derived features from the DOM tree and the text tokens on the HTML page, and used these features in deciding the priority of visit for pages in the crawler's frontier. Their initial results showed that by incorporating the features extracted from the HTML pages, the crawler is able to benefit. We believe such approach would also benefit the generation of our topical crawl map and increase the accuracy of the enhanced focused crawler. Nevertheless, extracting structural and textual features online is expensive.

Similarly, distances between on-topic pages are also important when creating the topical crawl map. Should a page link to on-topic pages with shorter hyperlink distance to it? Or should it link to pages farther away from it? We can imagine that different answers to above question would result in very different sets of crawled pages. By densely linking on-topic pages in proximity, we added extra connectivity in the local area. An enhanced crawler using this topical map would explore in more depth in the local area before deviate to some other areas. On the contrary, we could also link pages farthest away from each other. Such choice will lead the crawler to constantly crawl different neighborhoods on web. The universe of seen pages would potentially enlarge by a large amount. However, having the crawler jumping from area to area might have undesirable results. The enhanced crawler might miss pages that are easy catches otherwise. How would such choices affect the balance of re-crawling seen pages and exploring unknown areas in future crawls?

We have not evaluated these design choices in the creation of topical maps in our current implementation. We believe that interesting observations and test results will emerge from playing with these design choices. Our knowledge of the vast World Wide Web is still limited. We are just starting to look at the Web through a magnifying glass. We have designed tools to sample the web, poke at its structure in order to understand it better and utilize it more efficiently.

Bibliography

[1] Rennie, J. and McCallum, A.K.: Using Reinforcement Learning to Spider the Web Efficiently. In *Proceedings of ICML99 Workshop*, 1999.

[2] Diligenti, M., Coetzee, F.M., Lawrence, S., Giles, C.L. and Gori, M.: Focused Crawling Using Context Graphs. In *Proceedings of the 26th International Conference on Very Large Databases*, 2000.

[3] Cho, J., Garcia-Molina, J., Page, L.: Efficient Crawling through URL Ordering. In *Proceedings of 7th World Wide Web Conference*, 1998.

[4] Davison, B.D.: Topical Locality in the Web. In *Proceedings of the 23rd Annual International Conference on Research and Development in Information Retrieval*, July 2000.

[5] Chakrabarti, S., Punera, K. and Subramanyam, M.: Accelerated Focused Crawling through Online Relevance Feedback. In *Proceedings of the 11th International World Wide Web Conference*, 2002.

[6] Aggarwal, C., Al-Garawi, F., and Yu P. :Intelligent Crawling on the World Wide Web with Arbitrary Predicates. In *Proceedings of the 10th International World Wide Web Conference*, Hong Kong, May 2001.

[7] Menczer, F., Pant, P., Ruiz, F. and Srinivasan, P.: Evaluating Topic-driven Web Crawlers. In *Proceedings of the 24th Annual International ACM* SIGIR Conference, New Orleans, September 2001.

[8] Fetterly, D. Manasse, M. Najork, M. and Wiener, J.: A Large-Scale Study of the Evolution of Web Pages. In *Proceedings of the 12th International World Wide Web Conference*, May 2003.

[9] Broder, A., Kumar, R., Maghoul, F., Raghavan, P., Rajagopalan, S., Stata, R, Tomkins, A. and Wiener, J.: Graph structure in the Web. In *Proceedings of the 9th International World Wide Web Conference*, 2000

[10] Cho, J. and Garcia-Molina, H.: The Evolution of the Web and Implication for an Incremental Crawler. *Stanford University Technical Report*, 1999

[11] Chakrabarti, S., Joshi, M.M., Punera, K. and Pennock, D.M.: The Structure of Broad Topics on the Web. *In Proceedings of the 11th International World Wide Web Conference*, May 2002.

[12] Chakrabarti, S., van den Berg, M. and Dom, B.: Focused Crawling: A New Approach to Topic-Specific Web Resource Discovery. In *Proceedings of the 8th International World Wide Web Conference*, May 1999.

[13] Cho, J. and Garcia-Molina, H.: Estimating Frequency of Change. *Stanford University Technical Report*, 2000

[14] Kumar, R., Raghavan, P., Rajogopalan, S. and Tomkins, A.: Trawling the Web for Emerging Cyber-communities. In *Proceedings of 8th International World Wide Web Conference*, May 1999.

[15] McCallum, A.: Bow: A Toolkit for Statistical Language Modeling, Text Retrieval, Classification and Clustering. http://www.cs.cmu.edu/~mccallum/bow/, 1998.

[16] Mukherjea, S.: WTMS: A System for Collecting and Analyzing Topic-specific Web Information. *WWW9/ Computer Networks*, 33(1-6): 457-471, 2000.

[17] H. Leiberman, C. Fry, and L. Weitzman. Exploring the Web with Reconnaissance Agents. *CACM*, 44(8): 69-75, August 2001

[18] Open Directory Project. http://www.dmoz.org

[19] Kleinberg, J.: Authoritative Sources in A Hyperlinked Environment. In *Proceedings of the 9th ACM-SIAM Symposium on Discrete Algorithms*. January 1998

[20] Lv, Q., Cao, P., Cohen, E., Li, K., and Shenker, S. Search and Replication in Unstructured Peer-to-Peer Networks. In *Proceedings of ICS 2002,* June 2002.

[21] Cohen, E., Fiat, A., and Kaplan, H. Associative Search in Peer-to-Peer Networks: Harnessing Latent Semantics. In *Proceedings of INFOCOM 2003,* March 2003.

[22] Chawathe, Y., Ratnasamy, S., Breslau, L., Lanham, N., and Shenker, S. Making Gnutella-like P2P Systems Scalable. In *Proceedings of ACM SIGCOMM 2003,* August 2003.

[23] Leibowitz, N., Pipeanu, M., and Wierzbicki, A. Deconstructiing the Kazaa Network. In Proceedings of *The Third IEEE Workshop on Internet Application,* June 2003.

[24] Sripanidkulchai, K., Maggs, B., and Zhang, H. Efficient Content Location Using Interest-Based Locality in Peer-to-Peer Systems. In *Proceedings INFOCOM 2003,* March 2003.

[25] The Free Network Project. In *http://freenet.sourceforge.net,* 2001

[26] Krishnamurthy, B., Wang, J., and Xie, Y. Early Measurements of a Cluster-based Architecture for P2P Systems. *In Proceedings of ACM SIGCOMM Internet Measurement Workshop 2001*, November 2001.

[27] Sariou, S., Gummadi, P., and Gribble, S. A Measurement Study of Peer-to-Peer File Sharing Systems. In *Proceedings of MMCN 2002,* January 2002.

[28] Tang, C., Xu, Z., and Dwarkadas, S. Peer-to-Peer Information Retrieval Using Self-Organizing Semantic Overlay Networks. In *Proceedings of SIGCOMM 2003,* August 2003.

[29] Gummadi, K., Dunn, R., Saroiu, S., Gribble, S., Levy, H., and Zahorjan, J. Measurement, Modeling, and Analysis of a Peer-to-Peer File-Sharing Workload. Appeared in *Proceedings of SOSP-19*, October 2003.

[30] Ratnasamy, S., Francis, P., Handley, M., Karp, R., and Shenker, S. A Scalable Content-addressable Network. In *Proceedings of SIGCOMM 2001*, August 2001.

[31] Stoica, I., Morris, R., Karger, D., Kaashoek, F., and Balakrishnan, H. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *Proceedings of SIGCOMM 2001,* August 2001.

[32] Zhao, B. Kubiatowicz, J., and Joseph, A. Tapestry: An Infrastructure for Fault-tolerant Wide-area Location and Routing. *Tech. Report, University of California, Berkeley*, 2001.

[33] Rowstron, A., and Druschel, P. Pastry: Scalable, Distributed Object Location and Routing for Large-scale Peer-to-peer Systems. In *Proceedings of MIDDLEWARE 2001,* November 2001.

[34] Konrad, R. Napster Among Fastest-growing Net Technologies, CNET news.com. In *http:// news.cnet.com/2100-1023-246648.html*, October 2000.

[35] Ratnasamy, S., Hellerstein, J., Shenker, S. Range Queries over DHTs. In *Intel Research Technical Report, IRB-TR-03-011,* June 2003.

[36] KaZaA File Sharing Network. KaZaA. In *http://www.kazaa.com/,* 2002.

[37] Ripeanu, M., and Foster, I. Mapping the Gnutella Network – Macroscopic Properties of Large-scale P2P Networks. *IEEE Internet Computing Journal, 6(1)*, 2002.

[38] Bawa, M., Manku, G., and Raghavan, P. SETS: Search Enhanced by Topic Segmentation. In *Proceedings of SIGIR 2003*, July 2003

[39] Ebay Online Marketplace. Ebay. In *http://www.ebay.com,* 2003

[40] Plonka, D. Napster Traffic Measurement. Available at http://net.doit.wisc.edu/data/Napster, March 2000

[41] Saroiu, S., Gummadi, K.P., Dunn, R.J., Gribble, S.D. and Levy, H.M. An Analysis of Internet Content Delivery Systems. In *Proceedings of OSDI 2002,* Dec 2002

[42] Burt, R. S. The Network Structure of Social Capital. *Research in Organization Behaviour, 2000*

[43] Adler, P and Kwon, S. Social Capital: the Good, the Bad, the Ugly. *Pp. 89-115 in Knowledge and Social Capital. 2000*.

[44] Baker, W. Achieving Success through Social Capital. San Francisco, CA, *Jossey-Bass. 2000*

[45] Burt, R. Bandwidth and Echo: Trust, Information, and Gossip in Social Networks. In *Integrating the Study of Networks and Markets*. New York: Russell Sage Foundation, 2001

[46] Contractor, N, Whitbred, R., Fonti, F., Hyatt A., O'Keefe, B. and Jones, P. Structuration Theory and Self-organizing Communication Networks. In *Organization Science Winter Conference,* 2000.

[47] Erickson, B. Good Networks and Good Jobs: The Value of Social Capital to Employers and Employees. In *Social Capital,* edited by Nan Lin, Karen S. Cook and Ronand S. Burt. New York, 2001.

[48] Han, S. Churning Firms in Stable Markets. In *Social Science Research,* 21: 406-418, 1993.

[49] Labianca, G. and Brass, D. Negative Relationships in Organizations: The Case for Negative Asymmetry in Social Networks:, 2000.

[50] Lin, N. Social Networks and Status Attainment. In *Annual Review of Sociology:* 25: 467-487, 1999.

[51] Lin, N. The Position Generator: A Measurement for Social Capital. In *Social Capital,* edited by Nan Lin, Karen S. Cook, and Ronald S. Burt, New York: Aldine de Gruyter. 2001.

[52] McEvilly, B and Zaheer, A. Bridging Ties: A Source of Firm Heterogeneity in Competitive Capabilities. In *Strategic Management Journal* 20: 1133-1156, 1999.

[53] Meyerson, E. Human Capital, Social Capital and Compensation: The Relative Contribution of Social Contacts to Managers' Incomes. In *Acta Sociologica* 37: 383-399, 1994.

[54] Pennings, J. Lee, K. and Witteloostuijn, A. Human Capital, Social Capital and Firm Dissolution. In *Academy of Management Journal* 41: 425-440, 1998.

[55] Portes, A. Social Capital: Its Origins and Applications in Modern Sociology. In *Annual REview of Sociology* 24: 1-24, 1998.

[56] Sandefur, R. and Laumann, E. A Paradigm for Social Capital. In *Rationality and Society* 10: 481-501, 1998.

[57] Seidel, M. Polzer, J., and Stewart, K. Friends in High Places: The Effects of Social Networks on Discrimination in Salary Negotiations. In *Administrative Science Quarterly* 45: 1-24, 2000.

[58] Stuart, T. and Robinson, D. Network Effects in the Governance of Strategic Alliances in Biotechnology. In *Organization Science Winter Conference,* 2000.

[59] Swedberg, R. Markets as Networks. In *The Handbook of Economic Sociology,* edited by Neil J. Smelser and Richard Swedberg. Princeton, NJ: Princeton University Press, 1994.

[60] Shao, Y and Wang, R. BuddyNet: History-based P2P Search. In *27th European Conference on Information Retrieval*. Spain, 2005

[61] Shao. Y and Wang. R. Enhanced Focused-crawling through Topical Linkage. Unpublished manuscript.